



Requirements-Driven Design of Service-Oriented Interactions: An Evaluation

Ayman Mahfouz

15 June, 2010

Department of Computing
Faculty of Mathematics, Computing and Technology
The Open University

Walton Hall, Milton Keynes, MK7 6AA
United Kingdom

<http://computing.open.ac.uk>

Requirements-Driven Design of Service-Oriented Interactions: An Evaluation

Ayman Mahfouz
amahfouz@gmail.com

This report presents and evaluation of our proposed requirements-driven adaptation process for inter-enterprise service-oriented interactions [1] [2]. We evaluate our adaptation process using two case studies. The first case study we tackle builds on the medical example introduced in [2]. Using a constructed case study allows us to demonstrate our contributions in isolation from noise associated with a real world case study. The medical case study is small enough to be comprehended with little effort and yet it allows for illustrative applications of the adaptation process.

The second case study brings together a well-studied example from the literature and real world requirements for vehicle accident insurance and repair scenario. This case study has dual benefit. On the one hand, applying our approach to a widely studied example allows us to compare the approach to other approaches that have been applied to the example and benefit from prior analysis. On the other hand, the real-world scenario allows us to evaluate how our approach performs in a real-world setting.

1. Validating Our Approach Using a Constructed Example

To demonstrate the utility of our adaptation process we apply it to the running medical example [2]. We identify two new requirements that the interaction has to satisfy. We apply our adaptation process to incorporate each of these requirements into the requirements models for the interaction. Once the requirements models have been adapted we use our choreography

generation tool [3] to generate the choreography description from the adapted requirements. The starting point of our adaptation process is Figure Figure 1 which is reproduced from [2].

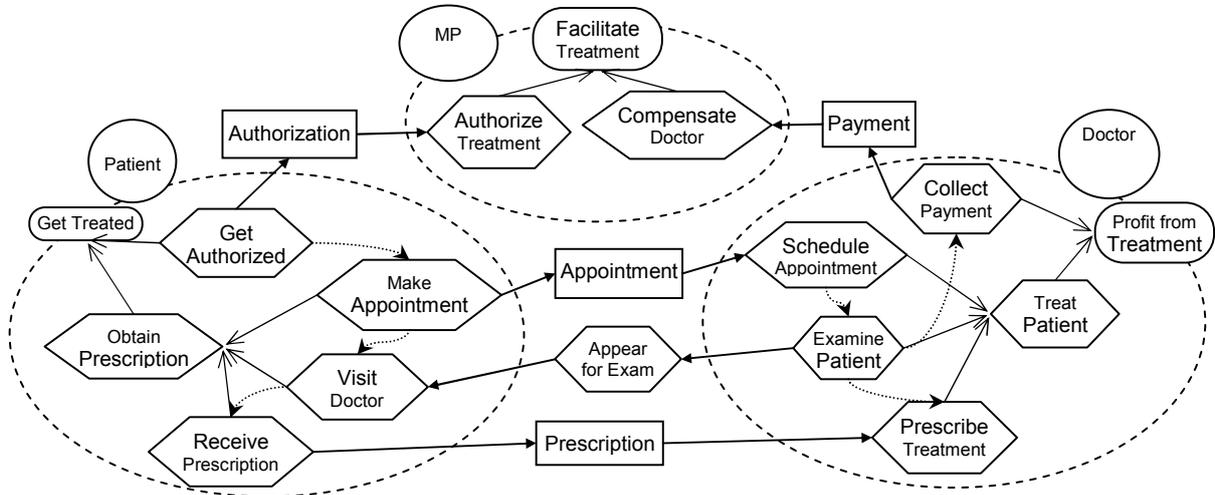


Figure 1 Combined Local-Global model of the medical interaction

1.1 First Adaptation: Validating Patient's Eligibility

The MP realizes the following need: prior to paying the Doctor they have to verify that treatment was performed on an eligible patient, i.e. a patient that has valid coverage and a valid authorization. We apply our adaptation process to incorporate this requirement into the model of **Error! Reference source not found.** and then we generate the adapted ACDL.

1.1.1. Applying the Adaptation Process

We detail the steps of applying the adaptation process. The steps are sequentially numbered for ease of reference. The process proceeds as follows:

1	MP adds an activity “Verify Eligibility” to their local model. To control the flow of the interaction, the MP makes the new activity precede “Compensate Doctor” activity so as to ensure no payment is ever made until the patient eligibility has been validated. The patient information required to approve the bill is a pre-condition on the new activity. The information is not available to any of the MP’s activities and hence the MP needs to
---	---

	suggest adding a dependency.
2	To obtain patient information and install a control point in the interaction the MP suggests adding a dismissible “Verification” dependency. “Verify Eligibility” is made the dependee and the Doctor is suggested as the depender. Additionally, the MP suggests that “Verification” precedes “Payment” to convey the precedence they added to their local model.
3	Since it in their interest to ensure that they get paid, the Doctor deems the added dependency reasonable. They accept the responsibility of the dependency as well as the added dependency precedence and proceed to explore adaptations to their local model.
4	The Doctor adds a “Request Verification” activity as the depender in the “Verification” dependency. This activity requires information about the patient authorization and coverage. Since no existing activity can provide the required information, the Doctor also adds an “Obtain Patient Info” activity and makes it precede “Request Verification”. Both new activities are made to be children of another new activity “Verify Patient Info”.
5	To comply with the precedence between “Verification” and “Payment” the Doctor must ensure that “Verify Patient Info” precede “Collect Payment”. Exploring alternatives for enforcing the precedence the Doctor finds the alternative of adding a direct precedence between the two activities not satisfactory. This alternative allows a scenario where the Doctor is denied payment, because the Patient is found to be not eligible, after they have already examined the Patient which is obviously not desirable. The Doctor finds a satisfactory alternative which is to make “Verify Patient Info” precede “Examine Patient”.
6	Doctor adds precedence from “Verify Patient Info” to “Examine Patient”.
7	Since “Obtain Patient Info” requires information from the patient, an “Eligibility”

	dependency from the Doctor to the Patient is suggested in the global model.
8	Patient accepts responsibility for the newly added dependency.
9	Patient adds a “Provide Eligibility” activity and makes it the dependee in the new dependency. The precondition on the activity is that the coverage information and authorization are available. Since these preconditions are the same as those of “Obtain Prescription”, the new activity is made a child of it. However, the Patient finds that the adapted model allows the Doctor to request eligibility information after “Visit Doctor” has been fulfilled, i.e. the Patient may be required to carry physical proof of eligibility to the Doctor’s office. In this scenario, if they forget to carry the proof they may get denied examination by the Doctor even after going through trouble of visiting their office, and hence the Patient finds this scenario undesirable.
10	To guarantee that they cannot be asked to provide eligibility after they visit the Doctor, the Patient explores alternatives for making “Provide Eligibility” precede “Visit Doctor”. They find the first alternative of adding a direct precedence satisfactory.
11	The added precedence implies the need to suggest adding precedence between the dependencies “Eligibility” and “Appear for Exam” in the global model.
12	The Doctor accepts the responsibility for the newly added dependency precedence.
13	Doctor finds that the adapted model puts them in a position where they need to cancel an appointment granted to a Patient who later fails to provide eligibility information. The Doctor explores alternatives for ensuring “Schedule Appointment” cannot be fulfilled unless “Verify Patient Info” has been fulfilled. The Doctor finds that making the second activity a child of the first achieves their need.
14	Doctor realizes they need to suggest adding the fulfillment of “Eligibility” dependency as

	a condition on the fulfillment of “Appointment” dependency.
15	Patient accepts added condition and finds the model acceptable.
16	At this point none of the participants requires further changes. The participants agree to the adapted model.

Figure 22 summarizes activities, dependencies, and links that were added/changed during the adaptation process. Given that starting point is **Error! Reference source not found.**, for brevity, the figure omits elements and links from the original model that have not been changed. For clarity, we include elements from the original model that have been linked to during adaptation, but we gray them out.

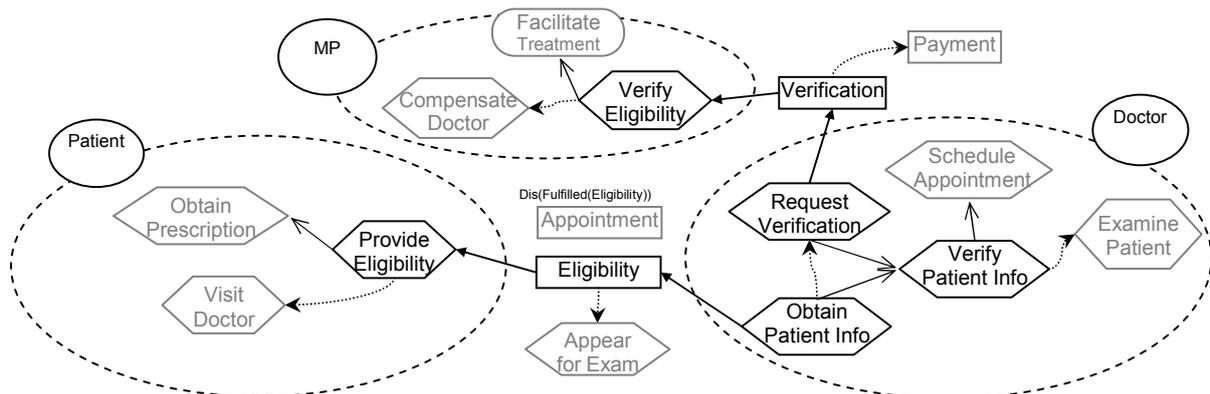


Figure 2 Summary of adaptations made to the medical example

1.1.2. Generating ACDL Description

The choreography description derived from the adapted model using our automated tool is shown in Figure 3. Note how this messaging specification ensures the patient will never obtain an appointment unless they provide proof of eligibility

Even with such a tailored example the tool has proven very useful. Prior to finishing the tool implementation, we had made an honest mistake while tackling an earlier incarnation of the

medical example [2] by listing a hand-constructed messaging specification that was inconsistent with the corresponding requirements model. The mistake was only revealed after running the tool on the requirements model, long after the work has been published.

```
Sequence
  Send AuthorizationRequest From Patient To MP
  If (AilmentCovered)
    Send AuthorizationResponse From MP To Patient
  Else
    Sequence
      Send AuthorizationRejected From MP To Patient
      Fail 'NOT AilmentCovered'.
  While (NOT AppointmentObtained)
    Sequence
      Send AppointmentRequest From Patient To Doctor
      Send EligibilityRequest From Doctor To Patient
      Send EligibilityResponse From Patient To Doctor
      If (SlotAvailable)
        Send AppointmentResponse From Doctor To Patient
      Else
        Send AppointmentRejected From Doctor To Patient
        Continue
    Parallel
      Sequence
        Send PaymentRequest From Doctor To MP
        Send PaymentResponse From MP To Doctor
      Send PrescriptionResponse From Doctor To Patient
```

Figure 3 Generated ACDL for the adapted medical example

1.1.3 Discussion

Whereas physical activities are not directly represented in the resulting messaging specification, performing the adaptation to the requirements models allowed us to take into account physical activities both as:

- a. Constraints: The need to verify eligibility before the Patient visits the Doctor played a role in deciding the order of messaging activities.
- b. Alternatives to messaging for realizing dependencies: The alternative of carrying physical proof of eligibility to the Doctor’s office was considered but found not satisfactory.

It can be argued that the Doctor could have realized the need to make “Verify Patient Info” precede the fulfillment of “Schedule Appointment” in step 5 of the adaptation rather than in step 13. However, we went with the longer adaptation scenario to show that omissions in specification of participant’s requirements may derail the process. Had the Doctor realized their need earlier, we would have ended up with a simpler adapted model.

However, the collaborative nature of the process allows each participant to incorporate requirements from their point of view, and hence shields them from one participant's mistake assuming that all changes are propagated to the global model. In this case the Patient was able to add a necessary precedence guard in step 10.

Nevertheless, after the Doctor corrected their specification the adapted model produced by the longer path is still correct. In fact, our automated tool realizes the redundancy in the final model, e.g. the precedence between "Provide Eligibility" and "Visit Doctor", and manages to generate the same ACDL specification from the final model and a model resulting from the adaptation process taking a shortcut in step 5.

It is also to be noted that even when all requirements of all participants have been included, the process may lead multiple solutions. For example, the Doctor could have added precedence from "Verify Patient Info" to "Schedule Appointment" and the process would have resulted in a model where the Patient provides their eligibility information before they even request and appointment.

1.2 Second Adaptation: Limiting Outstanding Balance

The Doctor realizes the need to limit the outstanding balance for every MP they deal with. The outstanding balance is the total of all bills that have been issued to the MP but not paid yet. Like we did in the first example, we apply our adaptation process to incorporate the new requirement into the model of **Error! Reference source not found.**

1.2.1 Applying the Adaptation Process

Since the outstanding balance increases whenever a bill is issued, the direct way to limit the balance is to constrain the issuance of bills. That is, prevent the creation of a "Payment" dependency if the condition Φ holds, where Φ represents "within credit limit". The process proceeds as follows:

1	Doctor explores alternatives for constraining the creation of "Payment" dependency as in (Error! Reference source not found.). The alternatives of adding Φ as a creation
---	---

	<p>condition on “Payment” or “Collect Payment” are found to be undesirable. These alternatives allow a state where the Doctor provides treatment to the Patient only to find later that they will not bill the MP for their effort because the credit limit has been reached. The Doctor proceeds to exclude a few other similarly unsatisfactory alternatives.</p>
2	<p>Doctor finds a satisfactory alternative which is to add Φ as a pre-condition on the fulfillment of “Schedule Appointment”. To make Φ a necessary condition for the fulfillment of the activity, a new child activity “Check Limit” is added to it, and Φ is made a pre-condition on the creation of “Check Limit” (Error! Reference source not found.).</p>
3	<p>Since the pre-condition on fulfillment of “Schedule Appointment” has changed and since the activity is responsible for fulfilling a dependency (Error! Reference source not found.), the Doctor suggests changing the dismissibility condition of the “Appointment” dependency. The new condition, Φ, is added to the conjunction of dependency dismissibility conditions.</p>
4	<p>Now the Patient has a decision to make. The Patient may object to the new dismissibility condition as it is not their fault that their MP is not paying the bills. If they object, the process will terminate in failure since the Doctor has exhausted the exploration of alternatives. If they agree the process continues.</p>
5	<p>The Patient propagates the change in “Appointment” dismissibility condition to “Make Appointment” activity (Error! Reference source not found.).</p>
6	<p>The Patient explores alternatives to guarantee that they can re-attempt to obtain an appointment. Since “Appointment” dependency already has a “many” cardinality the Patient finds no need to make further changes.</p>

None of the participants requires further changes, they agree to the adapted model.

1.2.2 Generating ACDL Description

The generated ACDL is shown in Figure 4. Note how the response to appointment request is now guarded by the condition on the credit limit. Note also that the creation condition of “Check Limit” has produced a redundant (but correct) check. In this case, it was sufficient to change the dismissibility condition of the dependency and forgo adding a local condition.

```
Sequence
  Send AuthorizationRequest From Patient To MP
  If (AilmentCovered)
    Send AuthorizationResponse From MP To Patient
  Else
    Sequence
      Send AuthorizationRejected From MP To Patient
      Fail 'NOT AilmentCovered'
  While (NOT AppointmentObtained)
    Sequence
      Send AppointmentRequest From Patient To Doctor
      If (SlotAvailable_AND_WithinCreditLimit)
        Sequence
          Send AppointmentResponse From Doctor To Patient
          AppointmentObtained = true
      Else
        Sequence
          Send AppointmentRejected From Doctor To Patient
          Continue
    Parallel
      Sequence
        Send PaymentRequest From Doctor To MP
        Send PaymentResponse From MP To Doctor
      Send PrescriptionResponse From Doctor To Patient
```

Figure 4 ACDL incorporating the credit-limit check adaptation

1.2.3 Discussion

Had the Patient rejected the suggested adaptation, the process may have ended in failure. The condition under which the appointment can be cancelled is not relevant to the Patient and they can reasonably object to it. The multi-participant nature of the interaction caused the Patient’s interests to be affected by the relation between the MP and the Doctor.

In reality, the Patient's opinion may not have the same weight as the other participants and they could end up being forced to accept the adaptation. However, our process does not account for such a situation and rather regards all participants having an equal say.

This adaptation demonstrated an example of a constraint that is independent of the interaction instance (i^3). In effect, the Doctor added a "maintained" goal whose condition is that the MP's credit is kept under the limit. The condition was used to constrain the instantiation of the "Payment" dependency as a "preventive" measure.

The cross-interaction nature of the condition to be maintained poses non-trivial problems inherent with state-sharing problems. The credit limit may change during the execution of an interaction instance due to events, in other interaction instances, that are not synchronized with the events in this instance. In this case, the goal has to be allowed to "toggle" rather than be kept "maintained", i.e. the MP may exceed their credit limit before the Doctor starts to reject appointments.

Finally, a viable adaptation alternative was not discovered by our process. The MP could have pre-conditioned issuing authorizations to patients on their credit limit with the Doctor. In this case, the MP will not issue authorization to a Patient if they risk exceeding their credit limit. To the Patient, rejecting a request for authorization is somewhat more acceptable than rejecting an appointment after an authorization has been obtained.

The reason this alternative was not discovered is that the search for alternatives in our application of the process is driven by local needs. Neither the MP nor the Patient had a need that drove the search for alternatives to constrain the fulfillment of "Authorization" dependencies. This can be remedied by intermediating the regulatory agency in the exploration of alternatives. The regulatory agency would prompt participants to adapt their local models based on requirements that are only relevant from other participants' local view.

It is to be noted however that Doctor cannot solely rely on the MP rejecting authorization when the credit limit is reached. The Doctor has to guard against the case where the MP fails to reject an authorization by rejecting the appointment.

2 Validating Our Approach Using a Real-World Scenario

The second case study we tackle combines a well-known “Vehicle Repair” example from the literature with requirements drawn from real world documents that regulate the vehicle insurance business. Our goal is to adapt the literature example to comply with the real-world requirements using our approach. First, we introduce the requirements as they are described in the literature, then we analyze documents that regulate the vehicle insurance business in North America, and finally we adapt the literature example to some of the requirements stated in the real-world documents.

2.1 Vehicle Repair Case Study Background

Government agencies overseeing the insurance business put forward rules to protect the interests of vehicle owners as well as all parties involved in vehicle insurance and repair. The vehicle repair case study tackles interaction for repairing an insured vehicle following an accident.

2.2 Requirements for Vehicle Repair from Literature – AGFIL Case

The original specification of the vehicle repair interaction was originally outlined in the AGFIL case study [4]. The case study was later revisited in [5], [6], and [7]. In what follows we will go over the original specification then we apply the results of the analysis conducted in [5] to create Tropos requirements models for the interaction.

Where the Tropos models of [5] diverge from the original specification, we stick with the original. On the other hand, where the Tropos models are under-specified we make reasonable assumptions to fill in the gaps, we state all the assumptions we make and rationalize them.

We refer to literature case, both original specification and the Tropos models corresponding to that specification, as the AGFIL case for short.

2.2.1 Original Specification of Vehicle Repair Scenario

Figure 5 depicts the original AGFIL use case which is reproduced from [5].

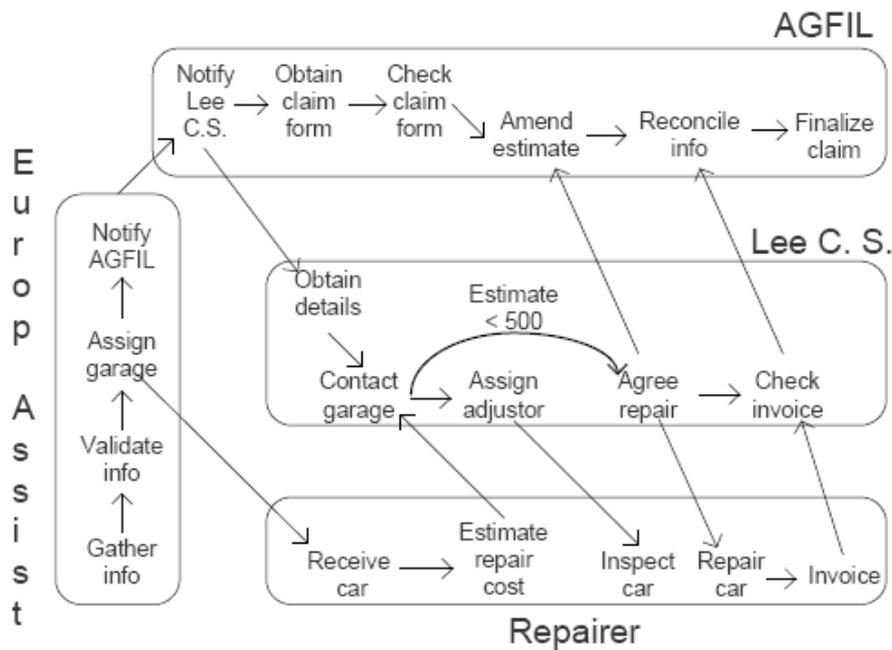


Figure 5 Original AGFIL flow specification.

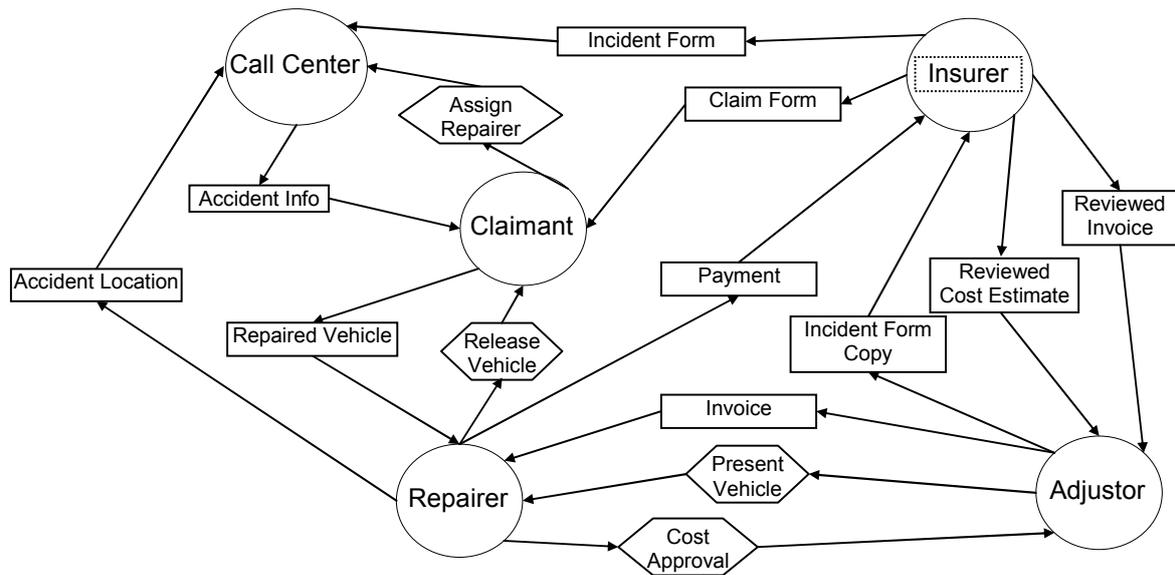
AGFIL is an insurance company that covers repair of vehicle damage incurred by policy holders. AGFIL provides claim reception and vehicle repair services to the policy holders. Additionally, AGFIL needs to assess claims to protect itself against fraud. AGFIL uses its partners, Europ Assist (EA), Lee Consulting Services (CS), and various repairers, for executing these tasks. EA provides a help-line to policy holders for reporting a claim, and directs them to an approved repairer facility. Lee CS performs damage assessment and presents invoices to AGFIL on behalf of the repairers. Several approved repairers provide repair services at their shops. AGFIL makes decisions on claim approvals, and provides payment to repairers.

2.2.2 Global View – High-level Role-Dependency Diagram

The interaction involves five roles: Claimant, Insurer, Adjustor, Repairer, and Call Center. In the original specification the role of the Claimant was mostly ignored after reporting the accident, because the case study was focusing on the Insurer business process. However, the global requirements are incomplete unless the dependencies between the Claimant and other

roles are included. The global context of the interaction is depicted in Figure 6. Notice how the context limits the scope of the interaction to the claim processing and vehicle repair. Signing up for insurance, collecting insurance premium, paying adjustor's fees, etc. are all excluded from the interaction, which is consistent with the original specification.

Among the total of 14 dependencies between the roles 3 are physical by nature since they involve hauling the vehicle, presenting it to the adjustor for inspection, and performing repairs on it. We also chose to have the "Accident Info" dependency be fulfilled via phone. It is the more common case that the Claimant reports the accident via phone, e.g. on the accident site, and it is also consistent with the original specification. Otherwise, the rest of the dependencies are fulfilled via messaging. The choices of whether each dependency is a notification or a request-response are drawn from the original requirements document.



Dependency	Classification
Accident Info	Physical (phone call)
Assign Repairer	Notification
Accident Location	Notification
Repaired Vehicle	Physical
Release Vehicle	Physical
Incident Form	Notification
Incident Form Copy	Notification

Dependency	Classification
Claim Form	Request-Response
Present Vehicle	Physical
Reviewed Estimate	Notification
Cost Approval	Request-Response
Invoice	Notification
Reviewed Invoice	Notification
Payment	Notification

Figure 6 Global model of the vehicle repair example.

2.2.3 Local Views and Combined Local-Global Diagram

We extracted flow requirements from the original specification and used them to construct local models for all the roles. Then, using the global model, we combined the local models as shown in Figure 7. All precedence links in this figure are imposed by the original flow, except for the link from “Repair Vehicle” to “Release Vehicle” which is logically necessary.

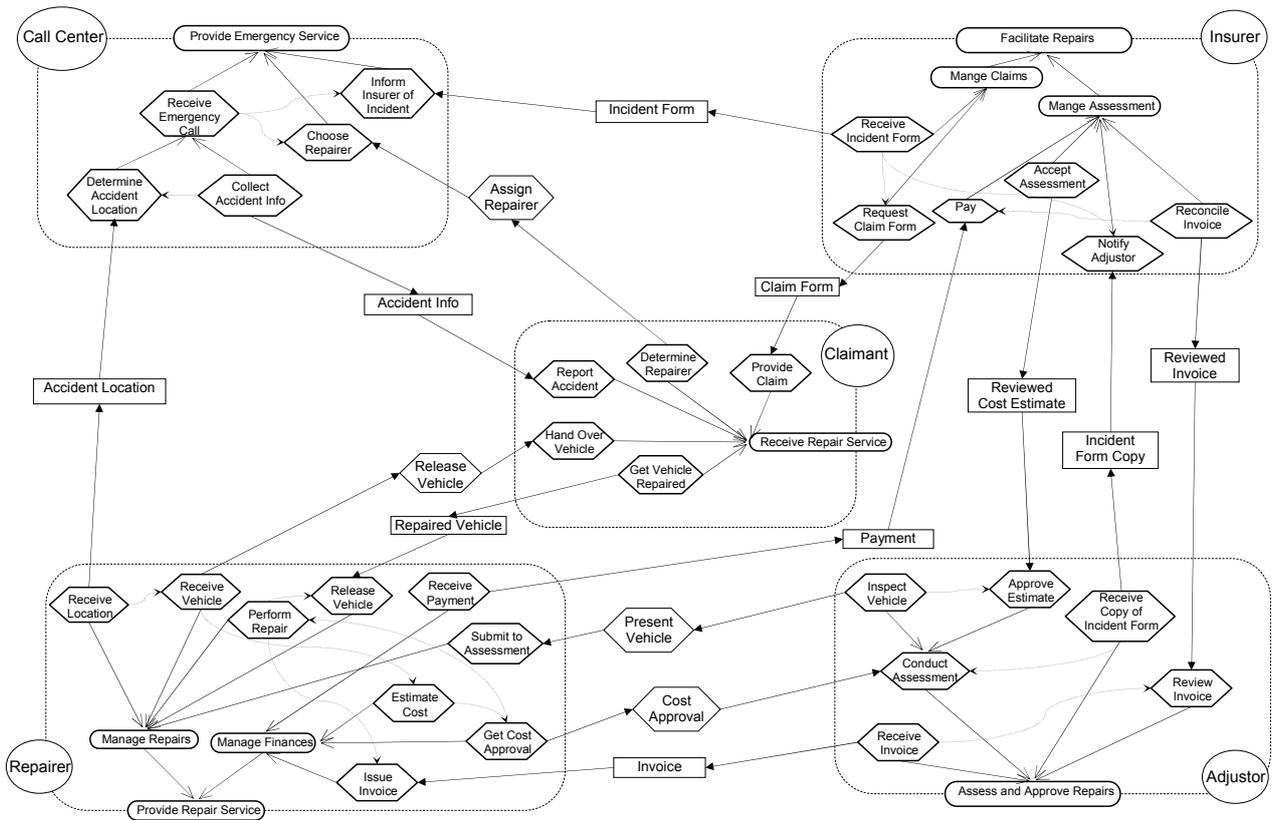


Figure 7 Combined local-global model for the vehicle repair example

2.2.4 Generated Choreography Description

By feeding the global-local model to our tool the messaging specification in Figure 8 is obtained. Worthy of notice is how the tool utilized the parallelism in the requirements model. The Call Center simultaneously notifies the Insurer of the incident and at the same time notifies the Claimant of the assigned repairer. One could argue, and indeed we wondered when we first saw the results, why wouldn't the Call Center also notify the Repairer of accident location at the same time. Going back to the requirements model we noticed that "Determine Accident Location" is a child of "Receive Emergency Call" and hence its fulfillment event has to precede its parent fulfillment. By changing around the model so that the two precedence links emanating from the parent activity are instead emanating from "Collect Accident Info" the higher level of parallelism was obtained.

```
Sequence
  Send AccidentLocationResponse From CallCenter To Repairer
  Parallel
    Sequence
      Send IncidentFormResponse From CallCenter To Insurer
      Parallel
        Sequence
          Send IncidentFormCopyResponse From Insurer To Adjustor
          IncidentFormCopyReceived = true
        Sequence
          Send ClaimFormRequest From Insurer To Claimant
          Send ClaimFormResponse From Claimant To Insurer
      Send AssignRepairerResponse From CallCenter To Claimant
    Send CostApprovalRequest From Repairer To Adjustor
  Wait Until (IncidentFormCopyReceived)
  Send ReviewedCostEstimateResponse From Adjustor To Insurer
  Send CostApprovalResponse From Adjustor To Repairer
  Send InvoiceResponse From Repairer To Adjustor
  Send ReviewedInvoiceResponse From Adjustor To Insurer
  Send PaymentResponse From Insurer To Repairer
```

Figure 8 ACDL description for the original AGFIL case

Also worthy of notice is how two independent execution paths eventually get synchronized in the generated messaging specification. The Adjustor has to have both a copy of the incident form as well as a request for cost approval from the Repairer before inspecting the vehicle. In the case the request for cost approval is received first, the “Wait Until” construct ensures that the Adjustor does not perform the inspection until the copy of incident form has been received.

2.2.5 Requirements for Vehicle Repair – Real World Documents

We analyzed public documents published by government agencies that regulate vehicle insurance and repair business in four of the most populous regions in North America: State of California Department of Insurance (CA) [8]; State of New York Department of Insurance (NY) [9]; State of Illinois Department of Insurance (IL) [10], and the Financial Services Commission of Ontario (FSCO) [11], Canada. From these documents we extracted several requirements that the specification of the AGFIL case does not satisfy. These requirements are summarized in the following table:

ID	Description
CA1	If further damage is found during the repair process, the shop will contact the insurer to get the additional cost of repairs approved. The insurer may send out an adjuster to re-inspect the additional damages.
CA2	If you do not hear from anyone, call your insurance company for assistance. If they are not responsive, or you believe there is an unreasonable delay in settling your claim, contact the Department of Insurance.
CA3	In the event that you do not agree with your company on the amount of loss either of you can demand an appraisal. Each party selects a competent appraiser. The appraisers then select an umpire.
CA4	The insurance company must stand behind the repairs of the recommended shop if the vehicle is not repaired properly.
FSCO1	When you file a claim for damage or loss, the payment made by the insurance company may be subject to a deductible, or the amount of the claim you will be responsible for paying yourself.

FSCO2	You may also be required to complete a claim form, also known as a Proof of Loss form.
FSCO3	In some cases the adjuster will want to meet with you in person.
FSCO4	As long as your insurance company approves the estimate, you may have your vehicle repaired at the repair shop of your choice.
IL1	You must immediately report all losses directly to your insurance producer or company.
IL2	Your insurance company may ask for several estimates.
IL3	Your insurance company is required to communicate with you within 21 working days after they are notified of the loss.
NY1	Your insurance company is required to obtain from you and your repair shop a “Certification of Automobile Repairs” form, in order to determine the extent to which your damaged car has been repaired.
NY2	If you fail to submit this “Certification of Automobile Repairs” form, your loss settlement on a subsequent loss may be reduced.

These documents contained a slew of requirements and we had to make some judgment about which requirements are relevant to the use case. For example, we ignored requirements that can be easily modularized into an interaction that is independent from the vehicle repair. Examples are determining who is at fault in the accident, the Insurer’s interaction with other Insurer’s to recover repair cost, getting reimbursed for medical expenditure on injuries resulting from the accident, filing a police report, and providing a rental car for the Claimant for the duration of the repairs.

2.4 Adapting the AGFIL Literature Case to Real-World Requirements

We analyze each of the requirements we have identified, we put forward suggestions for how to apply our approach to adapt the Tropos models for the AGFIL case to satisfy each of these

requirements. We detail the application of the adaptation process for a selected set of these requirements.

2.4.1 CA1: Discovering Further Damage during Repair

This requirement essentially means that repair can be performed over many iterations; during each iteration more damage may be found which is repaired in the next iteration, and so on. Whenever new damage is found the Repairer is required to get approval from the Adjustor for the new cost.

1	To represent an iteration of repair, the Repairer adds a “Perform All Repair” activity. The activity entails performing “Estimate Cost”, “Get Cost Approval”, and “Perform Repair”. The activity has a “many” cardinality and its fulfillment condition is that no more damage is found.
2	Realizing that they should not release the vehicle until all repairs are done, the Repairer adds a precedence link from “Perform All Repair” to “Release Vehicle” to replace the link that existed from “Perform Repair” to “Release Vehicle”.
3	The many cardinality of “Perform All Repair” is propagated to its children, and since “Get Cost Approval” is a depender in the “Cost Approval” dependency, the Repairer suggests adding a “many” cardinality to the dependency. Similarly, since “Perform Repair” precedes “Issue Invoice”, “many” cardinality is suggested for the “Invoice” dependency.
	Adjustor objects to having multiple invoices issued for one case and requires that all costs be aggregated at a single invoice issued at the end of repairs.
4	The Adjustor agrees to the change requiring multiple “Cost Approval”.
5	Adjustor marks “Conduct Assessment” and its child activities to have a “many”

	cardinality.
6	Since “Inspect Vehicle” inherits the cardinality of its parent, Adjustor informs Repairer they have to present vehicle and submit to assessment multiple times.
7	Repairer agrees to multiple assessments.
8	To remove the “many” cardinality from “Issue Invoice”, Repairer removes the precedence from “Perform Repairs” to “Issue Invoice”. To ensure the invoice is issued after all repairs are performed “Perform All Repairs” is made to precede “Issue Invoice”.
9	“Many” cardinality is propagated to “Reviewed Cost Estimate”, Insurer agrees to performing “Accept Assessment” multiple times, and all participants agree on changes.

The summary of changes made to the Repairer’s local model, where most of the adaptation were made, and the ACDL resulting from this adaptation is shown in Figure 9.

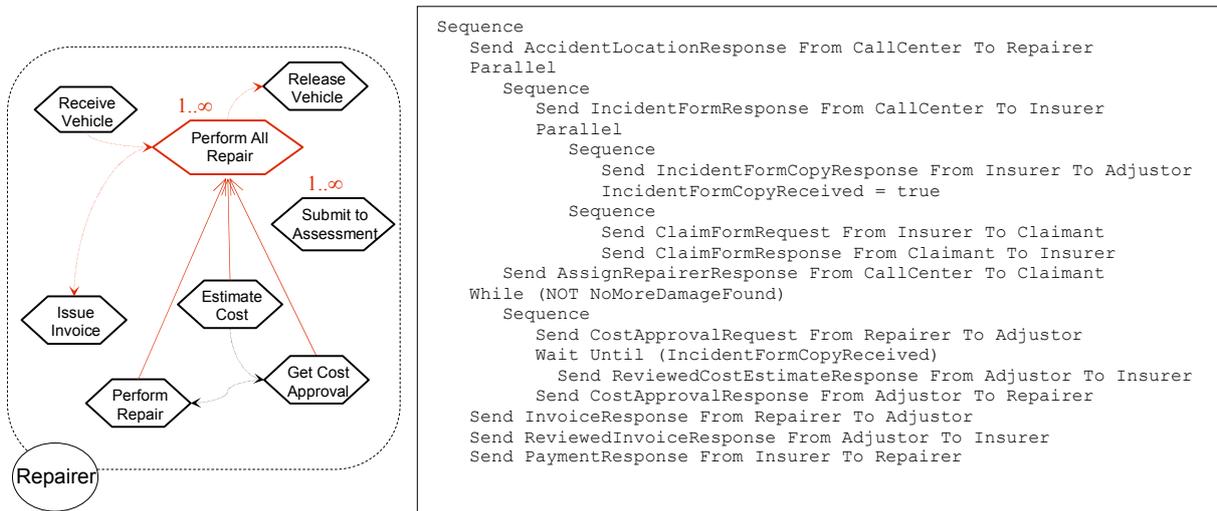


Figure 9 Adaptation to perform many repair iterations

As expected, the many cardinality resulted in the “While” repetition in the messaging. Notice also how the Adjustor performs redundant checks for the receipt of the incident form every time they are sent a cost estimate. This can be avoided by separating the check into its own activity but the Adjustor may choose to do the check every time as part of approving the cost.

2.4.2 CA2: Contacting the Department of Insurance

This requirement specifies that the Claimant has the right to contact the department of insurance to complain about unreasonable delays in claim processing. This requirement is not to be incorporated between the participants per se, but rather it is representative, as seen in the other public documents, of how the global observer gets involved in regulating the interaction. The global observer does not typically monitor every message exchange, but rather the participants report non-compliance (providing some evidence) and then the global observer can investigate, prompt non-compliant participants, and possibly enforce some penalties.

2.4.3 CA3: Involving an Appraiser and an Umpire for Arbitration

When the Claimant and the Insurer do not agree on estimates, they each need to select an appraiser that will conduct additional investigations. The appraisers choose an umpire that has the final say in case they still do not agree. This is a case where the failure of the interaction (i.e. disagreement on estimate) leads to starting another interaction with different/additional roles and a separate message flow. In this case, an “arbitration” interaction takes place between the roles of Appraiser, Umpire, and possibly Insurer and Claimant as well. Although this interaction can be designed separately in a manner similar to what we have presented, there are non-trivial questions of how to integrate it into the original interaction, i.e. coordinate the branching from the original interaction into the arbitration interaction and the return to the original interaction on completion of arbitration.

2.4.4 CA4: Insurer Stands Behind Repairs if Vehicle is not Repaired Properly

This is a high-level requirement stipulating that the Insurer is responsible for repairing the vehicle in a proper way, which may involve redoing the repairs over any number of times. There isn’t a single way to refine this requirement into activities and constraints on their execution. Presumably, it will be up to the Claimant to notify, the Insurer or Adjustor, that they are not

satisfied with repairs. The Adjustor will then probably need to carry out a new inspection and approve cost of re-doing repairs. This part of the process keeps repeating until the Claimant is satisfied with the repairs.

The suggested operationalization of this requirement can be handled in the same way we handled repetition in CA1. However, there may be an issue of how to reuse existing activities for estimating cost and conducting assessment in an optional path of the process. One possibility is to precondition the repetition of these activities on a condition that gets set when the Claimant receives the vehicle and indicate their happy with the repairs.

It may also be useful to allow physical dependencies to be dismissible. In this case, making the “Repaired Vehicle” dismissible would mean that the Claimant will refuse (reject) to pickup the vehicle if they are not satisfied with repairs. The rejection will trigger repetition of the estimation and repair part of the process.

2.4.5 FSCO1: Charging Claimant for Deductible

Under this requirement, the Claimant is obliged to pay the difference between what the Repairer charges and what the Insurer pays for repairs. It is left unspecified who charges the Claimant for the deductible so we will assume it is the Repairer, which is the likely alternative.

1	Repairer adds a “Collect Deductible” activity to their model. Since the Repairer will only know for sure how much to charge the Claimant after they have received the payment from the Insurer, they add a precedence link from “Receive Payment” to the new activity.
2	“Deductible” dependency is suggested by Repairer.
3	Claimant accepts new dependency.
4	Claimant adds “Pay Deductible” activity.
5	Repairer also realizes that it is in their interest not to release the vehicle to the Claimant

	until they have collected the deductible. They add precedence from “Collect Deductible” to “Release Vehicle”.
5	The added precedence between activities is propagated to a dependency precedence from “Deductible” to “Repaired Vehicle”.
6	Assuming that the Department of Insurance finds it legal to withhold the vehicle, the Claimant will then accept the new constraint.

Generating the ACDL from the adapted model results in two additional messages at the end of the interaction:

```
Send DeductibleRequest From Repairer To Claimant
Send DeductibleResponse From Claimant To Repairer
(RepairedVehicle fulfilled.)
```

The two messages directly follow receipt of payment by the Repairer and are followed by releasing the vehicle as the comment at the end indicates.

2.4.6 FSCO2: Optional Claim Form

The “may” in this requirement means that the Insurer may choose not to have the Claimant fill a claim form. That is, the Insurer makes this step optional in their model so that they may skip this step if they choose to. Although the Tropos models do not support that per se, we can use an “OR” refinement to represent this requirement. The Insurer adds a “Process Claim” activity, which is refined into two mutually exclusive activities: “Request Claim Form” (which already exists in the model) and an activity whose semantics mean “Process Claim without Requesting Claim Form”.

2.4.7 FSCO3: Meeting the Adjustor in Person

This statement gives the Adjustor the right to request to meet in person with the Claimant, and also get a sworn declaration from them about the accident as stated in the FSCO document. Conceivably, the Adjustor will call the Claimant and set a date and place for the meeting. The Adjustor and Claimant will then show up at the agreed date and place to discuss the claim. This translates into adding two dependencies: “Meeting Info” from the Claimant to Adjustor and

“Appear for Meeting” from Adjustor to Claimant, where the first dependency precedes the second. The Claimant will add two activities to their local model, one at the end of each dependency. Similarly, the Adjustor will add two corresponding activities to their model. The “Provide Meeting Info” is the dependee activity in the “Meeting Info” dependency (in the Adjustor’s model). This activity cannot execute until the Adjustor has been notified with the incident form. Hence, the Adjustor will explore alternatives for enforcing this precedence, e.g. by adding a precedence link between the two activities.

2.4.8 FSCO4: Claimant Chooses a Repair Shop

This is a variant of the vehicle repair interaction which appeared in all public documents we reviewed. In this variant the Claimant is responsible for hauling their vehicle to the repairer, obtaining an estimate, and sending it to the Adjustor.

There is nothing particularly challenging about this adaptation that we have not discussed before. However, it is not obvious that starting the adaptation from the model of Figure 7 is beneficial. Arguably, this interaction is sufficiently different that it may make sense to start from a clean slate. The step-by-step nature of our adaptation process makes more suited for performing incremental changes but not very effective for larger-grained ones.

2.4.9 IL1: Claimant Reports Damages Immediately to Insurer

Instead of waiting till the Insurer requests submission of claim, this requirement stipulates that the Claimant should send a claim form as soon as they can. This amounts to changing the annotation of the “Claim Form” dependency from Request-Response to Notification.

2.4.10 IL2: Claimant Required to Provide Multiple Estimates

This is an extension of the case where the Claimant chooses the repairer but the Insurer is given the right to request multiple repair estimates from any number of licensed repairers that the Claimant chooses. The repetition aspect of this requirement can be handled in the same manner as we did for requirement CA1.

However, there is non-trivial challenge where there is the need to represent multiple instances of the same role. Since there is more than one repairer involved the Repairer role is instantiated

multiple times. We currently have no way to represent this in ACDL, nor does WS-CDL. One potential solution is to represent the part of interaction where the Claimant interacts with “a” repairer as a separate modular interaction that gets invoked from the main interaction with the repairer specified as a parameter of the invocation. WS-CDL does have the notion of a sub-choreography that can be invoked from another choreography flow, and to support this kind of scenario we need to add a similar construct to ACDL.

2.4.11 IL3: Deadline for Response to a Request

We found that specifying a maximum time within which a participant is required to respond to be a common requirement. Neither Tropos nor ACDL have such a construct, but WS-CDL does have means for specifying both a timeout and a deadline. One way to extend both Tropos and ACDL to handle time-constrained responses can be worked out along the following lines:

- Add a “timeout” dependency annotation that is optionally used to specify the maximum time allowed for a response.
- Translate the dependency annotation to a “timeout” constraint that appears in some ACDL constructs, e.g. “Send” and “Sequence”.
- The latter is translated into WS-CDL “timeout” constraint.

However, a full solution to time-constrained actions is non-trivial as detailed in [12]

2.4.12 NY1: Obtaining a Certificate of Repairs from Claimant

To determine the state of the vehicle after repairs the Insurer requires the Claimant to submit a “Certificate of Repairs” that they obtain from the Repairer. This entails adding two Request-Response dependencies “Certificate of Repairs Copy” dependency from the Insurer to the Claimant and a “Certificate of Repairs” dependency from the Claimant to the Repairer. Since this document can only be produced after repairs are done, a precedence link is added from “Repaired Vehicle” to “Certificate of Repairs”.

2.4.13 NY2: Failing to Fill Certificate of Repairs Results in Reducing Next Settlement

If the Claimant fails to submit the Certificate of Repairs the Insurer is left uncertain about the state of the vehicle. If the Claimant is involved in another future vehicle the Insurer has the right

to reduce the amount they will pay for repairs. This is a case of a constraint that spans two or more instances of the interaction. The Insurer has to keep track of the state of prior repairs for each Claimant to be able to check this state when they receive a new Claimant. Adding the check is trivial, but the specification of conditions that rely on state that lives beyond the interaction instance is not well-handled in choreography languages.

2.5 Discussion

Having worked out the vehicle repair example in detail, we reflect on the results and highlight some of the main findings grouped into the following categories.

2.5.1 Capturing and Adapting Requirements

A considerably large set of requirements were easily incorporated using our approach. Adaptations that involved adding tasks or changing dependency annotations were particularly straightforward. However, our approach worked better when the required adaptations were incremental. It was less effective for larger-grained changes, which warranted re-working major parts of the requirements models.

The class of requirements that can be easily captured using our approach can be improved via some minor extensions to Tropos annotations. Annotating request-response dependencies with a timeout will allow us to handle the commonly occurring “deadline” requirement. The ability to mark an activity as optional will allow a straightforward solution to the need to represent optional activities, which came up a number of times. To a lesser extent, allowing physical dependencies dismissible will also help.

It is worthy of notice that our process supported adaptation driven by local needs, as in the healthcare example, as well as adaptation driven by the regulatory agency requirements, as in the vehicle insurance example. The two classes of adaptations were handled equally well by our approach due to our ability to alternate between the local and global models allowed in a systematic manner.

2.5.2 Building Robust Models

Even though the requirements models are built from a few primitive constructs, building robust models can become tricky as the models get larger. In earlier attempts to model the interaction, we made the mistake of placing a precedence link from one activity to another, where the two activities belonged to mutually exclusive execution branches. This leads to a deadlock since the second activity will never execute awaiting the activity on the first branch to finish, whereas only one of the branches will ever be taken. Similarly, adding precedence from an optional activity to a non optional activity is problematic. The optional activity may never be executed and the non-optional activity will then be deadlocked. Ideally, these errors can be discovered by a tool that aids architects in constructing the models.

Furthermore, requirements should not just be captured “correctly”, but they also need to be captured at the right grain. To get the detailed messaging specification, dependencies have to be very fine-grained. For instance, one might have chosen to represent the relation between the Insurer and Adjustor using a “Supervise Repairs” dependency resulting in the generation of very terse messaging specification. Breaking down dependencies into a very fine-grained level results in an increase in their number and a majority being “Notification” dependencies representing a single message rather than “Request-Response” dependencies that represent a pair of messages.

Even when requirements were captured at an appropriate fine-grained level, the choice to switch around the depender and dependee in a dependency, in a semantically equivalent way, produces slightly different messaging. For example, one could have chosen to model the “Accident Info” dependency from Call Center to Claimant as a “Collect Accident Info” task dependency in the opposite direction.

2.5.3 Refining Requirements

Further to the issue of dependency grain-level, some high-level dependencies are fulfilled indirectly, i.e. they do not correspond to any fine-grained dependencies between the same roles. For instance, the Claimant’s reliance on the Insurer to cover the cost of repairs, which would appear in a higher level AD diagram, is operationalized among other things into a dependency

between the Repairer and Insurer to reimburse the cost of repairs. A systematic refinement of the high level AD diagram, if one existed, into a fine-grained one is thus non-trivial.

In general, the refinement of a high-level requirement into detailed specification, i.e. specific activities and constraints, is not within the scope of our contributions. An analyst should use Tropos or whatever means they prefer for performing the refinement before incorporating the resulting activities and constraints into our adaptation process. This became apparent when dealing with requirements specified at an abstract level, such as CA4.

In the documents we analyzed we found a few requirements that were more declarative than flow-oriented. For instance, "... if the claimant does not provide claim information to their insurance company the claim maybe denied." This rule is cross-cutting to the interaction and may interrupt the normal flow at any point. This can potentially be represented as a parallel interaction with a guard condition, but there is a question whether there is a better representation for the interaction that can naturally incorporate these rules.

2.5.4 Specifying the Interaction Context

It may not be immediately obvious what constitutes local vs. global viewpoints in a textual description of an interaction. The original AGFIL specification was focusing on the business process of the insurance company and its partners, and thereby omitted the role of the Claimant, which is an essential role for the global view.

On the other hand, the public documents do not address in detail the relation that Insurers have with Adjustors, such as payment for the Adjustor's service, and hence it is outside the scope of the repair interaction. Perhaps this relation is regulated by other laws and for the purpose these documents they can be considered local needs. These findings emphasized the benefit of the global model as it specifies which participant inter-dependencies are part of the interaction.

The importance of the global model also derives from the importance of the global observer. Invariably, the public documents specified a dispute resolution service provided by the Department of Insurance. Claimants who disagree with their Insurer on eligibility for coverage or other problems resulting from different interpretation of the insurance policy can file complaints

through this service. Although specifying an interaction protocol is necessary for regulating the interaction, it is only part of the regulatory material. There are a lot of legal details about how to determine who is at fault in the accident, how to estimate the cash value of the car, etc. that cannot be captured in the messaging protocol.

There are some interesting issues with specifying roles and delegating responsibility. Some of the public documents specified that the Claimant can report the accident to their insurance “broker”, and that the broker takes care of carrying out the claim process. Interposing the role of the broker adds complexity as the responsibility is now more distributed. The Claimant now needs to be guarded against the broker’s failure to comply with the fulfilling their (delegated) obligations towards the interaction.

Another issue related to specifying of roles is the need to represent multiple instances of a role. For instance, this came up while trying to represent the requirement to submit estimates from multiple Repairers. One way to solve this problem is to modularize the choreography description into reusable sub-units. This will also help modularize exceptional cases, such as the arbitration interaction, into sub-choreographies that can be invoked from the main flow.

2.5.5 Generation of Messaging Specification

The ACDL generation provided some interesting feedback on the validity of the requirements model. In earlier attempts to model the interaction, we failed to capture some necessary precedence links that were imposed by the requirements. The tool alerted us that the some activities in the model were useless which prompted us to look for and locate the missing links. For instance, we had missed the precedence link from “Collect Accident Info” to “Determine Accident Location”, which lead to excluding many of the Repairer’s activities from the interaction. Figuring out which link is missing was straightforward as the latter activity was also omitted from the interaction.

Furthermore, through examining the messaging specification generated by the tool we identified incorrectly modeled requirements. Some messaging seemed to be under-constrained and this lead to identifying incorrectly placed precedence links. For instance, we originally had a

precedence link from “Estimate Cost” to “Perform Repairs” rather than from “Get Cost Approval” to “Perform Repairs” which resulted in a messaging specification that allows an invoice to be sent before the cost was approved. So even though manually constructing the entire messaging specification is hard, the partially-correct automatically-constructed messaging specification in this case proved useful as a means for requirements validation. However, these features of the tool can benefit from further automation.

References

- [1] A. Mahfouz, L. Barroca, R. Laney, and B. Nuseibeh, "Requirements-Driven Design of Service-Oriented Interactions," Open University, Milton Keynes TR2010/11, 2010.
- [2] A. Mahfouz, L. Barroca, R. Laney, and B. Nuseibeh, "Requirements-Driven Collaborative Choreography Customization," Proc. International Conference on Service-Oriented Computing (ICSOC'09), Springer, 2009, pp. 144-158.
- [3] A. Mahfouz, "CHoreography REquirements Tool (CHREQ) <https://sourceforge.net/projects/chreq/>," 2010.
- [4] S. Browne and M. Kellet, "Insurance (motor damage claims) scenario," CrossFlow Consortium Document Identifier D1.a, 1999.
- [5] P. R. Telang and M. P. Singh, "Enhancing Tropos with Commitments: A Business Metamodel and Methodology," Proc. Conceptual Modeling: Foundations and Applications, 2009,
- [6] N. Desai, A. K. Chopra, and M. P. Singh, "Amoeba: A Methodology for Modeling and Evolution of Cross-Organizational Business Processes," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 19, 2009, pp.
- [7] B. Orriëns and J. Yang, "A Rule Driven Approach for Developing Adaptive Service Oriented Business Collaborations," Proc. IEEE International Conference on Services Computing (SCC), IEEE Computer Society, 2006, pp. 182-189.
- [8] "So you've had an ACCIDENT what's next?," California Department of Insurance http://www.insurance.ca.gov/0100-consumers/0060-information-guides/0010-automobile/upload/Complete_Brochure_for_Web_Rev_05_05.pdf.
- [9] "Consumer Guide to Automobile Insurance <http://www.ins.state.ny.us/cauto.htm>," New York State Insurance Department.

- [10] "Filing an Auto Claim with Your Own Insurance Company http://www.insurance.illinois.gov/autoinsurance/auto_own_claim.asp," Illinois Department of Insurance.
- [11] "After an Auto Accident: Understanding the Claims Process," Financial Services Commission of Ontario <http://www.fscs.gov.on.ca/english/insurance/auto/>.
- [12] R. Kazhamiakin, P. Pandya, and M. Pistore, "Timed Modeling and Analysis in Web Service Compositions," Proc. First International Conference on Availability, Reliability and Security (ARES'06), 2006, pp. 840-846.