Requirements-Driven Collaborative Choreography Customization

Ayman Mahfouz[1], Leonor Barroca[1], Robin Laney[1], Bashar Nuseibeh [1]

[1]Computing Department, The Open University.
Walton Hall, Milton Keynes, MK7 6AA, UK
amahfouz@gmail.com, {L.Barroca, R.C.Laney, B.A.Nuseibeh}@open.ac.uk

Abstract. Evolving business needs call for customizing choreographed interactions. However, conventional choreography description languages provide only a partial view of the interaction. Business goals of each participant and organizational dependencies motivating the interaction are not captured in the specification of messaging. Absence of this critical business knowledge makes it hard to reason if a particular customization satisfies the goals of participants. Furthermore, there is no systematic means to assess the impact of change in one participant's process (local view) on the choreography (global view) as well as on other participants' processes. To this end, we argue for the benefits of representing choreography at the level of requirements motivating the interaction. We propose a framework that allows participants to collaborate on customizing choreographed interactions, while reconciling their competing business needs. To bridge the worlds of messaging and requirements, we employ an automated technique for deriving a choreography description from the customized requirements.

Keywords: Choreography, Requirements, Evolution, Viewpoints.

1 Introduction

A choreography description specifies the behavioral contract of participants in an electronic interaction from a neutral point of view [1]. Mutual obligations of the participants are specified in terms of constraints on the sequences of messages they can exchange. Using a choreography description language (CDL), such as WS-CDL[2], is becoming a de facto way for describing the "global" view of service-oriented interactions.

However, these languages focus almost entirely on operational aspects such as data formats and control flow. They fall short of capturing the business-domain knowledge behind the interaction. In particular, both the strategic motivations driving the participants to interact and the physical activities they are required to perform in order to fulfill their obligations are not directly represented in choreography.

This deficiency becomes critical when the choreography has to be customized to cater for emergent business needs. It is hard to ensure that a particular choice of customization to an existing choreography satisfies the business goals of participants.

To this end, we propose an approach for customizing choreographed interactions at the level of organizational requirements that motivate the interaction. Organizational requirements models capture intentions of the participants, strategic dependencies driving them to interact, and all activities they undertake during the interaction. This knowledge is essential for rationalizing customizations made to the interaction.

Since business goals of one participant (local view) are often conflicting with those of other participants, a particular choice of customization of the choreography (global view) may not be agreeable to all participants. Hence, we propose a framework that allows participants to collaborate on finding an alternative for customizing the interaction agreeable to all of them.

Our framework adopts Tropos [3] for representing organizational requirements. Tropos provides suitable notations for capturing and reasoning about a choreographed interaction in stakeholder-friendly terms. Furthermore, whereas leading CDLs have been criticized for inadequate formal grounding [4], the Tropos framework employs the formal notations of Formal Tropos (FT) [5] for precisely describing constraints that govern the behavior of participants in the interaction.

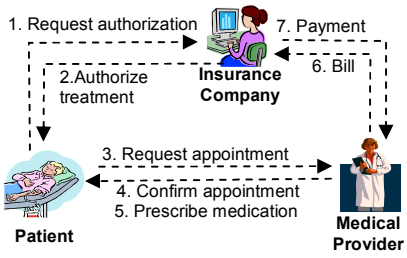
The formality of FT allows us to maintain consistency between the two representations, organizational requirements and the choreographed-messaging specification. We have previously shown [6] how organizational dependencies motivate choreographed conversations. We have also detailed how choreographed messaging can be derived from requirements [7]. In this paper we build on this work by proposing a framework that bridges global and local views of the interaction. The framework guides the collaborative customization of the interaction through an automatable process.

The rest of the paper is organized as follows: Section 2 introduces the notion of choreography customization and Abstract CDL (ACDL) using our running example. Section 3 motivates our work and gives an overview of our approach. Section 4 shows how we use Tropos to represent organizational requirements for an interaction. Section 5 outlines how we support impact analysis and traceability. Our customization process is detailed in section 6 and validated in section 7. Related work is discussed in section 8. Section 9 concludes and outlines future work.

2 Choreographed Interactions

A choreography description specifies a contract between a group of interacting roles in terms of sequences of messages they are allowed to exchange. Messaging between actual participants that play the choreographed roles at runtime has to abide by this contract. For example, consider the three roles: a patient, a medical provider (MP), and an in insurance company (IC). One potential interaction between these roles can be choreographed as follows:

A patient who needs to visit an MP must get an authorization from her IC first. When the patient receives an authorization number from the IC, she requests an appointment from the MP. After getting the confirmation the patient visits the MP to get examined by a doctor who later sends a prescription. The MP then bills the IC and gets back an electronic payment (Figure 1).




```
Sequence
   Patient Send AuthRequest To IC
   IC Send TreatmentAuth To Patient
   Patient Send AppointmentRequest To MP
   MP Send AppointmentConfirm To Patient
   Parallel
     MP Send Prescription To Patient
       Sequence
          MP Send Bill To IC
          IC Send Payment To MP
```

Fig. 1. Example choreographed medical interaction and its ACDL representation.

In this paper we use a simple pseudo-language for representing choreography in order to focus on our approach without distracting the reader by the quirky details of a particular CDL. Nevertheless, ACDL constructs are directly drawn from the leading CDL, WS-CDL [2], which makes the mapping to WS-CDL constructs almost trivial.

The three ACDL constructs used in this paper are: "**Send**" message activity to represent a message sent by a participant, a "**Sequence**" of activities that have to execute in order, and a "**Parallel**" composition of activities that can proceed simultaneously. The grammar of the language is given in Figure 2 (terminal symbols in bold). The version of ACDL used here does not include constructs for representing repetition or conditional choice between alternative execution branches.

```
Choreography → Activity
Activity → Message | Sequence | Parallel | NoOp
Message → P1 Send Message Name To P2
Sequence → Sequence Activity *
Parallel → Parallel Activity *
```

Fig. 2. Abstract Choreography Description Language (ACDL) grammar.

Message sending activities specify the participant who sends the message, P1, the participant who receives it, P2, and a literal "Message Name" that describes the message. All activities in a "**Sequence**" have to execute in order, where an activity cannot start unless the previous activity has completed. A "**Sequence**" activity is completed when the last activity in the sequence is completed. Individual branches of a "**Parallel**" can proceed concurrently. A "**Parallel**" activity is only completed when ***all*** branches are completed. The **NoOp** activity is a "do-nothing" activity. Figure 1 shows the ACDL for the medical example. Indentation represents nesting of activities.

3 Customizing Choreographed Interactions

We now motivate our work and present an overview of our approach.

3.1 The Problem

It is inevitable that the business requirements driving the interaction will change. As a result, the choreography description needs to be customized to reflect the new contract.

For example, consider an emergent need for the IC to protect itself from abuse of coverage. To protect its assets, the IC needs to ensure that it only covers treatment expenses for eligible patients. One way to achieve this goal is to require the MP to verify the insurance coverage of each admitted patient. The MP is thus required to submit the patient's insurance information to the IC so that the IC checks the validity of the patient's insurance policy. The IC will not hold itself liable for covering treatment expenses unless the MP verifies the patient information before submitting a bill. This requirement imposes a constraint on the order in which the MP performs its activities. A naïve realization of this added requirement is to have the MP send a "Verify coverage" message before sending the billing message. With conventional choreography descriptions we face two challenges:

1. It is hard to rationalize this, or any other, choice for capturing the customization without considering how well it satisfies the emergent business need.
2. It is not clear how to assess the impact of any suggested change to the choreography (global view) on the process of each participant (local view). For a participant, e.g. the patient, to agree on the change they have to assess its impact on their business goals.

These issues are exacerbated by the lack of representation of physical activities in choreography descriptions. Physical activities that are part of the interaction contract have to be taken into account when assessing a change.

3.2 Messaging Specification vs. Requirements

To rationalize a customization, it is crucial to consult problem-domain knowledge. However, choreographed messaging descriptions are operational in nature. They do not reveal much of the business rationale behind the interaction but rather focus on ***how*** the interaction is to be carried out, i.e. the control flow between activities. On the other hand, organizational requirements provide more abstract descriptions that focus on the ***why*** and ***what*** aspects of the interaction. We argue that Models of Organizational Requirements (MOR) are superior to messaging descriptions with respect to four representational areas, each of which is crucial to assessing alternative ways for capturing the required customization. These namely are:

1. Intention and Motivation. MOR for the interaction embody essential knowledge about motivations driving each participant including:

- Goals the participants wants to achieve
- Dependencies between participants enabling them to achieve their goals
- Risks and liabilities introduced by the dependencies

2. Refinement Mechanisms: MOR allow for refining high level goals into activities thereby providing rationalization of activities undertaken during the interaction. Refinement relates different levels of abstraction thereby providing traceability all the way down to the messaging specification.

3. Physical Activities. Electronic messaging is only part of the realization of the full interaction. Physical activities that the participants are obliged to perform as part of the interaction contract are not necessarily manifested in the messaging specification. For example, the patient's visit to the MP and its relation to other activities are not captured in the choreography description in Figure 1.

4. Behavioral Contract. MOR can be annotated with precise specification of participants' obligations. We employ these behavioral annotations to guide the refinement of models [7]. Furthermore, the use of formal logic enables automatic checking for the satisfaction of participants' goals.

3.3 Our Proposed Approach

We propose a framework for customizing choreographed interactions that combines the benefits of organizational requirements with the standards-based choreographed messaging descriptions.

While allowing the participants to collaborate on customizing the choreography (global view), our framework allows each participant to evaluate the impact of the customization on their individual business needs (local view). This dichotomy results in the four views (quadrants) of figure 3. We elaborate on Q1 and Q2 in section 4.

Our choreography customization framework entails: representing choreographed interactions at the level of organizational requirements models, performing required customizations to these models in a collaborative manner that benefits from the embodied domain knowledge, and deriving the resulting choreography description in an automated manner.

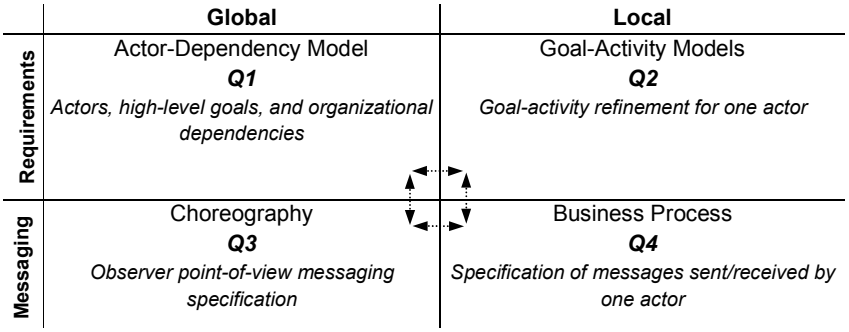


Fig. 3. The four views of our choreography customization framework.

4 Modeling Interaction Requirements

Tropos [3] is an agent-oriented software development methodology with a focus on organizational requirements at various levels of abstraction. We use Tropos for modeling interaction requirements as it provides a suitable framework for representing and reasoning about the business context for a choreographed interaction. Its models capture goals of participants (actors) in the interaction, mutual dependencies that motivate them to interact, and activities they undertake to fulfill their goals. We introduce how we model the global view of a choreographed interaction using Actor-Dependency (AD) models, how we model the local view using Goal-Activity (GA) models, and how behavioral dynamics of the model are described using FT.

4.1 Global View: AD Modeling

Actor-Dependency (AD) models provide a notation for representing the global view of the interaction at a high-level of abstraction by capturing the actors (participants) in the interaction, their high-level goals, and the inter-dependencies driving them to interact. Figure 4 is an AD model representing the medical interaction at a high-level. An actor is an active entity that performs actions to achieve its goals. The patient, the MP, and the IC are all actors. Model elements can either be internal to an actor (inside the dotted ellipse) or define dependencies whose fulfillment is delegated to other actors. An actor may depend on another for fulfilling a goal, performing an activity, or making some resource available.

A goal is a state of the world desired by one of the actors. For example, the "Get Treated" goal represents the patient's desire to get cured from an ailment. An activity is an abstraction of a course of action with well-defined pre- and post-conditions. The patient is required to perform the "Appear for Exam" activity to visit the MP's office. A resource is an informational or physical entity. For example, the "Payment" resource represents the compensation that the MP gets from the IC in return for providing services to the patient.

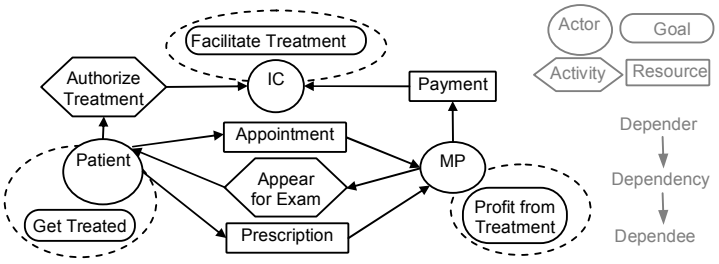


Fig. 4. Actor-Dependency model for the medical interaction.

4.2 Local View: GA Modeling

To detail the specification of the interaction, we successively refine AD models into Goal-Activity (GA) models [3]. Each GA model represents an actor's local view of the interaction. In the process, goals are refined into sub-goals and eventually realized by activities. Each actor considers and evaluates refinement alternatives based on how well they satisfy their goals [8]. Activities can be further refined into sub-activities that are either implemented by a service or carried out by a human agent.

Figure 5 shows the GA model of both the MP and the patient. Goals and activities internal to an actor are refined inside the dotted ellipse for that actor. Each actor takes responsibility for carrying out their internal activities during the interaction. For example, the "Get Treated" goal was refined into activities to get an authorization from the IC followed by getting a prescription from the MP. The latter is further refined into activities for setting up an appointment followed by visiting the MP and then receiving a prescription from the MP.

The business goals of participants may dictate some ordering of activities. For example, in the analysis process the MP realized the need to manage office schedule. Hence, the MP requires every patient to setup an appointment before they visit. Also, physical activities may impose ordering. For example, the MP has to examine the patient before prescribing treatment.

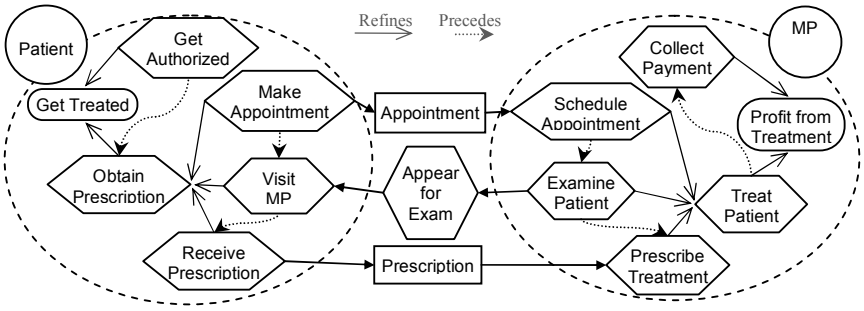


Fig. 5. Partial Goal-Activity diagram for the medical interaction.

4.3 Behavioral Specification: Formal Tropos

Behavioral obligations of participants can be captured in formal annotations used by the formal counterpart of Tropos, Formal Tropos (FT). Each activity, goal, resource, and dependency in the model is represented as an FT class, of which many instances may be created during an "execution" of the model. An execution of an FT model corresponds to a possible progression of the interaction. Model execution is useful for verifying that an interaction will proceed as designed. A partial FT specification for the "MakeAppointment" activity and the "Appointment" dependency classes is shown in figure 6, parts of which can be deduced by applying some heuristics [5].

Each class has attributes that define associations with other instances in the model. For example, the "Appointment" class has "makeApp" attribute that references the associated instance of "MakeAppointment" class.

Valid progressions of the interaction are specified by constraining the lifecycle of model elements using temporal logic. **Creation** and **Fulfillment** conditions define when an instance of a class is created (instantiated) and when it becomes fulfilled.

4.3.1 Creation. Creation of a goal or a dependency is interpreted as the moment at which the actor begins to desire the goal or need the dependency to be fulfilled. For example, an "Appointment" dependency will be created if there is an instance of "MakeAppointment" activity that needs to be fulfilled. For an activity, creation is the moment at which the actor has to start performing it. Note how FT specifies that "MakeAppointment" is created when its "super" activity, "Obtain Prescription", needs to be fulfilled thereby bridging two levels of abstraction. We use Cr(X) to denote the creation event of X.

4.3.2 Fulfillment. Fulfillment condition marks the end of the lifecycle of an instance. Fulfillment condition should hold whenever a goal is achieved, an activity is completed, or a resource is made available. For example, the "MakeAppointment" activity is fulfilled when the associated "Appointment" dependency has been fulfilled (i.e. appointment confirmation was received by the patient) whereas an instance of "Appointment" is fulfilled when the MP has completed the activity of scheduling an appointment. We use Fi(X) to denote the fulfillment event of X.

Dependency Appointment **Depender** Patient **Dependee** MP **Attribute** makeApp: MakeAppointment **Creation condition** ¬**Fulfilled**(makeApp) **Fulfillment condition** ∃ schedAp:SchedulApp (schedAp.actor = dependee ∧ **Fulfilled**(sa))	**Activity** MakeAppointment **Actor** Patient **Creation condition** ¬**Fulfilled**(super) **Fulfillment condition** ∃ a:Appointment (a.depender = actor ∧ a.makeApp = self ∧ **Fulfilled**(a))

Fig. 6. FT specification of "Appointment" and "MakeAppointment ".

5 Traceability and Impact Analysis

Our goal here is twofold: first, facilitate collaboration between participants to find a customization on which they all agree and second: systematically determine the messaging specification resulting from customization of requirements models.

5.1 Impact Analysis: Bridging Local and Global Views

To allow participants to assess the suitability of a customization (from their point of view) we must be able to determine the effect of a change in the choreography on any participant's process. Conversely, we need to determine the impact of changes in any of the participant's local model on the choreography so that other participants get to assess suggested customizations to the choreography from their point of view.

We employ dependencies to link GA and AD models. GA models explicate which specific activities are at both ends of each dependency, thereby providing linkage between the local view of each participant with the global view of the interaction.
FT precisely relates the lifecycle of dependencies to that of activities at both ends of a dependency. For example, in figure 6, note how the state of "Appointment" dependency determines the state of "MakeAppointment" activity. The patient cannot make progress on their internal process flow unless "Appointment" dependency is fulfilled. On the other hand, the "Appointment" dependency is only fulfilled when the MP have complete the "ScheduleAppointment" activity.

5.2 Traceability: Bridging Requirements to Messaging

Using FT to relate the lifecycle of activities to their "super" activity enables us to bridge requirements models to messaging specification. We exploit this traceability mechanism to show how dependencies drive the interaction thereby outlining an abstract view of the choreography [6]. For example, "Appointment" dependency indicates that the patient depends on the MP for obtaining an appointment, which implies that both actors need to interact to fulfill the dependency.

We have exploited these semantics to automate the generation of choreographed messaging from requirements models [7]. First, we infer the set of choreographed events from creation/fulfillment events of activities and dependencies. Then, we use the semantics of refinement, dependencies, and precedence between activities to come up with a partial ordering relation over these events. Finally, from the ordering relation, we generate a choreography description that satisfies the requirements [7].

Even though GA modeling details the activities of the interaction, it provides an important flexibility. It defers the choice of the medium through which activities are carried out. For example, the choreography designer may choose to include the "Prescription" in choreographed messaging or have it be fulfilled otherwise, e.g. paper documents, fax, etc. We take advantage of this by including all activities, including physical activities, in the customization process.

6 Choreography Customization Process

Bridging requirements to choreography allows us to perform required customizations to requirements models then derive the customized messaging. On the other hand, bridging the local and global views helps ensure that customizations to a choreography description do not violate the goals of any participant. Thus, our proposed customization process covers the 4 quadrants of figure 3.

The driver behind choreography customization is to satisfy an emergent business need. Several customization alternatives that satisfy this need may exist. Our process enables participants to ***collaborate*** on finding an alternative acceptable to all of them. Each participant gets to evaluate the suitability of alternatives from their local point of view as well as suggest other alternatives.

An advantage of our process is that it has no fixed starting point. Customization may start in any of the four quadrants of figure 3 and *move* between them. Consider the following example manifestation of the process:

1. Participant P1 identifies an emergent business need.
2. P1 considers a change in their GA model (which is in Q2) to fulfill that need.
3. To determine the effect of the suggested change on the global view we use dependencies to relate P1 GA model to the AD model (moving from Q2 to Q1).
4. The change in the AD model may imply (again Q1 to Q2) changes to another participant's, P2, GA model.
5. P2 evaluates suggested change from their point of view (Q2 again – but for P2).
6. P2 deems the suggested change unacceptable and suggests an alternative way for fulfilling P1's need.
7. The effect of the alternative on the AD model is worked out (Q2 to Q1).
8. A change in the AD model implies a change in the GA model of P1 (Q1 to Q2).
9. P1 agrees to the suggested alternative.
10. The choreographed messaging is then derived from the customized AD and GA models [7] (moving from Q1 to Q3).

Each step of the process involves one of the following:

1. Switch Views. To assume one of the four views of figure 3 our customization framework allows moving between its four quadrants as follows:

- Q1-Q3: Choreographed messaging constraints obtained from AD models as per [7].
- Q1-Q2: Ends of every dependency appearing in the AD model are activities appearing in a GA model, as in section 5.
- Q2-Q4: Ordering of messages sent and received by one participant is constrained by refinement and precedence between the activities of that participant as per [7].
- Q3-Q4: Messages sent/received by every participant appear in the choreographed messaging specification. For example, as in [9], [10]

2. Evaluate Alternative. Each participant needs to ensure that a suggested customization is acceptable from their local point of view. When a change is suggested to their GA model (e.g. to reflect a change in the AD model), a participant can verify that the customized model still achieves their business goals. A systematic way to evaluate a GA model is by executing it using a simulator [5] and checking whether every possible execution state is acceptable. If the participant deems one of the states unacceptable, they can then suggest an alternative customization.

3. Suggest Alternative. To aid a participant suggest an alternative customization, we provide systematic ways for finding alternatives for certain classes of customizations. For example, by bridging requirements to messaging as in section 5, we can auto-enumerate all possible alternatives for a customization that requires *adding* an event to the choreography along with an ordering constraint [6].

4. Perform Customization. Customizations that we tackle here are those that result from incremental, rather than radical, changes to requirements. Section 7 shows examples of adding a dependency, an activity, and a precedence constraint.

5. Agree on an Alternative. The customization process concludes when none of the participants objects to the candidate customization alternative. However, there is no guarantee that a solution agreeable to all participants will be found. If a point is reached where at least one of the participants objects to the last remaining candidate solution, the requested customization may be deemed unreasonable. An alternative may then be sought at a higher level requirements model, e.g. as in [3] and [8].

7 Validation

We now use the medical example to demonstrate our customization framework. Revisiting the medical example, we start the process from the original suggested customization to messaging:

Starting from the initial suggestion by the IC

1. The IC suggests a customization where they get a message asking them to verify a patient's coverage prior to receiving a bill (Q4 for IC).
2. This translates (Q4-Q3) to adding a "verify coverage" message that precedes the billing message in the customized choreography description.
3. Consequently (Q3-Q4 for the MP), the MP has to send a "verify coverage" message before sending the billing message (Q3).
4. The "verify coverage" request-response messages imply (Q3-Q1) an added organizational dependency.

Adding the "Verification" dependency and required activities

5. The "Verification" dependency is then added to the AD model (Q1).
6. To initiate the fulfillment of the dependency (Q1-Q2) the MP has to perform a "Verify Coverage" activity (Q2 for MP).
7. The new activity is added to the GA model of MP. From the original requirement imposed by the IC, the activity has to precede "Collect Payment".
8. The first candidate solution that satisfies the new imposed constraint is to have the new activity immediately precede "Collect Payment".
9. The MP analyzes the suggested solution through simulation (Q2). The MP determines that the solution allows a state where a prescription has already been sent to a patient whose insurance information has not been verified. This state is deemed undesirable because if the coverage is not eventually verified, the MP will not get paid.
10. To find an alternative point for performing the "Verify Coverage" activity, the MP explores other alternatives [6]. Rather than directly preceding the billing activity, "Verify Coverage" can be made to precede any other activity that transitively-precedes the billing activity.
11. One such alternative is to have the "Verify Coverage" activity precede "Issue Prescription". But again, an execution of the model (Q2) deems this unacceptable as it allows a state where a doctor wastes his time examining the patient only to find later that she is not covered by the IC.

12. Continuing in the same manner, the MP finds the first viable solution which is to have "Verify Coverage" precede "Examine patient".

Adding the "Coverage" dependency and required activities

13. The MP adds a "Get Coverage Info" activity (Q2) which entails (Q1-Q2) adding a "Coverage Info" dependency (Q1). The MP requests that the patient provides coverage information prior to the examination,.
14. The patient adds a sub-activity, "Provide Coverage", to "Obtain Prescription". The new activity is assigned to fulfill "Coverage Info" dependency (Q1-Q2).
15. The first point "Get Coverage Info" can be performed is right before Cr(Examine Patient) and right after Fi(Visit). This implies that the patient will physically carry the coverage information to the MP office.
16. The patient finds this option undesirable as an execution of the model (Q2 for patient) shows it allows states where the patient goes through the trouble of visiting the MP but not get examined, e.g. if verification fails due to some system outage.
17. Continuing as specified in [6], a viable solution is found where verification is made to precede the Fi(Appointment). Thus, the patient suggests providing coverage information prior to getting the appointment confirmation.

Agreeing on a customization and concluding the process

18. To add "Get Coverage Info" right before Fi(Appointment) the MP makes it a sub- activity of "Schedule Appointment".
19. The MP agrees the patient's suggestion.
20. All participants agree to the suggested solution.
21. Having agreed on a customization, the choreography messaging is then derived automatically from the Tropos models.

Figure 7 summarizes customizations made to the Tropos models. By feeding our choreography derivation tool [7] the Tropos model as input it outputs the ACDL description shown in figure 8. Note that a design decision was made to realize "Prescription" as a messaging, rather than physical, activity.

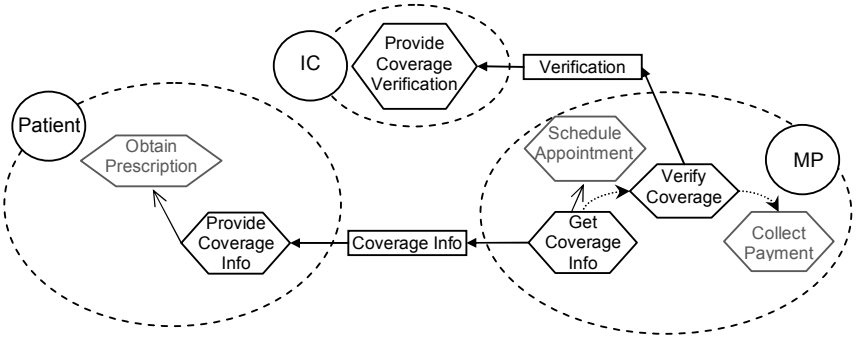


Fig. 7. Summary of the customizations made to the requirements model.

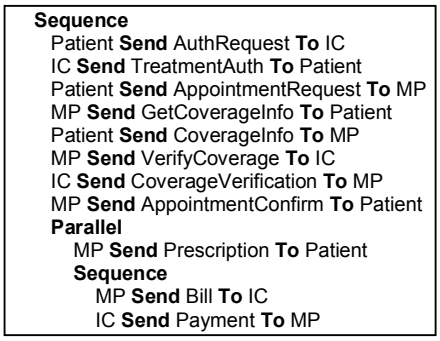


Fig. 8. Choreography description derived from the customized requirements model.

8 Related Work

Most of the research on choreography has focused on representation [11], generating process skeletons [12], and verifying the compliance of the collective behavior of a set of processes with a choreography description [13]. While highly-dynamic service interactions have been a long-sought goal [14], choreography customization is an emerging area [15] with little support for business-level reasoning [4].

Although, our work shares the spirit of attempts to integrate commitments with Tropos [16] [17], our structured customization process and automatic derivation set our approach apart, especially that it is not clear in [17] how activities can be related to messaging. The Amoeba methodology [18] for evolving cross-organizational interaction is promising, albeit it does not adequately distinguish between the local and global views of the interaction thereby obscuring the needs of each participant.

Most of the work addressing customization of service interactions focused on adapting orchestrations [19] [20] rather than choreography. More importantly, with the exception of [21], the business needs driving the interaction are not addressed.

Representing organizational requirements for distributed actors is well-established [22], and also is evolution in agent-oriented systems [23]. However, both were yet to be applied to choreographed service interactions in a way that explicates the multiple views on the interaction. Our work is consistent with the dichotomy given in [24], albeit that work does not address customization. Otherwise, relating viewpoints in service interactions was established only at the messaging level [9]. Attempts to relate choreography to business rules have also only addressed operational aspects [25].

Finally, although UML activity diagrams [26] are widely used to represent choreographed interactions, the formality and the levels of abstractions of Tropos [3] make it superior for analyzing business goals and reasoning about their satisfaction.

9 Conclusions and Further Work

Ever-changing business needs call for customizable choreography descriptions. Conventional CDLs are not well-suited for customization as they embody little of the domain knowledge required to reason about participants' goals. In particular, the business goals of participants and strategic dependencies motivating the interaction are not explicitly represented. We proposed representing choreographed interactions at the level of organizational requirements. Tropos models embody knowledge about the goals of the participants, the dependencies driving the interaction, and all activities performed during the interaction including physical activities not represented in conventional CDLs.

We proposed a framework that enables participants to collaborate on customizing the choreography (global view) while at the same time ensuring their individual business needs (local view) are satisfied. We utilized the formality of FT to analyze the impact of choreography customization on each participant's processes. We provided systematic ways for finding customization alternatives and evaluating them.

Once participants have agreed on an alternative, we use our automated technique to derive the customized messaging specification from Tropos models. Using an example, we demonstrated how our framework exploits domain knowledge embodied in requirements models to decide how the required customization is to be performed.

The generated ACDL is a skeleton that needs to be refined in a design phase, e.g. by specifying message data types. In particular, ACDL employs request-response messaging whereas more complex patterns may realistically be needed. We will exploit the FT for inferring more detailed messaging, such as repetition and branching. Furthermore, we plan formalize data flow aspects of our analysis.

References

[1] C. Peltz, "Web Services Orchestration and Choreography," IEEE Computer, vol. 36, pp. 46-52, 2003.

[2] "Web Services Choreography Description Language Version 1.0, http://www.w3.org/TR/ws-cdl-10/," W3C, 2005.

[3] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An Agent-Oriented Software Development Methodology," Journal of Autonomous Agents and Multi-Agent Systems, vol. 8, pp. 203-236, 2004.

[4] A. Barros, M. Dumas, and P. Oaks, "Standards for Web Service Choreography and Orchestration: Status and Perspectives," presented at Business Process Management Workshops, Nancy, France, 2006.

[5] A. Fuxman, L. Liu, J. Mylopoulos, M. Pistore, M. Roveri, and P. Traverso, "Specifying and analyzing early requirements in Tropos," RE Journal, vol. 9, pp. 132-150, 2004.

[6] A. Mahfouz, L. Barroca, R. Laney, and B. Nuseibeh, "Customizing Choreography: Deriving Conversations from Organizational Dependencies," presented at Enterprise Distributed Object Computing Conference (EDOC), Munich, Germany, 2008.

[7] A. Mahfouz, L. Barroca, R. Laney, and B. Nuseibeh, "From Organizational Requirements to Service Choreography," accepted for publications in SEASS'09, co-located with IWCS'09, July 6-10, Los Angeles, USA.

[8] P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani, "Formal Reasoning Techniques for Goal Models," in Journal on Data Semantics, vol. 2800, LNCS, S. Spaccapietra, S. T. March, and K. Aberer, Eds.: Springer, 2003, pp. 1-20.
[9] R. M. Dijkman and M. Dumas, "Service-Oriented Design: A Multi-Viewpoint Approach," International Journal of Cooperative Information Systems, vol. 13, pp. 337-368, 2004.
[10] J. M. Zaha, M. Dumas, A. H. M. t. Hofstede, A. P. Barros, and G. Decker, "Service Interaction Modeling: Bridging Global and Local Views," presented at EDOC,China, 2006.
[11] J. M. Zaha, A. P. Barros, M. Dumas, and A. H. M. t. Hofstede, "Let's Dance: A Language for Service Behavior Modeling," presented at OTM (1), Montpellier, France, 2006.
[12] J. Mendling and M. Hafner, "From Inter-Organizational Workflows to Process Execution: Generating BPEL from WS-CDL," presented at ACM / IEEE 8th International Conference on Model Driven Engineering Languages and Systems, Montego Bay, Jamaica, 2005.
[13] H. Foster, S. Uchitel, J. Magee, and J. Kramer, "Model-based Verification of Web Service Compositions," presented at 18 International Conference on Automated Software Engineering (ASE'03), 2003.
[14] E. D. Nitto, C. Ghezzi, A. Metzger, M. P. Papazoglou, and K. Pohl, "A journey to highly dynamic, self-adaptive service-based applications," Automated Software Engineering, vol. 15, pp. 313-341, 2008.
[15] S. Rinderle, A. Wombacher, and M. Reichert, "On the Controlled Evolution of Process Choreographies," presented at 22nd International Conference on Data Engineering (ICDE'06), Atlanta, GA, USA, 2006.
[16] A. U. M. and and M. P. Singh, "Incorporating Commitment Protocols into Tropos," in Agent-Oriented Software Engineering VI, vol. 3950/2006: Springer, 2006, pp. 69-80.
[17] P. R. Telang and M. P. Singh, "Enhancing Tropos with Commitments: A Business Metamodel and Methodology," presented at Conceptual Modeling: Foundations and Applications, 2009.
[18] 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.
[19] A. Charfi and M. Mezini, "Aspect-Oriented Web Service Composition with AO4BPEL," presented at The European Conference on Web Service, Erfurt, Germany, 2004.
[20] B. Orriëns and J. Yang, "A Rule Driven Approach for Developing Adaptive Service Oriented Business Collaborations," presented at IEEE International Conference on Services Computing (SCC), Chicago, Illinois, USA, 2006.
[21] R. Kazhamiakin, M. Pistore, and M. Roveri, "A Framework for Integrating Business Processes and Business Requirements," presented at Enterprise Distributed Object Computing Conference (EDOC'04), Monterey, California, USA, 2004.
[22] E. Yu, "Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering," presented at 3rd IEEE Int. Symp. on Requirements Engineering, Washington D.C., USA, 1997.
[23] J. Khallouf and M. Winikoff, "Goal-Oriented Design of Agent Systems: A Refinement of Prometheus and its Evaluation," International Journal Agent-Oriented Software Engineering, vol. 3, pp. 88-112, 2009.
[24] P. Traverso, M. Pistore, M. Roveri, A. Marconi, R. Kazhamiakin, P. Lucchese, P. Busetta, and P. Bertoli, "Supporting the Negotiation between Global and Local Business Requirements in Service Oriented Development," ITC-irst, Trento, Italy 2004.
[25] A. Berry and Z. Milosevic, "Extending Choreography With Business Contract Constraints," International Journal of Cooperative Information Systems (IJCIS), vol. 14, pp. 131-179, 2005.
[26] V. Vitolins and A. Kalnins, "Semantics of UML 2.0 Activity Diagram for Business Modeling by Means of Virtual Machine," presented at EDOC'05, Enschede, The Netherlands, 2005.