Requirements-Driven Design of Service-Oriented Interactions

Ayman Mahfouz, *Webalo*

Leonor Barroca and Robin Laney, *The Open University, UK*

Bashar Nuseibeh, *The Open University, UK, and Lero, Ireland*

A design framework helps modularize and reconcile stakeholder concerns in interenterprise service interactions. It's supported by a tool that automatically generates messaging protocols from requirements models.

Service-oriented architecture (SOA) enables interenterprise service interactions. Services provide platform-independent abstractions around software systems, thereby enabling interoperability between heterogeneous systems.[1] Several languages are emerging as standards for describing interfaces and interaction protocols that specify service-oriented systems:

- Web Services Description Language (WSDL, www.w3.org/TR/wsdl) for specifying message types and service operation signatures;
- Web Services-Choreography Description Language (WS-CDL, www.w3.org/TR/ws-cdl-10) for specifying multiparticipant messaging protocols from a global point of view between a set of abstract roles;[2]
- Web Services-Business Process Execution Language (WS-BPEL, www.oasis-open.org/committees/wsbpel) for coordinating service messaging from a single-participant point of view;
- Web Services-BPEL Extension for People (BPEL-4People, www.oasis-open.org/committees/bpel4people) for specifying human tasks within a WS-BEPL specification; and
- SOA-Modeling Language (SoaML, www.omg.org/spec/SoaML) for specifying high-level service architectures, business information models, and service component architectures.

Even though these languages provide interoperable specifications, they have two serious limitations with respect to designing interenterprise interactions. First, they focus on operational aspects of the interaction and so are detached from the business goals motivating individual enterprises to participate in the interaction. Second, they specify only electronic messaging and leave out the physical activities that are often crucial to achieving business goals.

These deficiencies call for a richer interaction specification. In particular, capturing business goals and their refinements into activities—electronic and physical—calls for specification at an abstraction level that's suitable for business-level reasoning. We developed a framework for this purpose that uses models of organizational requirements (MORs) to specify interactions.[3] MORs capture the goals that motivate participants to interact in the first place, as well as all the activities that constitute the interaction. We also

0740-7459/10/$26.00 © 2010 IEEE

```
Sequence {
   Claimant Send Claim To Insurer
   Insurer Send ClaimApproval To Claimant
   While (NOT AppointmentConfirmed) Do{
      Claimant Send AppointmentRequest To Repairer
      Choice {
         Repairer Send AppointmentConfirmed To Claimant
         Repairer Send AppointmentRejected To Claimant
      }
   }
   Parallel {
      Repairer Send VehiclePickupDate To Claimant
      Sequence {
         Repairer Send Invoice To Insurer
         Insurer Send Payment To Repairer
      }
   }
}
```

Figure 1. WS-CDL pseudocode for example vehicle-repair interaction. The protocol specifies electronic messaging between the interacting roles from a global point of view and leaves out physical activities.

implemented an automated tool that generates messaging protocols from MORs. We've evaluated our approach in two real-world case studies.

Current Approaches

WS-CDL uses the following constructs to describe messaging protocol and control flow (presented in pseudolanguage for brevity):

- Send…To specifies sending a message of a certain type from a sender role to a recipient role.
- Sequence encloses activities that must execute in order.
- Parallel encloses activities that may execute concurrently.
- Choice represents a conditional choice between mutually exclusive options.
- While (condition) Do represents repetition.

Figure 1 uses these structures to describe a messaging protocol for a vehicle-repair interaction among three roles: Insurer, Claimant, and Repairer. In this protocol, the Claimant submits a repair claim to the Insurer and then obtains an appointment at the Repairer's shop. Eventually, the Repairer notifies the Claimant that the repairs are done (by specifying a vehicle pickup date) and bills the Insurer for the cost.

Languages such as WS-CDL are used to specify messaging between interacting roles from a global observer's viewpoint. In this example, the global observer might be a regulatory agency, such as a state government's insurance commission. The global point of view supports interoperability by abstracting away from the internal business process of each role.

The operational nature of these descriptions detaches the messaging protocol from the business goals it's meant to achieve. This makes it hard to design protocols that satisfy participants' goals without explicitly representing them and relating them to messaging activities. SoaML attempts to represent high-level service architectures. Nevertheless, it lacks mechanisms for refining these architectures into messaging protocols. Architects using SoaML must construct a multiparticipant messaging protocol manually with no systematic means for ensuring that it's consistent with the high-level architecture or that it satisfies the participants' goals. Nor does SoaML help reconcile conflicts among different participants' goals.

Another limitation of emerging standards is their failure to specify physical activities—that is, activities carried out by humans beyond the electronic medium. Physical activities are often crucial to achieving an interaction's goals. For instance, the vehicle-repair interaction is pointless if the Repairer doesn't physically perform the repair, even if all the required protocol messaging takes place as specified. Furthermore, the standards-based descriptions don't specify the order of physical activities relative to the electronic messaging. For instance, we can't specify that the Repairer must finish all repairs before billing the Insurer. Even though BPEL4People tackles human activities, it's limited to specifying a human's interaction with the electronic system and from only one participant's viewpoint.

These deficiencies call for a richer interaction specification. In particular, the need for capturing business goals and their refinements into activities—messaging and otherwise—calls for specifying the interaction at the level of the MORs that motivate the messaging.[3]

Separation of Design Concerns

Whereas the high-level nature of MORs makes them suitable for business-level reasoning, messaging protocols are useful as machine-readable specifications. These two representations not only address different concerns but also serve different purposes for the two stakeholder types—that is, interaction participants and the global observer. To ensure the interaction design properly serves all stakeholders, we separate the concerns of interaction design along two axes (see Figure 2).

The Stakeholder Axis

First, we separate the concerns of interaction participants from those of the global observer.

Interaction participants. Each interaction participant is a stakeholder wishing to fulfill its business needs from a local viewpoint. To ensure that they achieve their goals in joining the interaction, participants must determine which activities performed during the interaction contribute toward fulfilling their goals.

Equally important are the constraints imposed by internal business policies, such as data flow between business activities and preconditions on their execution. Participants' adherence to the interaction protocol must not violate any of their internal policies and vice versa.

Global observer. The global observer is a stakeholder whose concerns are to encourage participation and facilitate participant interaction. These concerns aren't specific to any participant but broadly beneficial to them all. For instance, a global observer's objectives might be to promote trade, enable advancement across an industry sector, or ensure public safety.

To encourage participation, the global observer helps potential participants assess and mitigate the risks involved. The global observer must also ensure fairness by rationalizing the balance between participant obligations and rights. Unfair rules will deter participants from joining the interaction.

To facilitate the interaction, the global observer aims to ensure interoperability. Specifying role obligations up front is essential to this purpose. The specified obligations become a standard contract for participants wishing to play a role in the interaction.

The Abstraction Axis

The second axis separates business concerns, represented in MORs, from concerns related to messaging specification.

MORs. Because MORs represent the interaction at the business-concept level, they can address business concerns in ways that machine-oriented messaging specifications can't. MORs are more suited to processing by humans, such as business analysts and architects acting on behalf of stakeholders. These experts can focus on identifying enterprise goals and reasoning about the means to fulfill them. Specifically,

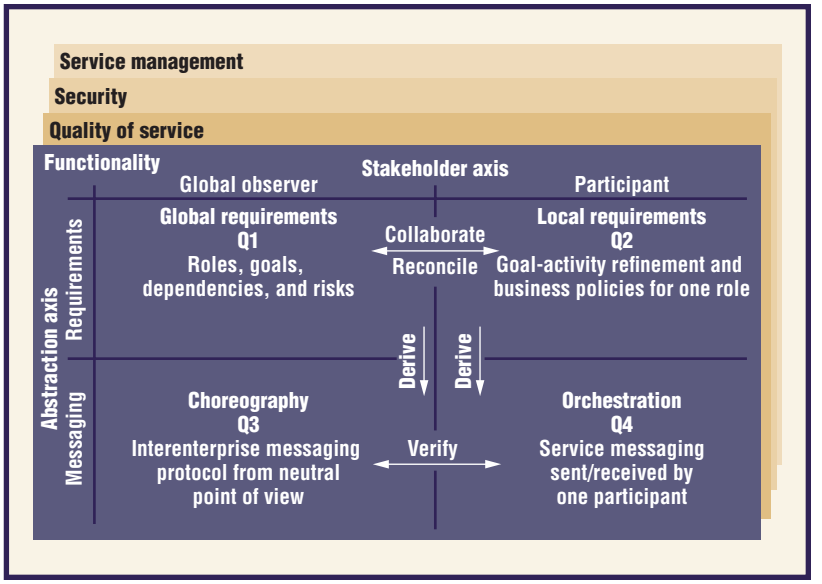

Figure 2. Framework for multiparticipant interaction-protocol design. The framework separates functional design concerns of an interaction into four quadrants (QI–Q4). Nonfunctional concerns orthogonal to protocol specification are represented as parallel planes.

- analysts identify, represent, and decompose business problems in ways that let them deepen their understanding of problems and share business-domain knowledge;[4] and
- architects explore and evaluate alternative solutions for business problems and rationalize the decisions made in choosing solutions. To specify a business solution, architects must identify business activities—electronic or physical—required for implementing the solution and ensure that their execution satisfies the business goals.

All stakeholders share the concern of an interaction's viability. However, when local business needs conflict, MORs capture the interconnections between participants' business processes and thereby provide a means for resolving the conflicts.

Messaging. Messaging protocols address concerns about the correctness of message contents and sequences during the interaction. The protocol specification is the basis for ensuring that participants adhere to their obligations during runtime. Because services and software clients execute these protocols, the specification must be available in a machine-readable language.

Messaging specification also addresses concerns about intraenterprise messaging coordination. An enterprise might participate simultaneously in many different interactions. In addition to fulfilling its obligations in each interaction, an enterprise must coordinate its overall messaging activities to ensure that the business process complies with internal business policies.

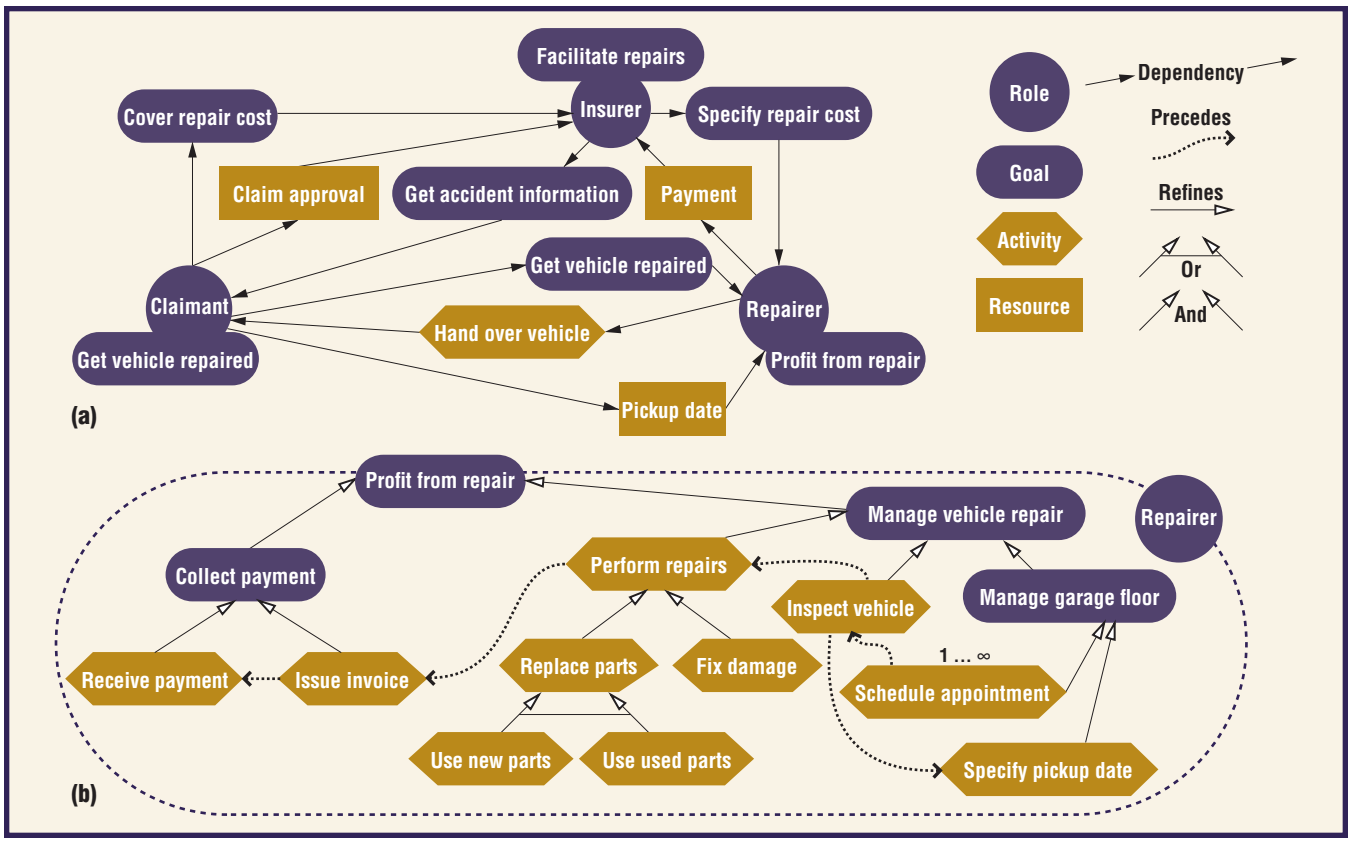

Figure 3. Vehicle-repair requirements diagrams. (a) A role-dependency model of global interaction requirements depicts all three roles and their dependencies, and (b) a goal-activity model for the Repairer role depicts one stakeholder's local requirements.

Four Viewpoints for Service-Interaction Design

Separating interaction design concerns along the axes of stakeholders and abstractions produces the four viewpoints shown in Figure 2. Each viewpoint embodies a subset of a certain stakeholder's concerns.

Q1: Global Requirements

This view embodies the global observer's concerns for specifying the interaction context in terms of interacting roles, high-level motivations for the interaction, dependencies that make the interaction possible, and risks that come with these dependencies.

Role-dependency (RD) diagrams are a suitable tool for this purpose.[5] Figure 3a depicts our proposed use of RD diagrams to represent the vehicle-repair interaction's global requirements. Each role is represented in a circle, and the corresponding goals are attached to it. Dependencies between roles motivate the interaction. Roles depend on each other for fulfilling goals, performing activities, or furnishing resources. For example, the Claimant depends on the Repairer to fix the vehicle.

Dependencies can be fulfilled either physically or via electronic messaging. For instance, a Claimant must physically haul a vehicle to the Repairer to fulfill "Hand over vehicle" dependency. MORs allow flexibility in designing how each dependency is fulfilled. For instance, the interaction can be designed to let the Insurer either mail a check or electronically make the "Payment."

RD diagrams help rationalize responsibilities in fulfilling goals. For example, the Claimant's expectation that the Insurer will "Cover repair cost" is consistent with the Repairer's reliance on the Insurer for "Payment."

RD diagrams also enable reasoning about risks involved in delegating responsibility. For example, it's reasonable to assume that the Repairer has the necessary expertise to fulfill the "Specify repair cost" goal, but it arguably entails the risk of fraud. Identifying such risks drives further analysis to mitigate them or explore alternatives.

Finally, RD diagrams specify what roles and goals aren't included in the interaction. For instance, the role of "Parts supplier" and the goals related to ordering vehicle parts aren't part of the interaction in Figure 3a.

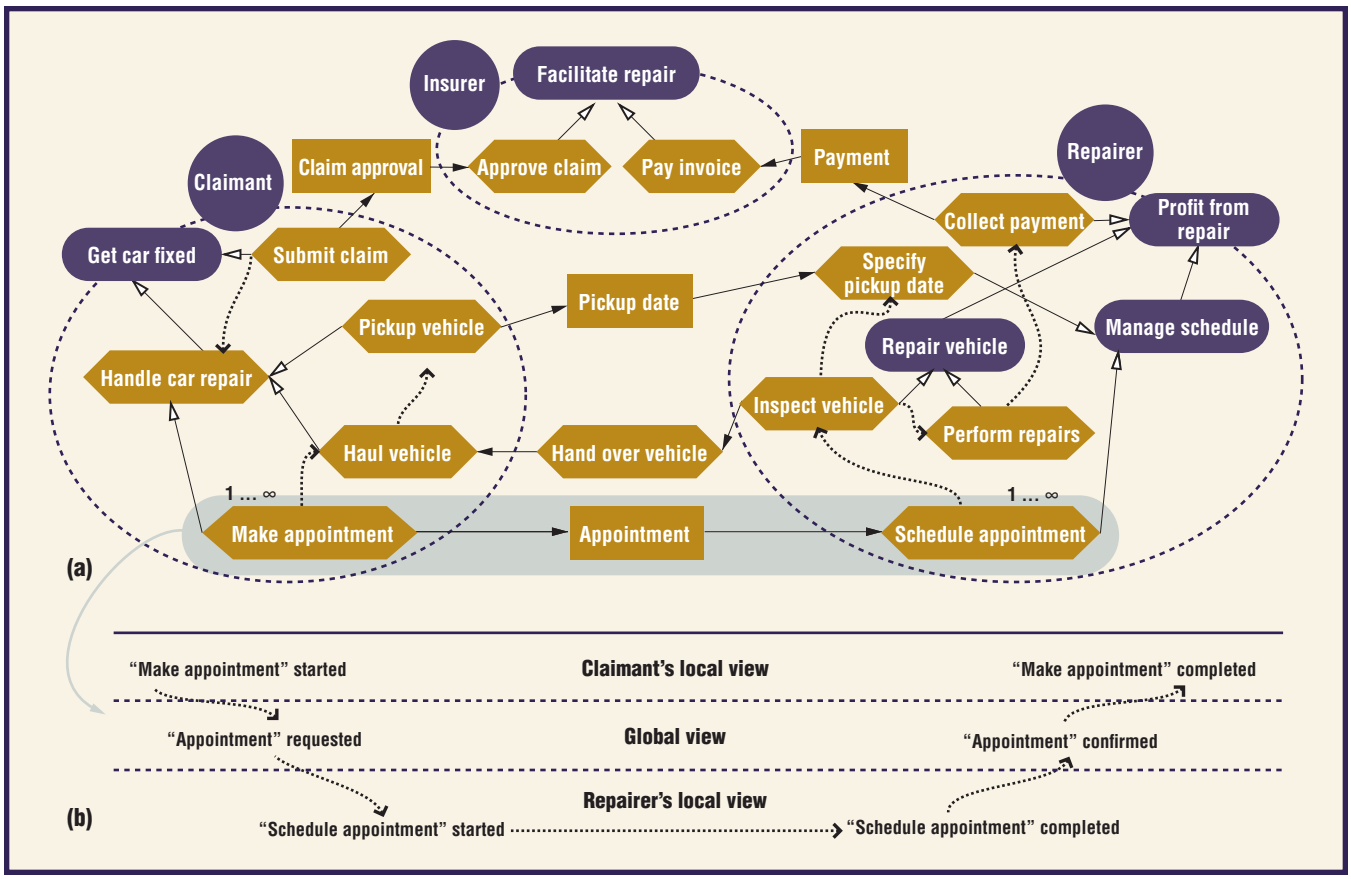

Figure 4. Linking global and local models. (a) The combined local-global model specifies the interaction by binding together the three local goal-activity models (abridged). (b) The order of interenterprise activity execution is inferred from intermodel dependencies.

Q2: Local Requirements

This view embodies one participant's concerns, which are to specify its business goals, determine the activities required to fulfill these goals, and ensure that these activities comply with business policies.

Goal-activity (GA) diagrams are a suitable tool for representing this view.[5] They provide mechanisms for successively refining high-level goals into finer-grained goals and eventually into the activities assigned to one role. Because a GA diagram reflects one role's viewpoint, it can include goals and activities relevant only to that role and not necessarily to the global view. The GA in Figure 3b captures the local view of the Repairer role.

GA diagrams specify the activities, both physical and electronic, a participant carries out to achieve the assigned goals. Refining high-level goals into operational activities establishes relations between them and enables reasoning about how the activities contribute to goal achievement. For example, the Repairer needs to "Specify pickup date" as part of achieving "Manage garage floor," whereas the "Pickup date" in Figure 3a appears to serve a purpose only for the Claimant.

GA diagrams also capture business policies. They represent data-flow and ordering constraints between activities as activity-precedence links. For instance, Figure 3b shows that the Repairer must finish all repairs before issuing an invoice. Using MORs, we can explicitly represent the order of physical activities relative to messaging activities.

Q3: Choreography

This view specifies the messaging protocol from the global stakeholder's viewpoint using languages such as WS-CDL. The protocol describes the valid messaging sequences allowed between interacting roles. It provides a standard against which the global stakeholder assesses participants' compliance.

Q4: Orchestration

This view specifies the messaging and control flow between services implemented by a single participant, either internally or with the outside world. It uses standard process-description languages such as BPEL.

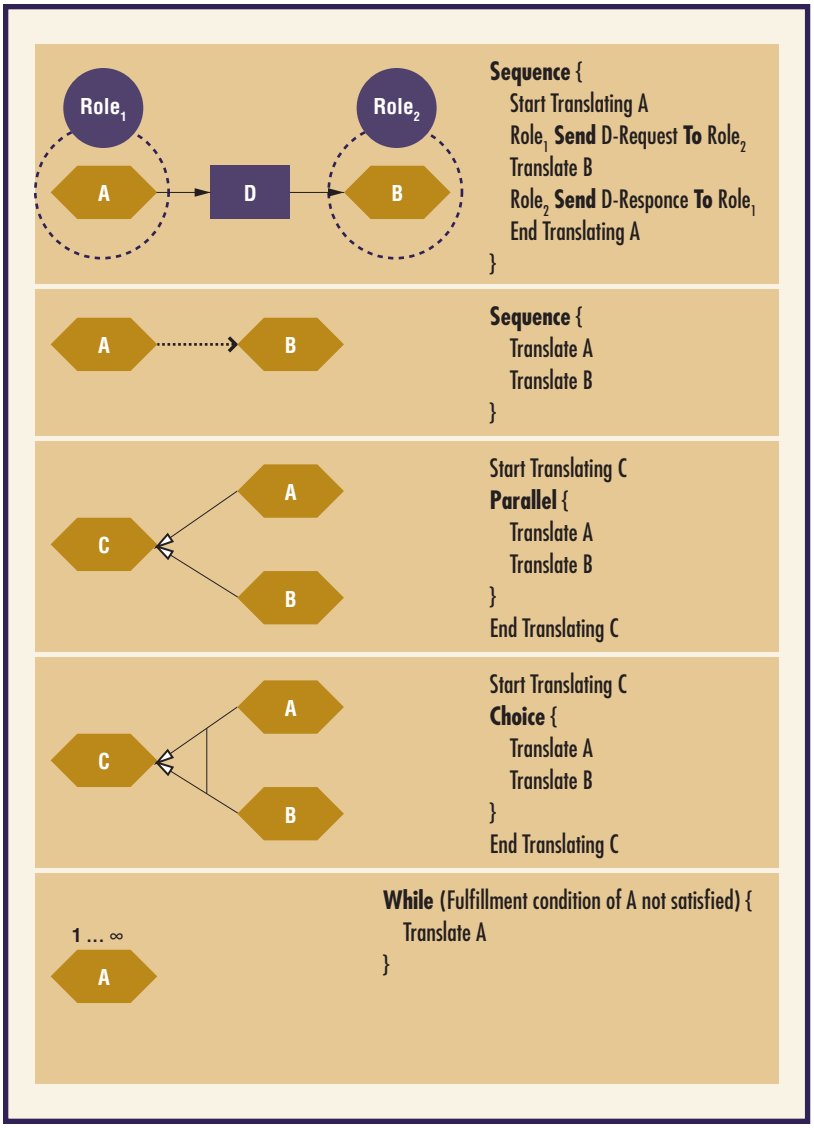

Figure 5. Rules for deriving messaging protocols from requirements model. Each requirements model fragment is translated into messaging-protocol constructs that satisfy the requirements.

Framework for Service Interaction Design

By relating the representations in the four viewpoints, we construct a framework that maintains consistency between them. We focus here on relating Q1 to Q2 and Q1 to Q3.

Consistency from Q1 to Q2

Each dependency in the RD model binds an activity in the depender role GA model to an activity in the dependee role GA model, thereby linking the local models.

Figure 4a shows the result of using dependencies to relate the local GA models of the vehicle-repair interaction roles into a combined local-global model. By combining the GA models for all roles through the intermediating global model, we establish interenterprise ordering of activity execution. The order keeps the depending activity from executing to completion until the dependee activity has fulfilled the dependency. Figure 4b shows how the execution of the Claimant's "Make appointment" activity is tied to that of the Repairer's "Schedule appointment" activity via the "Appointment" dependency. Tying the GA models together supports participant negotiations to reconcile conflicting needs.

Consistency from Q1 to Q3

Dependencies also imply what messages the participants will exchange.[3] A dependency fulfilled electronically typically implies two messages: a depender request and a dependee response providing information that fulfills the dependency. For example, the "Appointment" dependency implies that the Claimant sends the Repairer a message requesting an appointment, and the Repairer replies with the appointment date and time. We determine the set of protocol messages by examining dependencies and the message order by examining execution constraints at both ends of each dependency.

Figure 5 summarizes the basic rules for automatically deriving messaging protocols from requirements models.

Requirements-Driven Interaction Design

By elevating the abstraction level for specifying interactions, our design process focuses on stakeholder requirements. Additionally, by relating local viewpoints to the global viewpoint, it helps participants collaborate with the global stakeholder to reconcile their needs. The design process's forward-engineering version starts with collaborative specification of interaction requirements followed by derivation of the messaging protocol from the combined local-global requirements model. The process provides a path to proceed systematically from possibly conflicting business requirements of multiple enterprises all the way to the specification of interenterprise messaging.

Collaborative Specification of Requirements

Participants collaborate on specifying interaction requirements while the regulatory agency mediates negotiations between them. The design process proceeds iteratively as follows:

- Q2: A participant P1 changes to its local view to comply with business policies or fulfill an emergent goal.
- Q2 to Q1: If the change to P1's local model

involves adding or changing activities that have dependencies, the participant requests propagating the change to the global model.
- Q1: The regulatory agency reviews P1's requested change to the global model and approves it if it finds it reasonable.
- Q1 to Q2: The regulatory agency notifies the participant at the other end of the dependency, P2, of the added or changed responsibility. P2 can then accept the new dependency and propagate its impact to its local model.
- Q2: P2 adapts its local model to fulfill its responsibility toward the added dependency.

The process's iterative nature makes it suitable for application to an existing model.[6] Assuming that the model in Figure 4a is the starting point, the design process can proceed as follows:

- Q2: To guarantee fulfillment of the "Collect payment" goal, the Repairer decides to add an activity, "Verify claim approval," to its local model. This activity must be performed prior to inspecting the vehicle.
- Q2 to Q1: Realizing that this activity requires the Claimant to provide information, the Repairer suggests adding a "Proof of claim approval" dependency to the global model and suggests that the Claimant must fulfill it before car inspection.
- Q1: The state government's insurance commission deems this suggestion reasonable and agrees to it.
- Q1 to Q2: The Claimant is notified of the new dependency and accepts the new responsibility of providing proof of claim approval.
- Q2: The Claimant adds an activity to its local model for providing the approval prior to handing the vehicle to the Repairer.

The process terminates when all requirements have been captured in MORs, at which point we derive the messaging protocol automatically from MOR using the rules of Figure 5.

Deriving Messaging Protocols from Requirements Models

We implemented an automated tool, Chreq (*Cho*reography *Req*uirements) that accepts MORs as input. Using MOR-precise semantics,[7] together with the rules in Figure 5, Chreq generated the messaging protocol for the vehicle-repair example in Figure 1 when fed the MOR in Figure 4a.

Chreq also generates comments, interleaved with the protocol, to indicate points at which physical activities should execute. We also developed transformations from the pseudolanguage used in this article to WS-CDL constructs.

You can download Chreq from https://sourceforge.net/projects/chreq. The download includes source files for the example in this article and those of the case studies.

The vehicle-repair example we've described here is an abridged version of a real-world case study built for a European insurance company. We applied our approach to the full version of the case study, first modeling the original requirements for the European market. Next, we analyzed requirements from real public documents published by Departments of Insurance in several states in the US and Canada. Then we applied our process to the original model to redesign it for the North American context. Finally, we generated the messaging protocol for the redesigned models.

We also applied our approach to a case study from the healthcare domain. In both cases, results were encouraging:[8]

- We easily captured most of the public document requirements by applying our design process iteratively.
- The design process allowed systematic exploration and evaluation of design alternatives based on business policies.
- The design process took into account physical activities naturally as both design constraints and implementation alternatives to electronic messaging.
- Our tool automatically derived messaging protocols from MORs, thereby ensuring consistency between the requirements and the protocol. In fact, the tool helped identify errors in hand-constructed messaging protocols published earlier.

The evaluation helped identify areas to improve our approach:

- Even though MORs are built from a few primitive constructs, there is a learning curve to creating robust models. To smooth this curve, we built a set of patterns that architects can apply to incrementally create requirements models.
- MOR diagrams can get complicated quickly.

Deriving messaging protocols from MORs ensures consistency between the requirements and the protocol.

IEEE Software

About the Authors

Ayman Mahfouz is chief architect at Webalo and a PhD student in requirements-driven adaptation of service-oriented interactions at the Open University, UK. His practical experience is in developing enterprise and mobile software. Mahfouz has an MS in software engineering from the Arab Academy for Science and Technology, Egypt. He's a member of IEEE and the ACM. Contact him at amahfouz@gmail.com.

Leonor Barroca is a senior lecturer in computing at the Open University, UK. Her research interests are in software engineering, particularly in bridging the gap between requirements and design; she is also interested in research skills development in postgraduate education. Barroca has a PhD in computer science from the University of Southampton, UK. Contact her at l.barroca@open.ac.uk.

Robin Laney is a senior lecturer in computing at the Open University, UK. His research interests include software composition, a range of interdisciplinary work in music computing, and the role of software in achieving zero-carbon buildings. Laney has a PhD in computer science from King's College, University of London. Contact him at r.c.laney@open.ac.uk.

Bashar Nuseibeh is a professor of software engineering and chief scientist of Lero, the Irish Software Engineering Research Centre, and a professor of computing at the Open University, UK. His research interests are in requirements engineering and design, security and privacy, and technology transfer. Nuseibeh received his PhD in software engineering from Imperial College London. He's editor in chief of *IEEE Transactions on Software Engineering*. Contact him at b.nuseibeh@open.ac.uk.

We need to develop techniques for modularizing MORs into reusable parts, especially for optional and exceptional execution paths. To help manage the complexity, we plan to integrate our tool with an automatic graph layout tool.

- It remains to be seen how our approach supports reverse engineering—that is, semiautomatic reconstruction of MORs from existing messaging protocols.
- Some problematic aspects of WS-CDL remain challenging at the MOR level—for example, specifying business needs that require synchronizing multiple instances of an interaction.

Overall, these results encouraged us to plan further evaluations, which include getting other practitioners to apply our design process to their business cases.

Acknowledgments

Bashar Nuseibeh is supported, in part, by Science Foundation Irelend grant 03/CE2/I303_1.

References

1. M.P. Papazoglou and D. Georgakopoulos, "Service-Oriented Computing," *Comm. ACM*, vol. 46, no. 10, 2003, pp. 25–28.
2. C. Peltz, "Web Services Orchestration and Choreography," *Computer*, vol. 36, no. 10, 2003, pp. 46–52.
3. A. Mahfouz et al., "Customizing Choreography: Deriving Conversations from Organizational Dependencies," *Proc. 12th Int'l IEEE Enterprise Distributed Object Computing Conf.* (EDOC 08), IEEE CS Press, 2008, pp. 181–190.
4. E. Yu and J. Mylopoulos, "Understanding 'Why' in Software Process Modelling, Analysis, and Design," *Proc. 16th Int'l Conf. Software Eng.* (ICSE 94), IEEE CS Press, 1994, pp. 159–168.
5. P. Bresciani et al., "Tropos: An Agent-Oriented Software Development Methodology," *J. Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, 2004, pp. 203–236.
6. A. Mahfouz et al., "Requirements-Driven Collaborative Choreography Customization," *Proc. Int'l Conf. Service-Oriented Computing* (ICSOC 09), Springer, 2009, pp. 144–158.
7. A. Mahfouz et al., "From Organizational Requirements to Service Choreography," *Proc. 2009 Congress on Services – I*, IEEE CS Press, 2009, pp. 546–553.
8. A. Mahfouz, *Requirements-Driven Design of Service-Oriented Interactions: An Evaluation*, tech. report TR2010-14, The Open University, 2010; http://computing-reports.open.ac.uk/2010/TR2010-14.pdf.