Verification of WS-CDL

Raman Kazhamiakin, Marco Pistore
ITC-Irst, Trento, Italy
{raman,pistore}@itc.it
Outline

• Viewpoints on Web service composition
  o Orchestration
  o Choreography

• WS-CDL overview
  o Structural model
  o Behavioral model
  o Advanced features

• Verification problems and approaches
  o Specification analysis
  o Conformance checking
  o Realizability problem

• Conclusions
Outline

- **Viewpoints on Web service composition**
  - Orchestration
  - Choreography

- **WS-CDL overview**
  - Structural model
  - Behavioral model
  - Advanced features

- **Verification problems and approaches**
  - Specification analysis
  - Conformance checking
  - Realizability problem

- **Conclusions**
Web service composition

- An assembly of **pre-existing** distributed services available on the Web in order to provide a **new** functionality

- Built on top of the Web service platform
  - Exploit stack of protocols that allow for service integration, cooperation, and information exchange independently of the underlying hardware, technology, language and operating system

<table>
<thead>
<tr>
<th>Service aggregation</th>
<th>Choreography</th>
<th>WS-CDL, WSCI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Orchestration</td>
<td>WS-BPEL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service basics</th>
<th>Discovery and publishing</th>
<th>UDDI, WSIL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Functional description</td>
<td>WSDL, abstract BPEL</td>
</tr>
<tr>
<td></td>
<td>Data description</td>
<td>XML, XML Schema</td>
</tr>
<tr>
<td></td>
<td>Messaging</td>
<td>SOAP, WS-Reliability, WS-Addressing</td>
</tr>
</tbody>
</table>
Designing service composition

- Definition of the application...
  - ...**structure**: what the participating services are, how they are related
  - ...**behavior**: how the application is executed, how the services interact with each other

  *complex distributed business process*
Designing service composition

• Definition of the application...
  o **structure**: what the participating services are, how they are related
  o **behavior**: how the application is executed, how the services interact with each other

  *complex distributed business process*

• Use composition specification standards
  o Orchestration & Choreography languages
  o Complementary viewpoints
  o Depend on the design goal
Orchestration

Composition designer

Participating service

Participating service

Participating service

Participating service
Orchestration

- **Process** is always **controlled** from the perspective of **one of the parties**
  - Local enforcement of the application progress
  - Refers to an executable process that interacts with internal/external services
  - Business Process Execution Language (WS-BPEL)
Orchestration

- Control flow of the business process
- Data manipulation and exchange
- Event handling
- Error and compensation handling

Diagram:

- Buyer:
  - decide PO
  - 2 buy or not 2 buy
  - send PO
  - receive offer
  - send Ack
  - send Reject

- Seller:
  - 

Supporting Composition of Distributed Business Processes
Choreography
Choreography

- **Multipart collaboration** description that does not depend on a centralized controller
  - No owner of the application, fully distributed, non-executable application
  - Peers progress independently, interacting and exchanging data in order to accomplish common goal
Choreography: goals

- To describe a **global blueprint** specification of a collaboration to which the **local implementations** of participants should **conform**

- To offer a **means** by which the **rules of participation** can be clearly **defined and agreed**
Choreography: goals

- To describe a **global blueprint** specification of a collaboration to which the **local implementations** of participants should **conform**

- To offer a **means** by which the **rules of participation** can be clearly **defined and agreed**
**Choreography: modeling approaches**

- **Bottom-up: behavioral interfaces**
  - Set of local specifications of the participants
  - Abstract, observable description
  - WS-Choreography Interface (WSCI), Abstract BPEL

![Diagram of buyer and seller interactions](image)
Choreography: modeling approaches

- Top-down: **global workflow**
  - Common observable behavior, global message exchange
  - Declarative reactive rules for progress definition
  - Flexible information-driven model
  - WS Choreography Description Language (WS-CDL)

![Diagram of a workflow process]

- **B**: Buyer
- **S**: Seller
- **W**: Warehouse
- **P**: Payment
- **initiate shipment**
- **cancel**
- **confirm**
- **check available**
- **request PO**
- **provide offer**

19.01.2007 seminar series on distributed systems
Outline

• Viewpoints on Web service composition
  o Orchestration
  o Choreography

• **WS-CDL overview**
  o Structural model
  o Behavioral model
  o Advanced features

• Verification problems and approaches
  o Specification analysis
  o Conformance checking
  o Realizability problem

• Conclusions
WS-CDL

- Pure **Choreography** language
- **Stateful** model of application
- Definition of both **Data Flow** and **Control Flow**
- Simple, expressive and structured notation
- Recursive compositionality – **reuse**
- **Formal** basis – Global Model Formalism
- **Declarative** coordination, information alignment, compensation, exception handling
WS-CDL Basics: structural model

• One participant may implement several roles
• Each role defines the behaviors it can exhibit (WSDL interfaces)
WS-CDL Basics: structural model

- Where, to and how communicate messages
- What channels can be passed
- How to correlate interactions to proper service/conversation instances
WS-CDL Basics: data model

- Data types
  - XML Schema

- Variables
  - local to roles
  - Variable usage: information exchange, state, channel, exception

- Tokens and token locators
  - Shortcuts to important pieces of information

```xml
informationType name="PO" type="PurchaseOrder"

variable name="POVar" informationType="PO"

token name="poID" informationType="xsd:int"

tokenLocator tokenName="poID"
  informationType="PO" part="Order"
  query="/Order/poID"
```
**WS-CDL Basics: control flow model**

- **Interactions**
  - One- or two-way message exchange
  - Specify channel, operation, partners
  - Set of request/respond exchanges
  - Timeout
  - State recording

```xml
interaction
  channelName="sell-channel"
  operation="makeOrder" initiate="true"
  from="Buyer" to="Seller"

exchange action="request" ...
exchange action="response" ...

record role="Seller" action="response"
  source="AckSent"
  target="OrderState"
```

```
request PO
  B S
check available
  S W
provide offer
  S B

cancel
  B S
cancel
  B S
confirm
  B S
payment
  S P
initiate shipment
  S W
```
WS-CDL Basics: control flow model

- Structured activities
  - Sequence

```
request PO
check available
provide offer
confirm
cancel
initiate shipment
payment
initiate shipment
```
WS-CDL Basics: control flow model

- Structured activities
  - Sequence
  - Parallel
WS-CDL Basics: control flow model

- Structured activities
  - Sequence
  - Parallel
  - Choice
WS-CDL Basics: control flow model

- **Workunit**
  - Defines constraints on the execution progress
  - **Guard**: XPath boolean condition on the workunit
  - **Repeat**: XPath boolean condition on the repetition of the workunit
  - **Block**: to wait or not until the guard evaluates to true

**Blocking**
```
when (guard) {
    body
} until (not repetition)
```

**Non-blocking**
```
while (guard) {
    body
} until (not repetition)
```
WS-CDL Basics: control flow model

• Workunit
  o Defines constraints on the execution progress
    • Guard: XPath boolean condition on the workunit
    • Repeat: XPath boolean condition on the repetition of the workunit
    • Block: to wait or not until the guard evaluates to true

Blocking
[Code]

when (guard) {
  body
}until (not repetition)

Non-blocking
[Code]

while (guard) {
  body
}until (not repetition)

• Composing choreographies
  o perform activity to invoke a child choreography
Advanced WS-CDL: events

- Event handing
  - **Blocking workunit**: reaction rules
    - Information events: \( \text{getVariable(} \text{product, quantity, Seller}) = 0 \)\n    - Synchronization events: \( \text{isAligned(order,order,Seller2Buyer)} \)\n    - Time events: \( \text{hasDurationPassed} \) \( \text{hasDeadlinePassed} \)\n    - Choreography state events: \( \text{hasExceptionOccured} \) \( \text{hasChoreographyCompleted} \) \( \text{getChoreographyStatus} \)
Advanced WS-CDL: events

- Event handing
  - Blocking workunit: reaction rules
    - Information events: \( \text{getVariable(product, quantity, Seller)} == 0 \)
    - Synchronization events: \( \text{isAligned(order, order, Seller2Buyer)} \)
    - Time events: \( \text{hasDurationPassed} \), \( \text{hasDeadlinePassed} \)
    - Choreography state events: \( \text{hasExceptionOccured} \), \( \text{hasChoreographyCompleted} \), \( \text{getChoreographyStatus} \)

- Error handling
  - Special \textbf{exceptionBlock} workunits referenced by exception name
Advanced WS-CDL: events

- Event handing
  - Blocking workunit: reaction rules
    - Information events: "getVariable(product, quantity, Seller) == 0"
    - Synchronization events: "isAligned(order,order,Seller2Buyer)"
    - Time events: "hasDurationPassed" "hasDeadlinePassed"
    - Choreography state events: "hasExceptionOccured" "hasChoreographyCompleted" "getChoreographyStatus"

- Error handling
  - Special `exceptionBlock` workunits referenced by exception name

- Compensation handling
  - Modify effect of previously completed choreography: "confirm", "rollback"
  - Special `finilizerBlock` workunits referenced by finilizer name
Advanced WS-CDL: synchronization

- Interaction Based Information Alignment
  - A **requirement** that the partners have **agreement on the outcome** of the interactions, i.e., have a common understanding of variables
  - **Assumes but does not define** a certain low-level protocol that ensures interaction based information alignment
Advanced WS-CDL: synchronization

- Interaction Based Information Alignment
  - A **requirement** that the partners have **agreement on the outcome** of the interactions, i.e., have a common understanding of variables
  - **Assumes but does not define** a certain low-level protocol that ensures interaction based information alignment

- Choreography Coordination
  - A **requirement** to guarantee that all involved roles will **agree** on how the **choreography ended**
  - The choreography is aligned as a whole regardless a particular interaction
  - **Assumes but does not define** a coordination protocol
WS-CDL: pros and cons

- Positively
  - Simple, expressive and formally grounded notation
  - Should become a standard
  - Tool support (Pi4SOA Eclipse plugin)
- Negatively
  - Criticized (arguably) for pre-maturity, too much details, incomplete formalization
  - Still in draft state, not supported by IBM and MS
  - Difficult to map to execution languages: data events, global conditions, etc
Outline

• Viewpoints on Web service composition
  o Orchestration
  o Choreography

• WS-CDL overview
  o Structural model
  o Behavioral model
  o Advanced features

• Verification problems and approaches
  o Specification analysis
  o Conformance checking
  o Realizability problem

• Conclusions
Challenging problems

• Composition specification analysis
  o Verification of the choreography specification

• Realizability problem: from specification to implementation
  o generate end point projections
  o Guarantee correctness of the projection operation

• Conformance checking
  o Given a choreography specification and a set of participant implementations, check that the implementation “correspond” to the specification
    • At design time
    • At run time
Composition verification

Verification of the specification against various behavioral properties

- Consistency checking: principal scenarios are possible
- Deadlock freeness, correct termination, event ordering
- Interaction scenarios, e.g. Message Sequence Charts

- Challenges
  - Simple model checking problem when the data and time are ignored, but undecidable in general
  - Often too abstract to provide reasonable results
Composition verification: approaches

- **General**
  1. Formalize choreography language (transition system, Petri nets)
  2. Formalize properties of interest (Temporal logic, automata)
  3. Apply model checking algorithm to verify a property

- [Foster et al 2005]
  - FSP algebra-like formalism, MSC scenarios, LTSA analysis tool
  - No data, no time: incomplete approach

- [Diaz et al 2005/2006]
  - Focus on timed properties, timed automata, UPPAAL model checker
  - Analysis at low level
  - No data
**Realizability problem**

*Extract local implementation skeletons from choreography such that its properties are preserved*

- Step 1: define projections
  - How to "localize" activities? **WS-CDL activities are always bound to a role**
  - How to project workunit conditions? **Global decision condition can be split**

---

**Supporting Composition of Distributed Business Processes**

---

19.01.2007 seminar series on distributed systems 37
Realizability problem

Extract local implementation skeletons from choreography such that its properties are preserved

- Step 2: check properties of the composition of projections
  - Do they behave as expected?
Realizability problem: approaches

• General
  1. Formal **global** model: choreography representation
  2. Formal **local** model: projection representation
  3. Formal **composition** of local models: common behavior of projections
  4. Formal **criteria of correspondence** between global and composite models

• [Fu et al 2003]
  o Conversation protocols: non-deterministic automaton
  o Local peer automatons + unbounded ordered queues
  o Combination of three conditions implies realizability
    o **Lossless join**: composition of local peer languages produces the same behavior
    o **Queues are optional**: synchronous product does not lose behavior
    o **Autonomy of peers**: in a state can do either only send or only receive, or terminate
Realizability problem: approaches

• General
  1. Formal **global** model: choreography representation
  2. Formal **local** model: projection representation
  3. Formal **composition** of local models: common behavior of projections
  4. Formal **criteria of correspondence** between global and composite models

• [Fu et al 2003]
  o Allow for data in messages [Fu et al 2004]
  o Sufficient but not necessary conditions
  o Does not consider internal transitions – not really a WS-CDL model
  o Does not consider correlation and dynamic process creation
Realizability problem: approaches

• General
  1. Formal **global** model: choreography representation
  2. Formal **local** model: projection representation
  3. Formal **composition** of local models: common behavior of projections
  4. Formal **criteria of correspondence** between global and composite models

• [Carbone et al 2006]
  o Global calculus: process algebra with channel passing and session typing
  o EPP calculus: a “localized” version with send/receive constructs
  o Combination of three principles implies realizability
    o **Connectedness**: causality between activities
    o **Well-threadedness**: session-based principle of locality
    o **Coherence**: preserve local consistency of the service
Realizability problem: approaches

- General
  1. Formal **global** model: choreography representation
  2. Formal **local** model: projection representation
  3. Formal **composition** of local models: common behavior of projections
  4. Formal **criteria of correspondence** between global and composite models

- Connectedness:

---

19.01.2007 seminar series on distributed systems
Realizability problem: approaches

• General
  1. Formal **global** model: choreography representation
  2. Formal **local** model: projection representation
  3. Formal **composition** of local models: common behavior of projections
  4. Formal **criteria of correspondence** between global and composite models

• [Carbone et al 2006]
  o “Official” formal model of the WS-CDL working group
  o Strongest theoretical and algorithmic solution for realizability problem
  o ...but not the same as WS-CDL
Realizability problem: approaches

- Relaxing correspondence requirements
  - Problem: the requirement on **exact** behavior match is too strong
  - Solutions: define a **hierarchy** of correspondence notions
    - Exact match
Request for Quotation

Global order of activities can not be enforced!
Realizability problem: approaches

- Relaxing correspondence requirements
  - Problem: the requirement on *exact* behavior match is too strong
  - Solutions: define a *hierarchy* of correspondence notions
    - Exact match
    - Exact match modulo *reordering with internal* events
Request for Quotation

Global order of interactions can not be enforced!
Realizability problem: approaches

- Relaxing correspondence requirements
  - Problem: the requirement on **exact** behavior match is too strong
  - Solutions: define a **hierarchy** of correspondence notions
    - Exact match
    - Exact match modulo reordering with internal events
    - The **order** in which the participant is supposed to **receive messages** is not affected
Request for Quotation

Local order of reception can not be enforced!
Realizability problem: approaches

- Relaxing correspondence requirements
  - Problem: the requirement on exact behavior match is too strong
  - Solutions: define a hierarchy of correspondence notions
    - Exact match
    - Exact match modulo reordering with internal events
    - The order in which the participant is supposed to receive messages is not affected
    - The order in which the participant is supposed to receive messages from a particular partner is not affected
Realizability problem: approaches

• Relaxing correspondence requirements
  o Problem: the requirement on exact behavior match is too strong
  o Solutions: define a hierarchy of correspondence notions
    • Exact match
    • Exact match modulo reordering with internal events
    • The order in which the participant is supposed to receive messages is not affected
    • The order in which the participant is supposed to receive messages from a particular partner is not affected

• [Kazhamiakin, Pistore 2006]
  o Analysis based on observing interaction schema
    • Sensitive to reordering of events globally, locally, peer-2-peer basis
    • Comparing different queue structures based on algorithm [WWW’06]
    • Require completeness of interaction (no message losses), and channel boundedness
  o Requires additional analysis of the resulting composition
  o Does not consider session types and correlations
Conformance checking

Verify that the behavior of a set of local service implementations corresponds to the choreography model

- General
  1. Formal **global** model: choreography
  2. Formal **local** and **composition** models: implementations
  3. Formal **criteria of correspondence**: conformance relation (equivalence)
Realizability problem

Verify that the behavior of a set of local service implementations corresponds to the choreography model

• General
  1. Formal **global** model: choreography
  2. Formal **local** and **composition** models: implementations
  3. Formal **criteria of correspondence**: conformance relation (equivalence)

• Challenges
  o **Design-time** and **run-time** conformance checking
  o **Data/time** issues in conformance checking
  o **Asynchronous** communications with queues in implementations vs. atomic interactions in choreography
  o Information **alignment** and **coordination** requirements
    • Does the implementation ensure the coordination/alignment?
Problems: information alignment

Failure:
Data alignment (b.offer != s.offer)
Realizability problem

Verify that the behavior of a set of local service implementations corresponds to the choreography model

- Approaches
  - automata-based [Baldoni et al. 2005]
  - Process-algebra based [Busi et al. 2005]
  - Usually consider synchronized interaction, omit data/time issues
- [Kazhamiakin, Pistore 2006]
  - State transition systems +
    - asynchronous communications, unbounded queues, data management
  - Abstraction-based symbolic equivalence checking
  - Alignment and coordination checking (as CTL model checking)
    - Interaction alignment rule: no losses, no intermediate emissions, data values
    - Coordination rule: can terminate in all and only coordinated states
  - Requires much heavier machinery
Ongoing and future work

Modeling
- Work on relations with various languages (e.g., BPEL)
- Design tools
- Execution semantics for choreography enforcement

Design-time analysis
- Bring the stuff together
- Implementations

Run-time conformance and monitoring
Improved code generation techniques
Questions