Asynchronous Conversation Patterns

Ian Cooper

Who are you?

I am a polyglot coding architect with over 20 years of experience delivering solutions in government, healthcare, and finance and ecommerce. During that time I have worked for the DTI, Reuters, Sungard, Misys, Beazley, Huddle and Just Eat Takeaway delivering everything from bespoke enterprise solutions, 'shrink-wrapped' products for thousands of customers, to SaaS applications for hundreds of thousands of customers.

I am an experienced systems architect with a strong knowledge of OO, TDD/BDD, DDD, EDA, CQRS/ES, REST, Messaging, Design Patterns, Architectural Styles, ATAM, and Agile Engineering Practices

I am frequent contributor to OSS, and I am the owner of: https://github.com/BrighterCommand. I speak regularly at user groups and conferences around the world on architecture and software craftsmanship. I run public workshops teaching messaging, event-driven and reactive architectures.

I have a strong background in C#. I spent years in the C++ trenches. I dabble in Go, Java, JavaScript and Python.

www.linkedin.com/in/ian-cooper-2b059b

Agenda

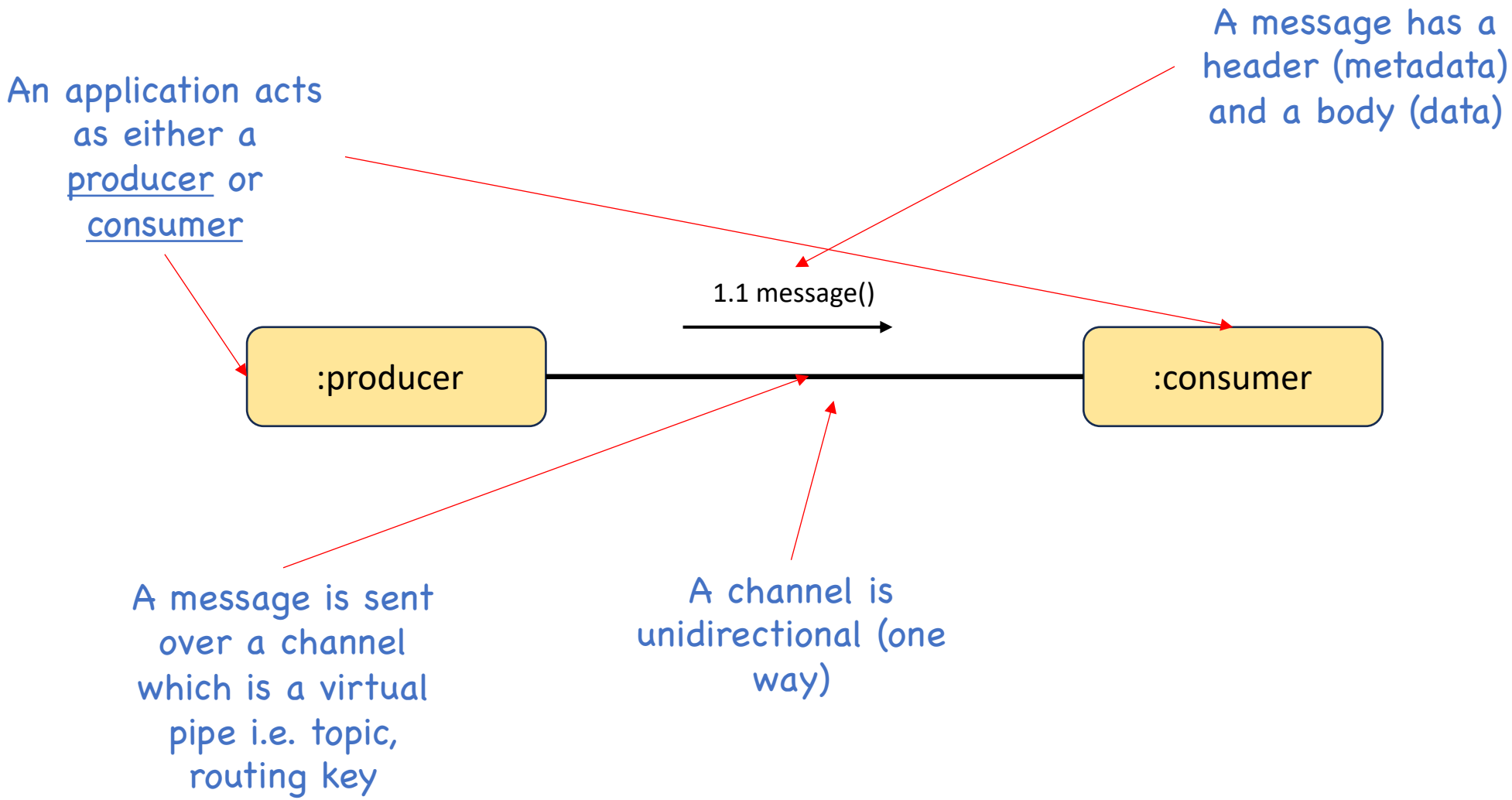
- Messaging & Eventing
- Activity
- Repair and Clarification
- Conversations

Messaging & Eventing

Going beyond integration patterns

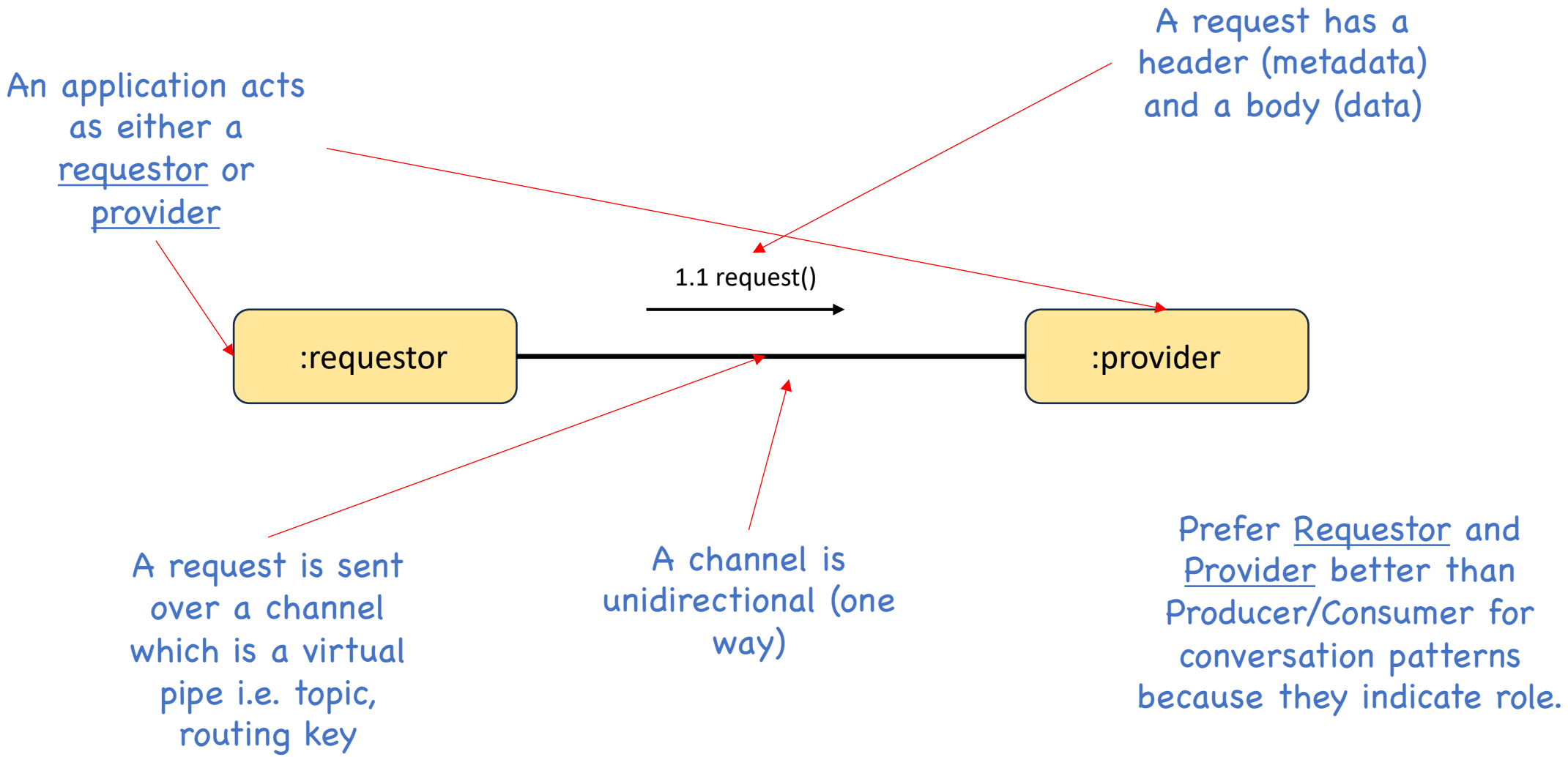
Messaging Participants

Messaging	Activity	Repair and Clarification	Conversations



Messaging Participants

Messaging	Activity	Repair and Clarification	Conversations

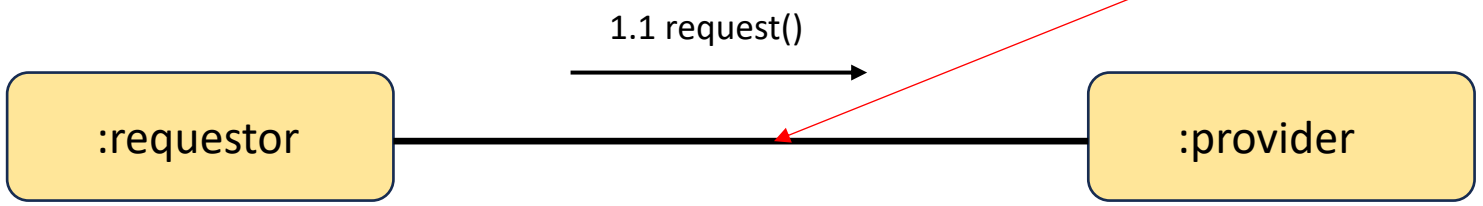


In-Only (Fire and Forget) – Consumer As Provider

Messaging	Activity	Repair and Clarification	Conversations

Typically we call this **fire-and-forget**.

Under a In-Only pattern the requestor sends a request to the provider, but does not seek an acknowledgment of completion of the requested operation

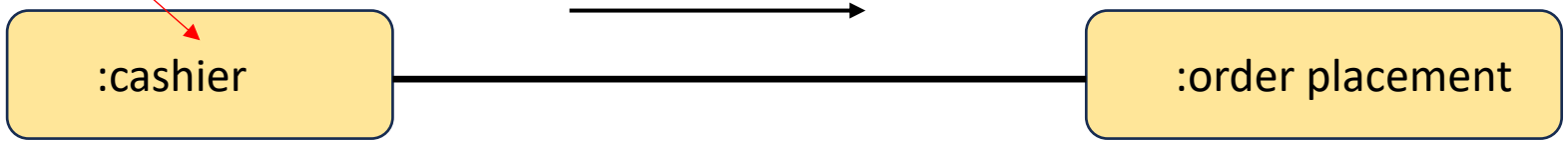


In-Only (Fire and Forget)

Messaging	Activity	Repair and Clarification	Conversations

Requestor: Has a paid for basket it wants to turn into an order

Typically fire and forget is used where we are finished with our part in a flow, and transferring control. We need no response as we are done..



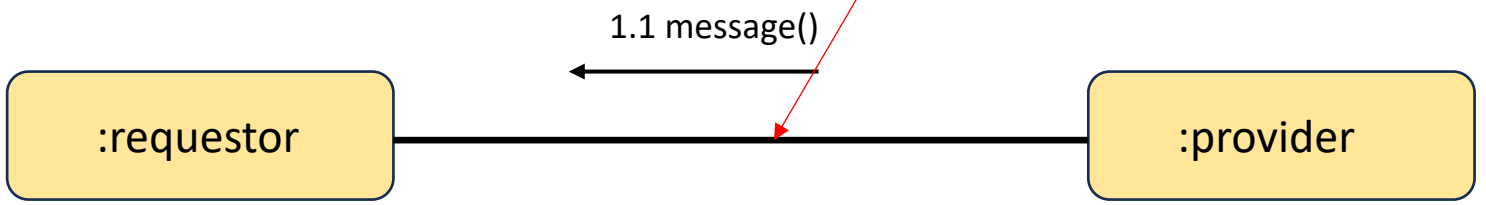
Provider:
Raises an Order

Out-Only (Notification) - Producer As Provider

Messaging	Activity	Repair and Clarification	Conversations

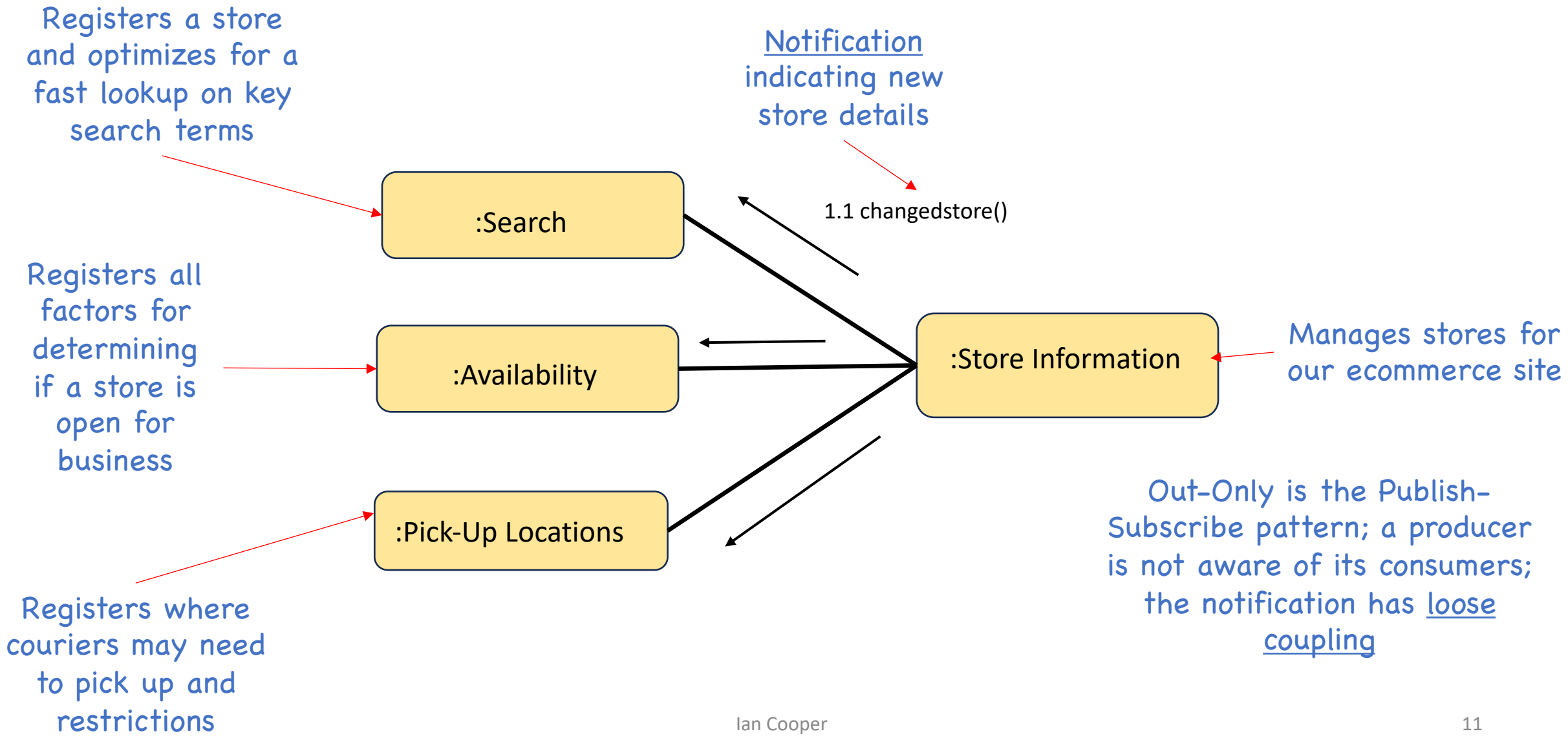
Typically we call this a **notification**.

Under an Out-Only pattern the requestor subscribes to the provider, and an operation is triggered by receipt of a message but it does not acknowledge the message to the provider.



Out-Only (Publish-Subscribe)

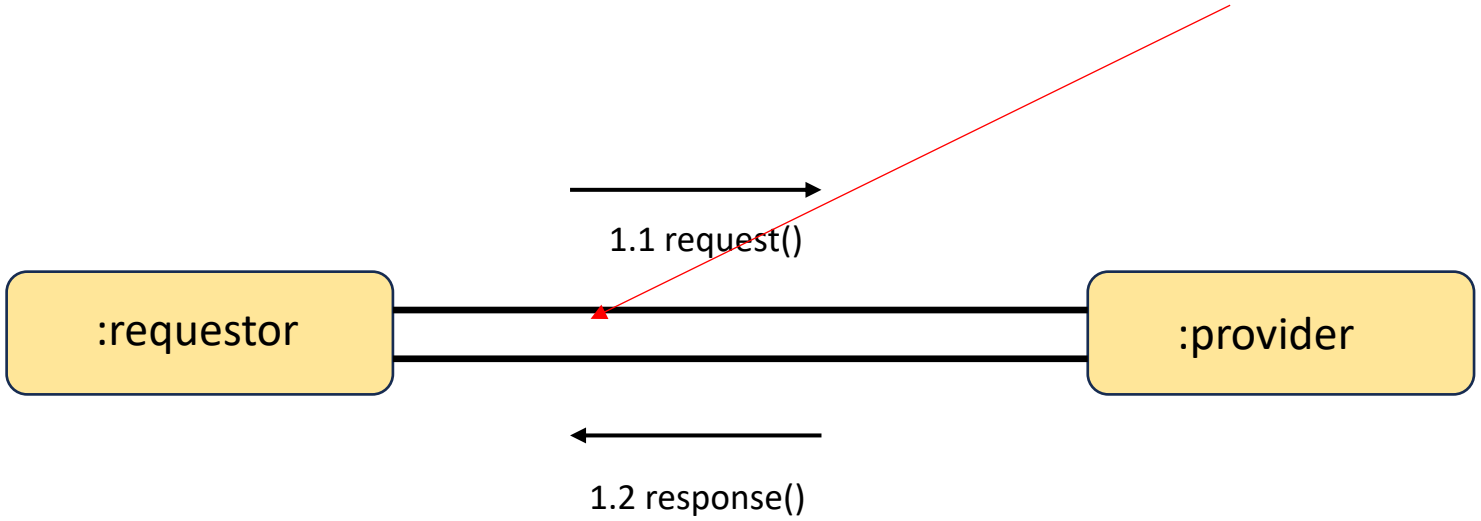
Messaging	Activity	Repair and Clarification	Conversations



In-Out (Request-Response) – Both Consumer and Producer

Messaging	Activity	Repair and Clarification	Conversations

Under an In-Out pattern the provider receives a request from the requestor, and returns a response from the operation triggered by that message.



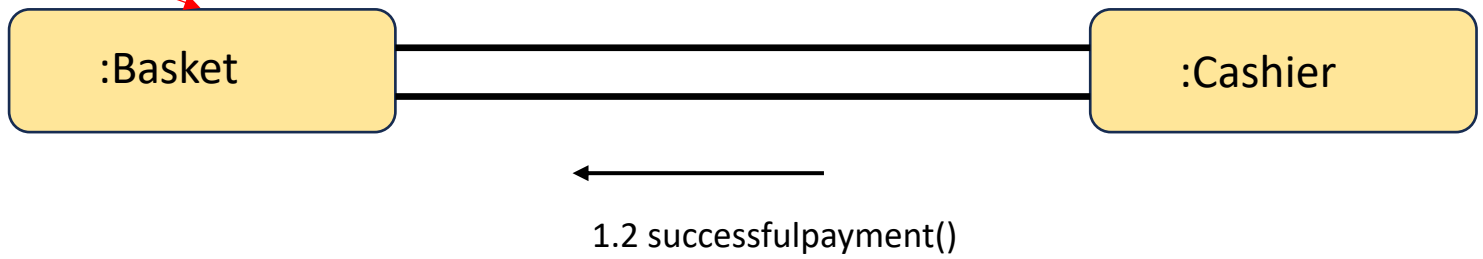
Request-Response (Command-Acknowledgement)

Messaging	Activity	Repair and Clarification	Conversations

Manages items for purchase by a customer of our ecommerce site

Command: asking checkout for the operation to price a basket

Operation: Takes payment for the agreed amount and turns a basket into an order



In-Out can support the Request-Response or Command-Acknowledgement pattern; a requestor triggers an operation on the provider and gets a response

Response: indicating the basket has been paid for

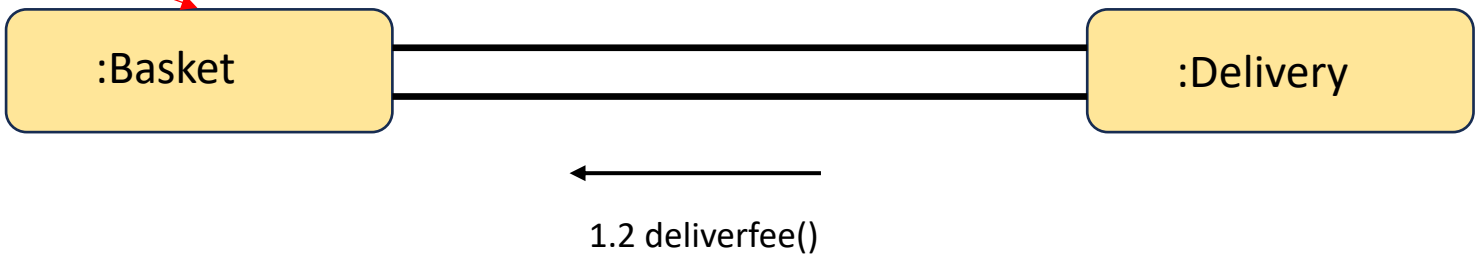
In-Out (Query-Result)

Messaging	Activity	Repair and Clarification	Conversations

Manages items for purchase by a customer of our ecommerce site

Query asking for delivery fees for this basket

Operation: Determines the delivery fee based on store and items in the basket

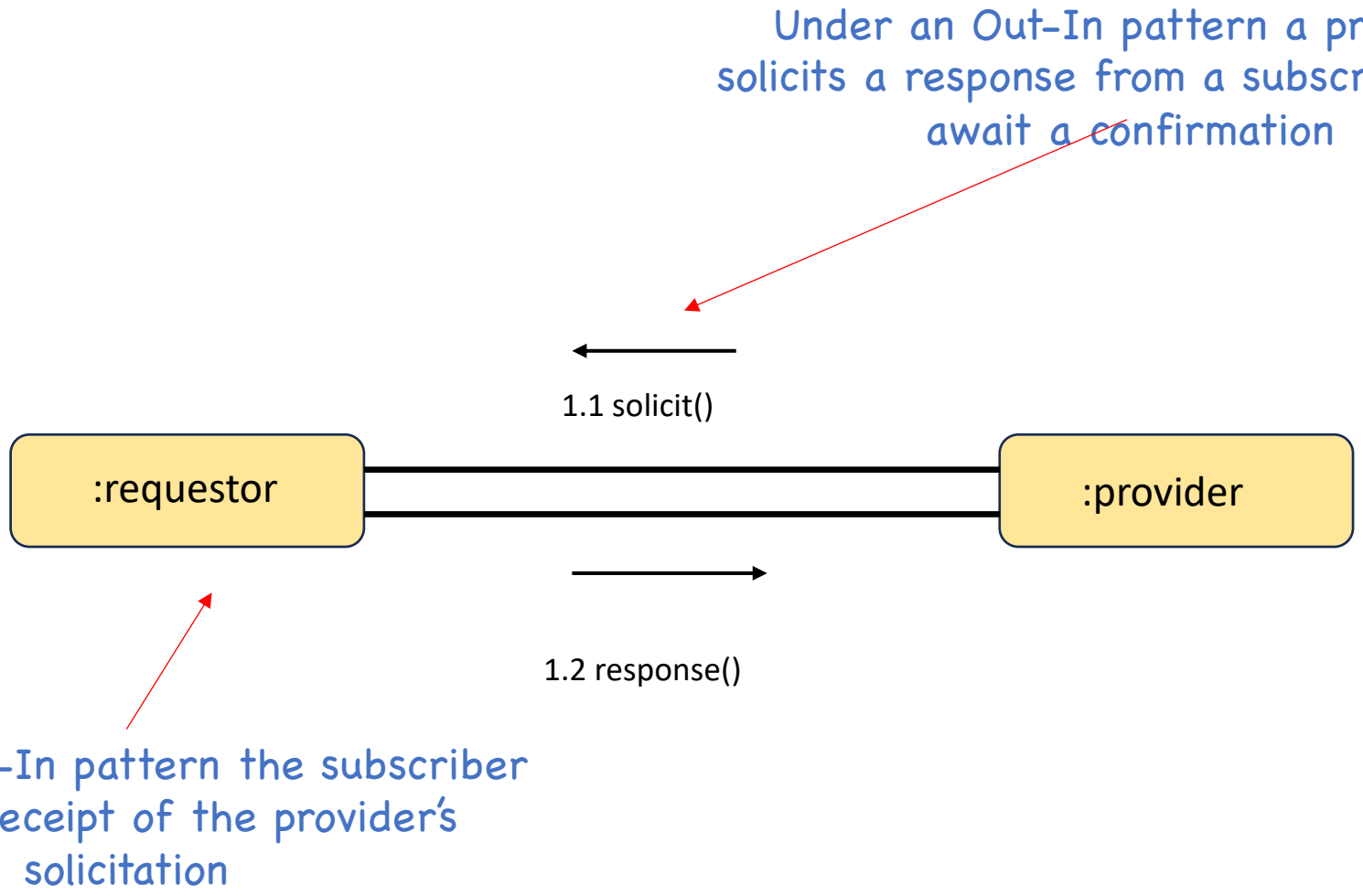


In-Out can support the Query-Result pattern; a requestor triggers a query on the provider and gets a result

Result indicating the fee for the current state of the basket

Out-In (Solicit-Response) - Both Consumer and Producer

Messaging	Activity	Repair and Clarification	Conversations

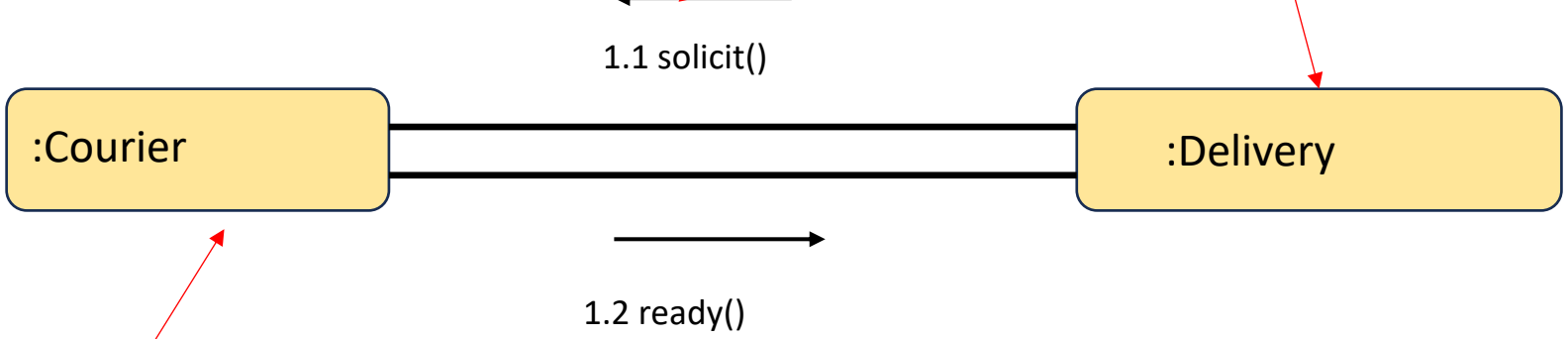


Out-In (Solicit-Response)

Messaging	Activity	Repair and Clarification	Conversations

Query to see if a courier is available to receive orders for delivery; usually Delivery service will follow with notifications of available work.

Delivery service assigns delivery request to drivers



Courier offers jobs depending on location and busyness

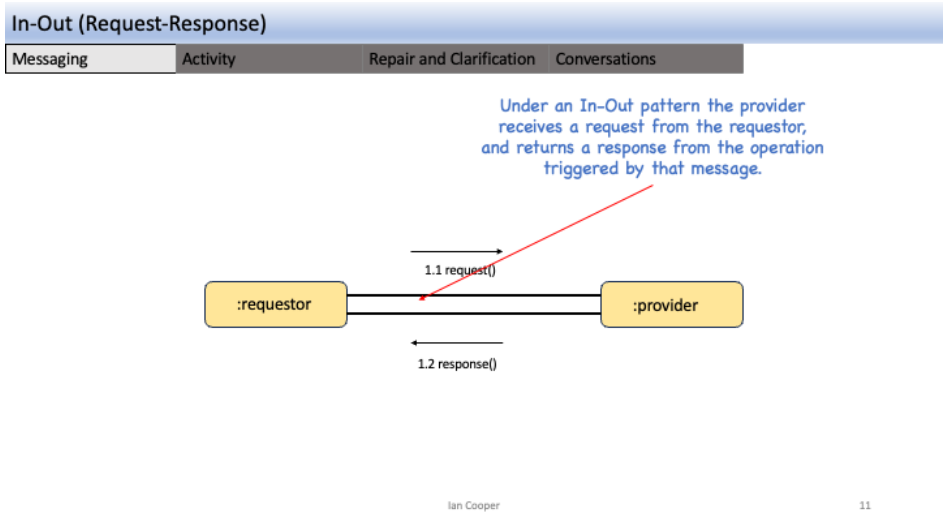
Messaging

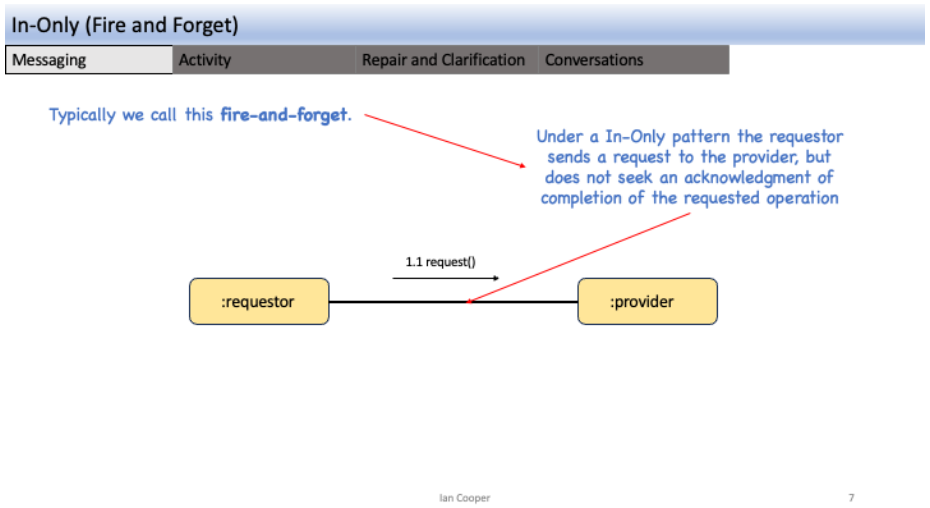
Has Intent

Request An Answer (Query)
Transfer of Control (Command)
Transfer of Value

Part of a Workflow
Part of a Conversation

Concerned with the Future





Eventing

Messaging	Activity	Repair and Clarification	Conversations

Eventing

Provides Facts

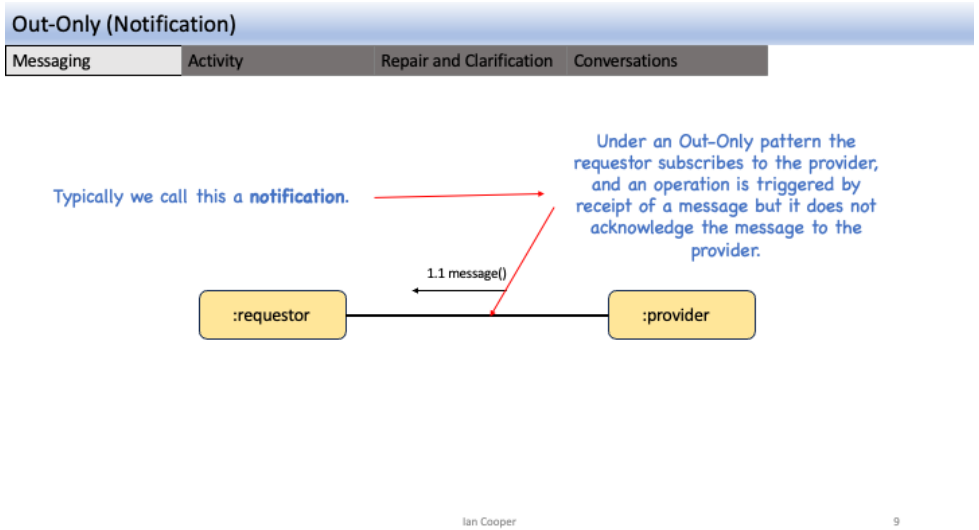
Things you Report On

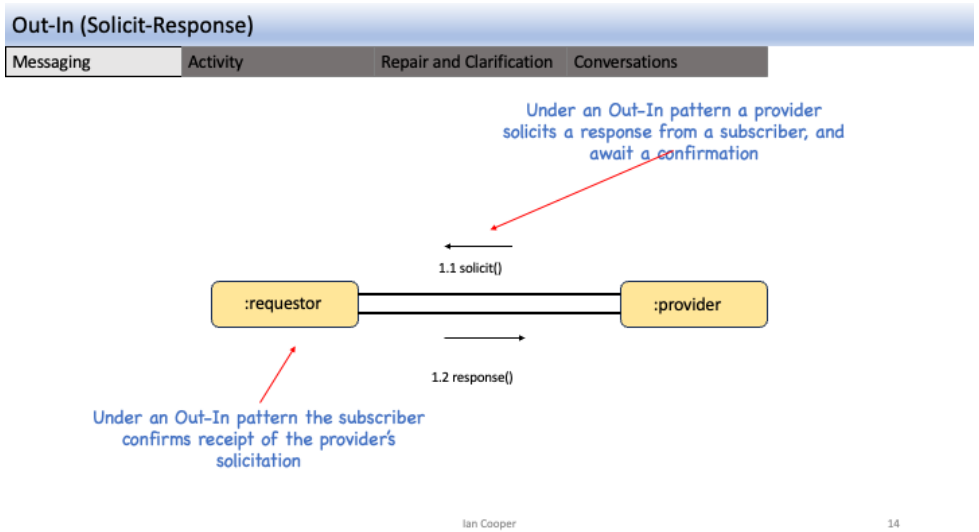
No Expectations

History

Context

Concerned with the Past



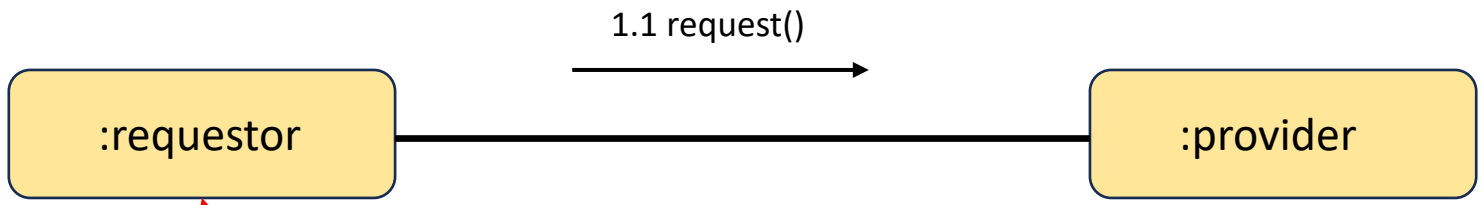


Activity

How do we manage workflows in requestors and providers

In-Only (Fire and Forget)

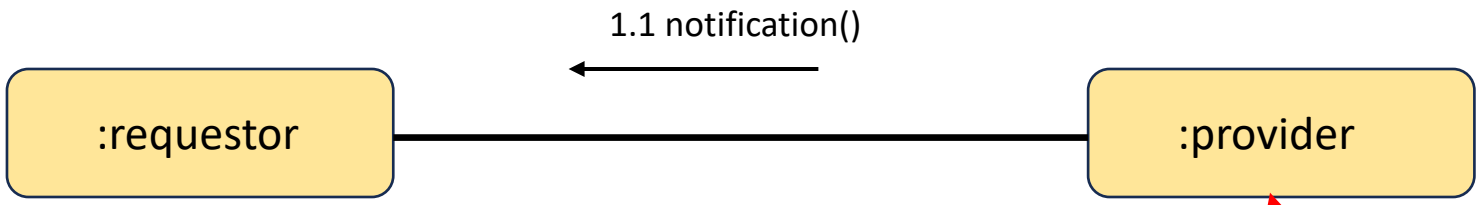
Messaging	Activity	Repair and Clarification	Conversations



Because we are **fire-and-forget** there is no need to maintain state to correlate with a response; we don't expect a response

Out-Only (Notification)

Messaging	Activity	Repair and Clarification	Conversations



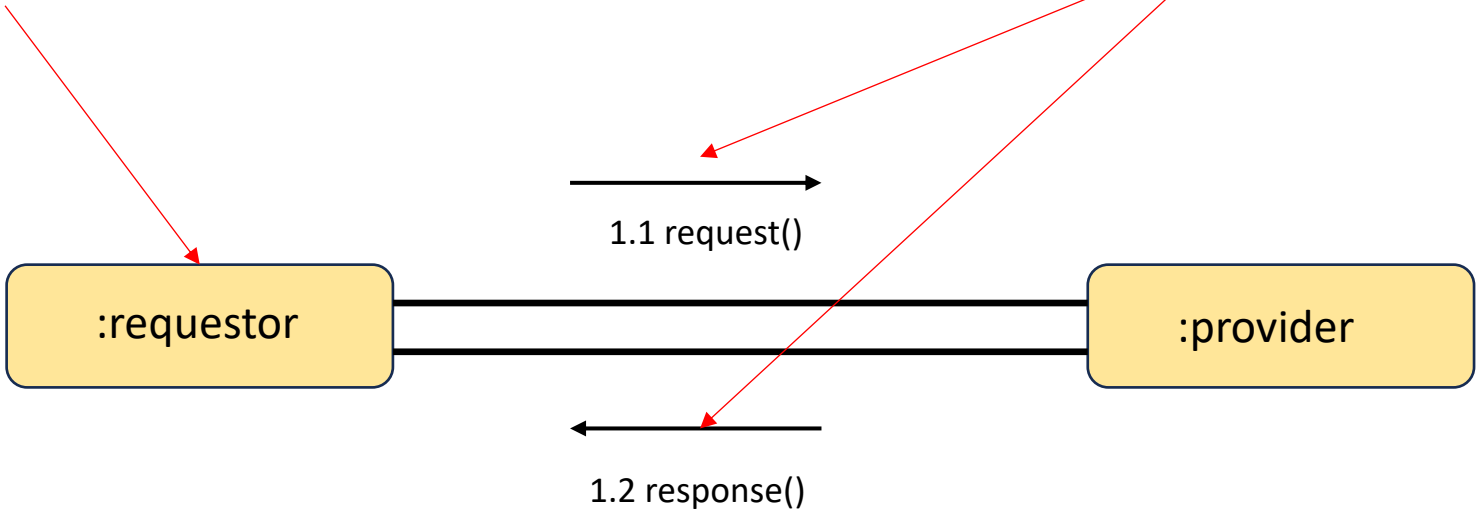
Because we only **notify** there is no need to maintain state to correlate with a response; we don't expect an acknowledgement

In-Out (Request-Response)

Messaging	Activity	Repair and Clarification	Conversations

We may resume a workflow that we suspended, whilst we were waiting for a response

Because we expect a response to our request we may need to save state which we then correlate with the request

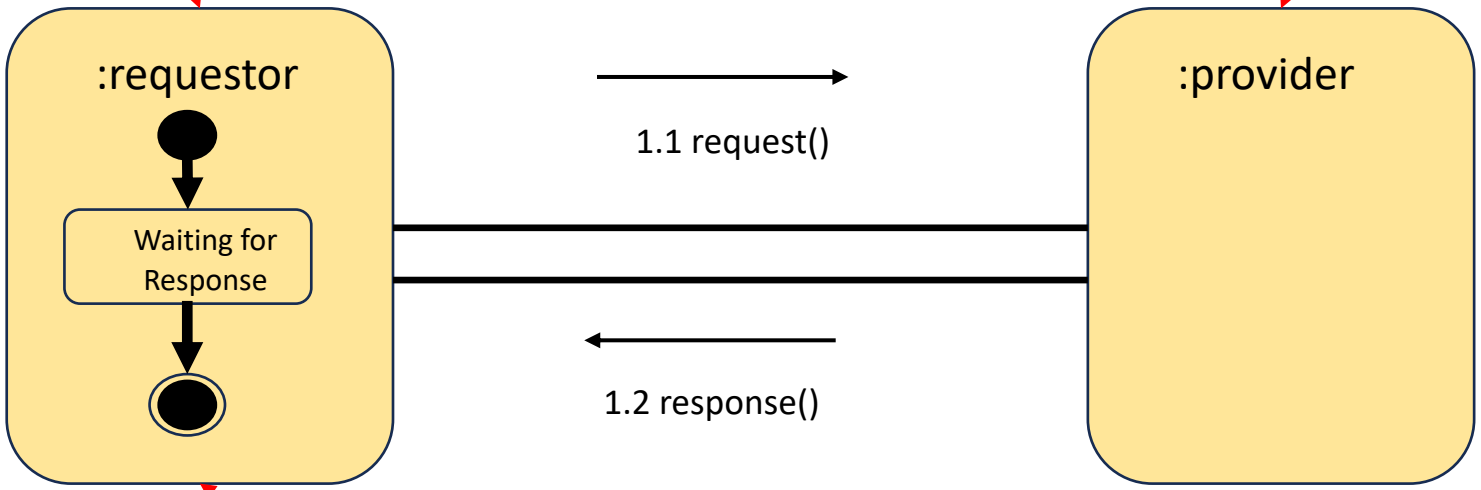


In-Out (Request-Response)

Messaging	Activity	Repair and Clarification	Conversations

Because the flow is asynchronous, the requestor enters a waiting for response state – as it cannot complete its own operation until it gets the response

The provider can remain stateless as it only needs to return a response to the requestor



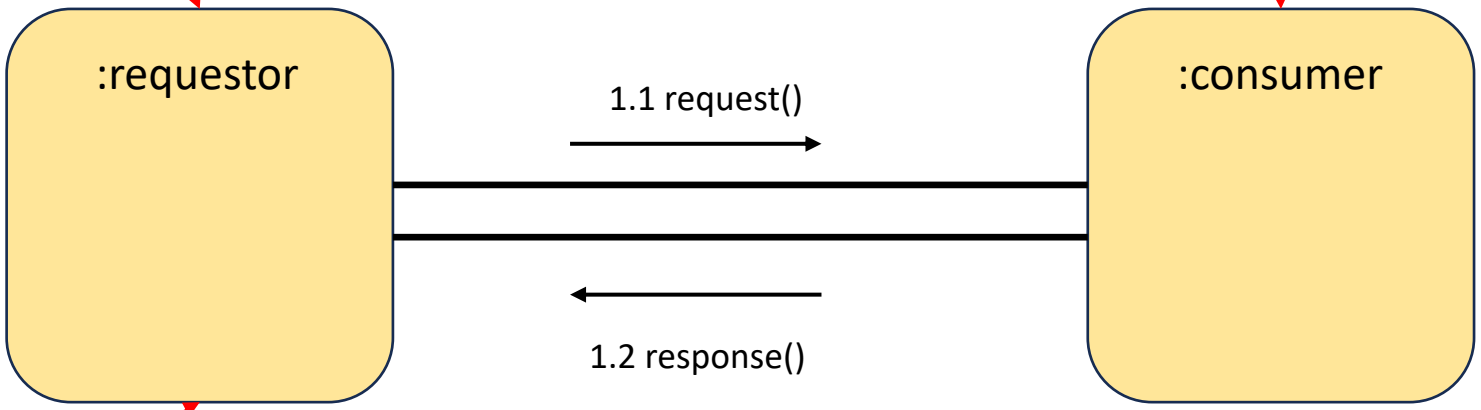
The requestor must be STATEFUL

In-Out (Request-Response)

Messaging	Activity	Repair and Clarification	Conversations

The requestor can remain stateless by including the state of the conversation within the message it sends

The provider returns the state of the conversation, adjusted for its operation results



The requestor does not need to be STATEFUL, as it just obtains the state of the conversation from the message

N.B. in HTTP this idea becomes HATEOAS

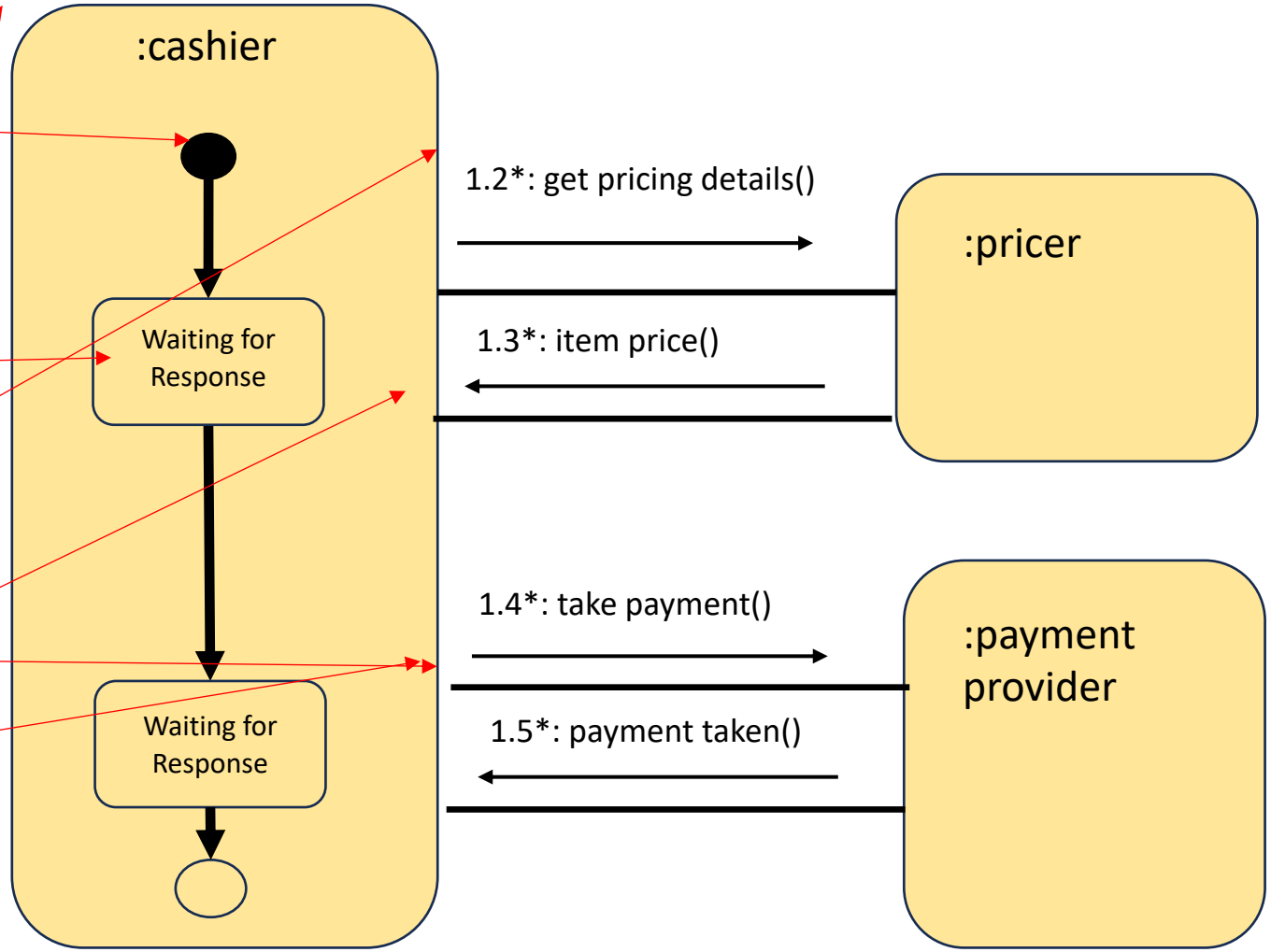
Activity

Messaging	Activity	Repair and Clarification	Conversations

An activity is the code that runs in response to receiving a message.

An activity also stores the state associated with the activity – what did we do in our last turn?

An activity also knows the messages that is has sent and received, so it can be reconstructed.



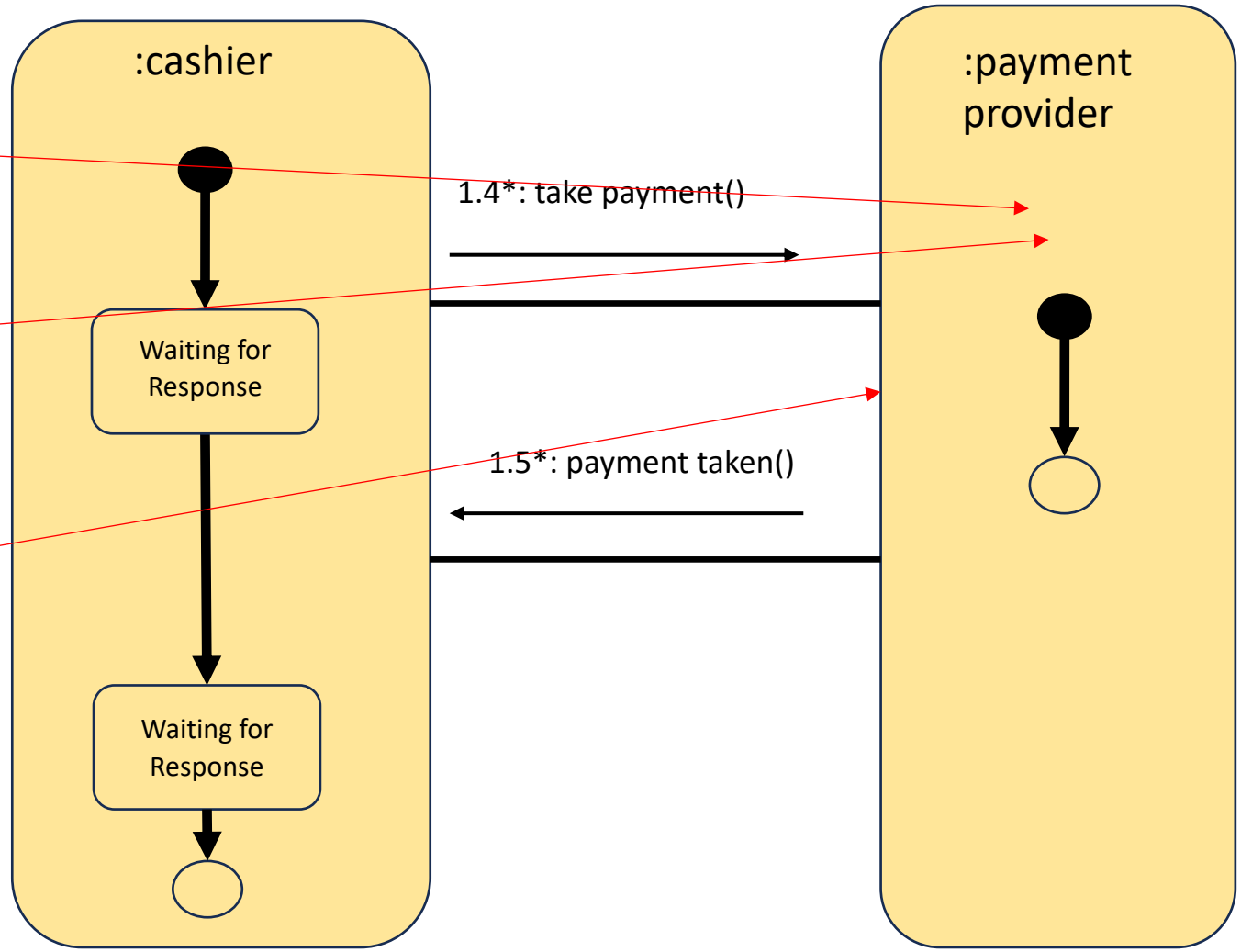
Activity

Messaging	Activity	Repair and Clarification	Conversations

An activity is the code that runs in response to sending a message.

An activity *may* store state associated with the activity – but also may be stateless.

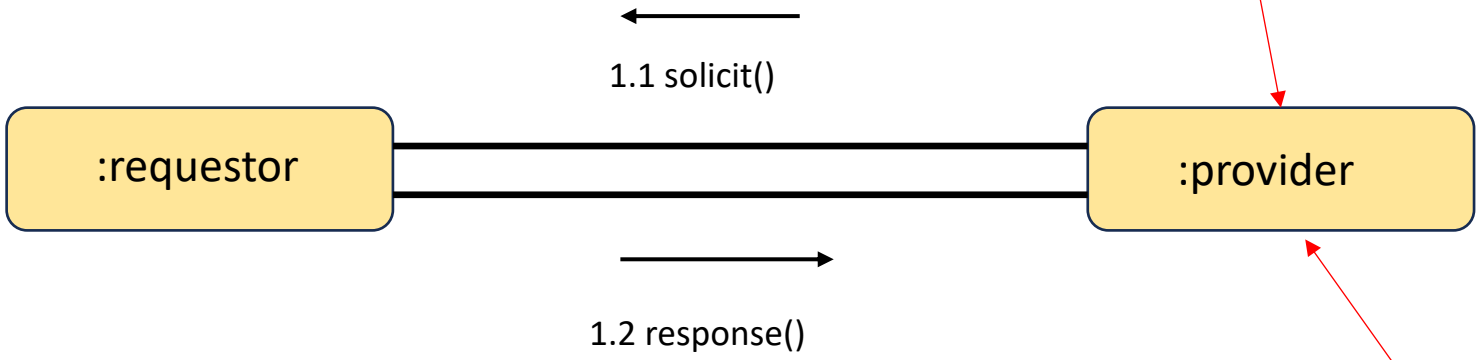
Each activity also knows the messages that is has sent and received, so the conversation can be reconstructed.



Other Activity State

Messaging	Activity	Repair and Clarification	Conversations

A provider that has subscribers may solicit responses from them; this requires keeping track of those subscribers to correlate the responses.



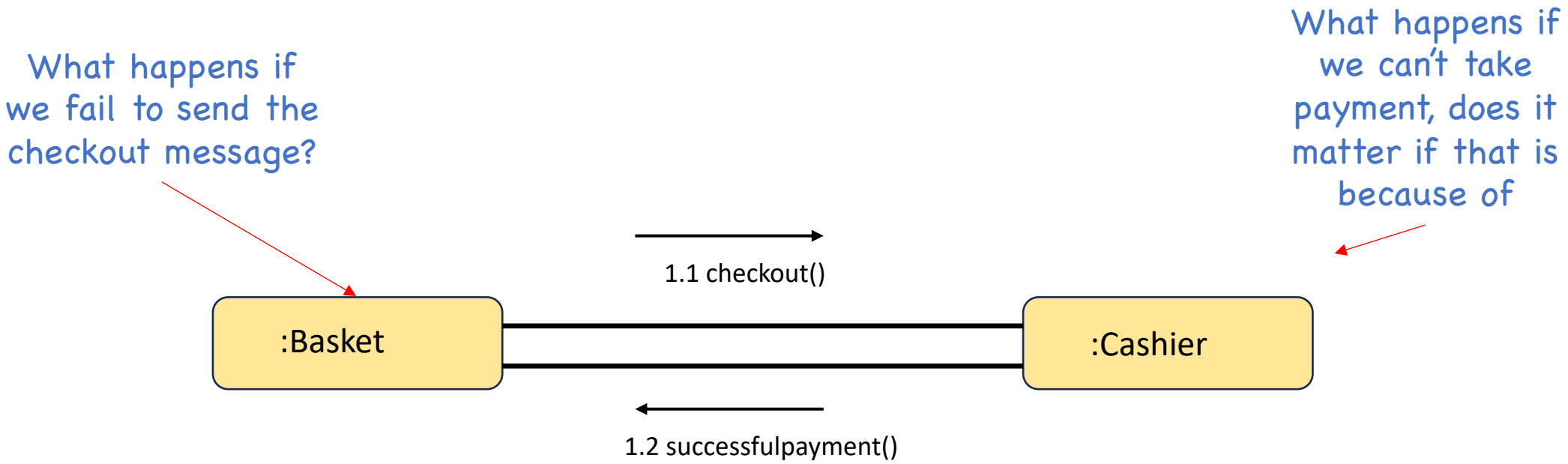
Under an Out-In pattern a provider solicits a response from a subscriber, and await a confirmation

Repair and Clarification

What happens when it all goes wrong

Failure

Messaging	Activity	Repair and Clarification	Conversations

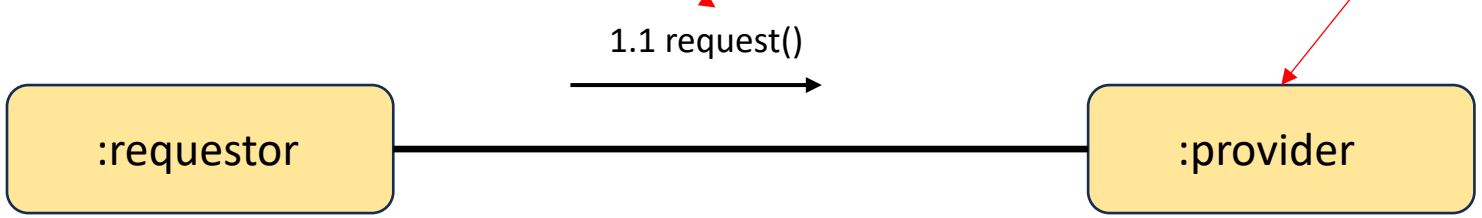


Repair and Clarification

Messaging	Activity	Repair and Clarification	Conversations

The requestor triggers an operation on the provider by sending it a message

We need to consider the possibility of a fault being triggered by the operation triggered by the message.

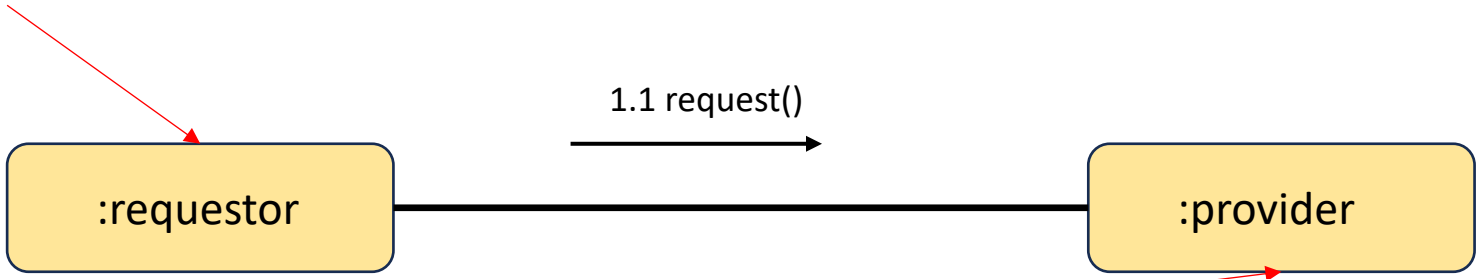


What guarantees does the provider make to the requestor about communicating any faults that are triggered?

No Fault

Messaging	Activity	Repair and Clarification	Conversations

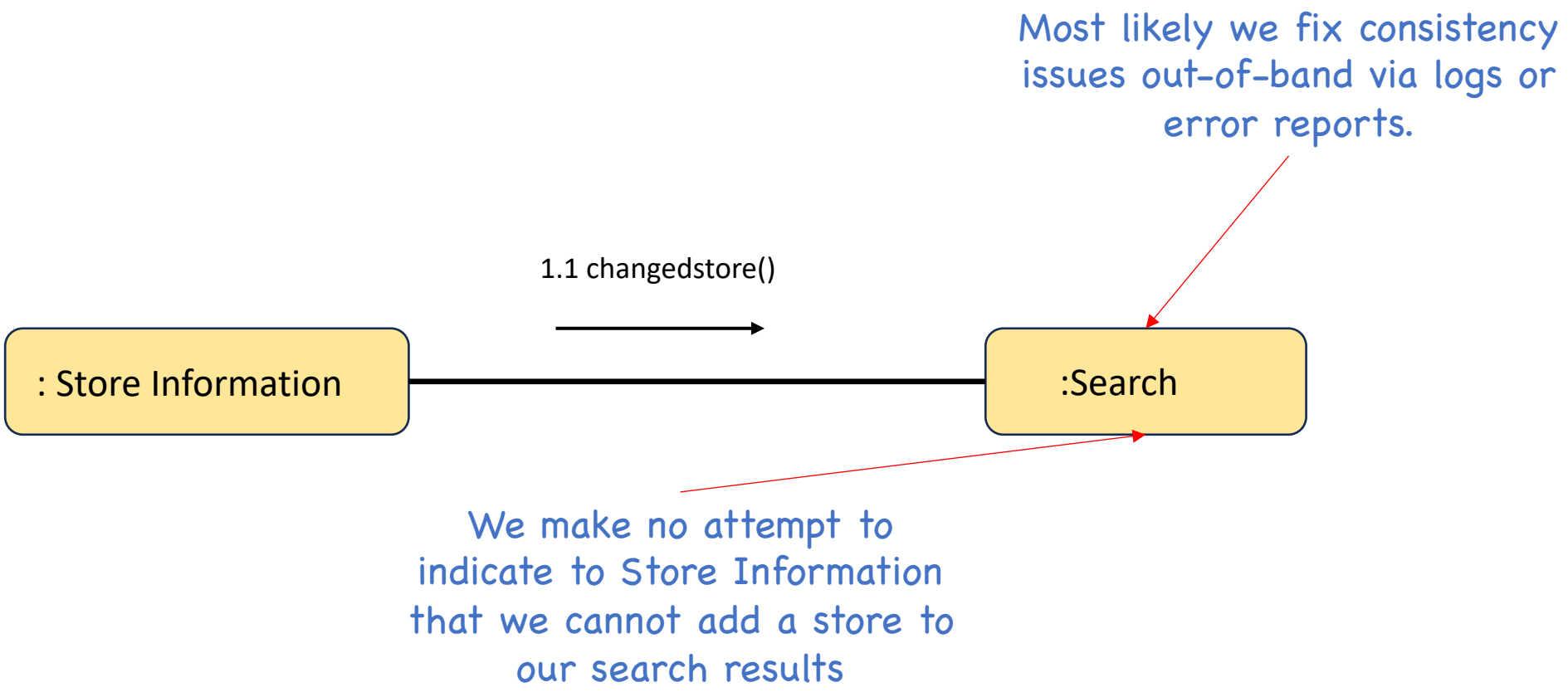
From the requestor's perspective, faults are an application issue for the provider, and not its concern



Under a No Fault pattern the provider makes no attempt to communicate triggered faults from the operation to the requestor

No Fault

Messaging	Activity	Repair and Clarification	Conversations

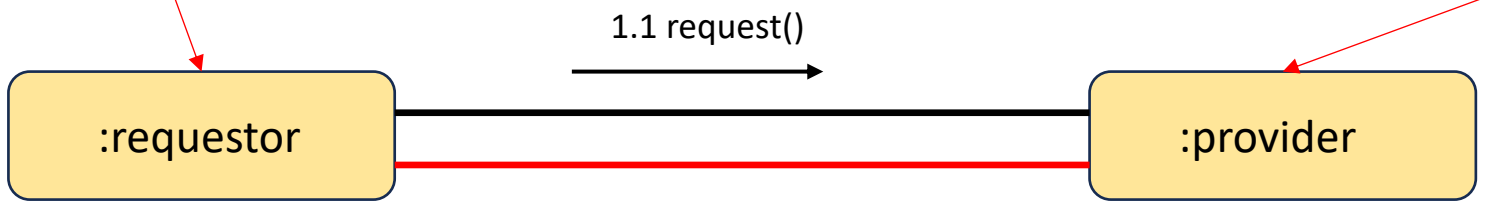


Message Triggers Fault

Messaging	Activity	Repair and Clarification	Conversations

Assumption here is that the requestor can take action on receipt of the fault; but does not normally take a response.

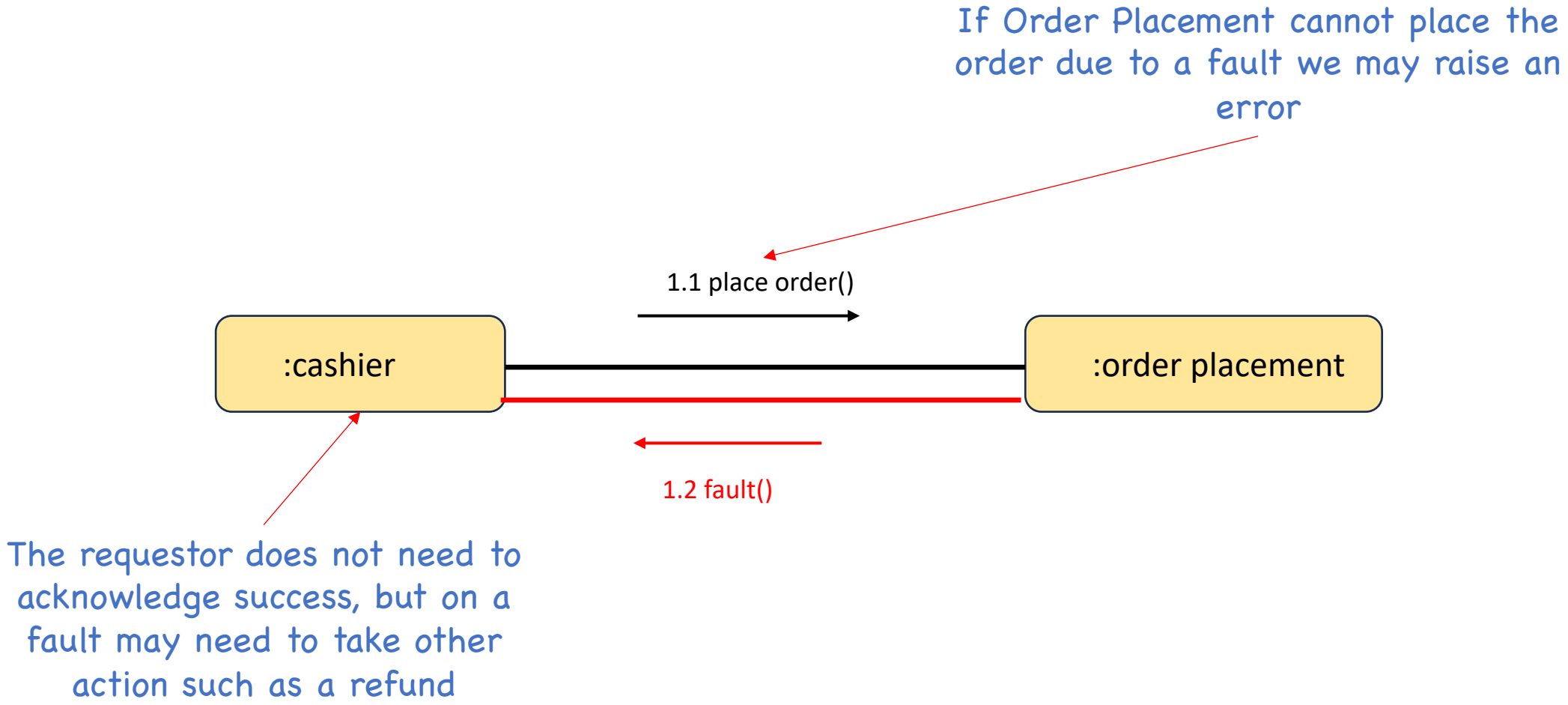
Under Message Triggers Fault pattern a **provider** propagates triggered faults from the operation back to the **requestor** that triggered the operation via a message



The message must have the opposite direction and go back to the requestor

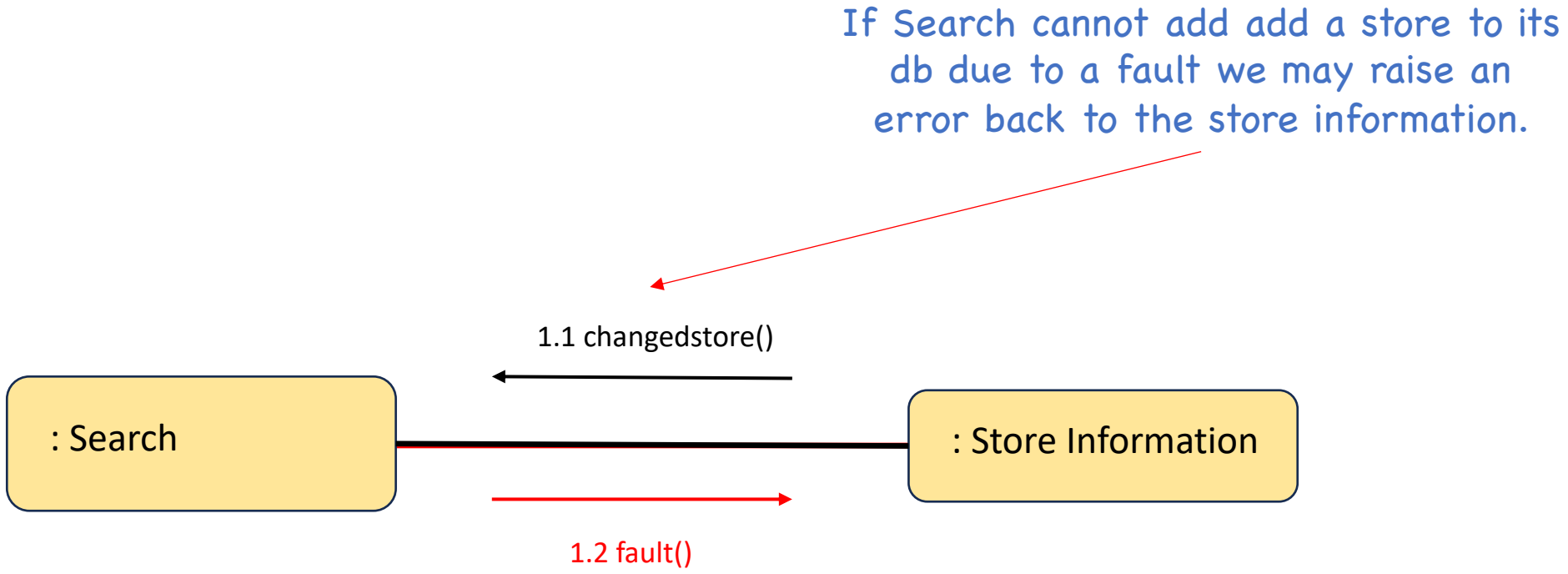
Message Triggers Fault (Robust In-Only)

Messaging	Activity	Repair and Clarification	Conversations



Message Triggers Fault (Robust Out-Only)

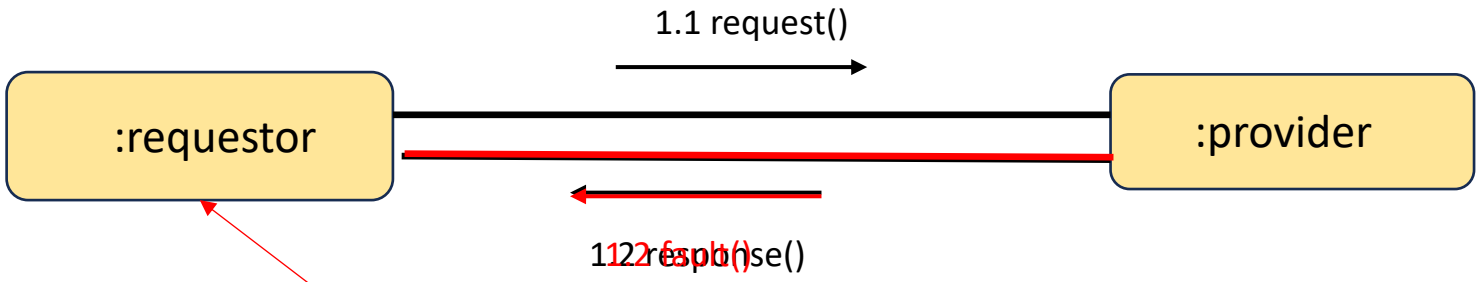
Messaging	Activity	Repair and Clarification	Conversations



Fault Replaces Message

Messaging	Activity	Repair and Clarification	Conversations

Under the Fault Replaces Message pattern a provider propagates faults triggered by an operation by switching to a fault flow, replacing any message subsequent to the first with a fault.



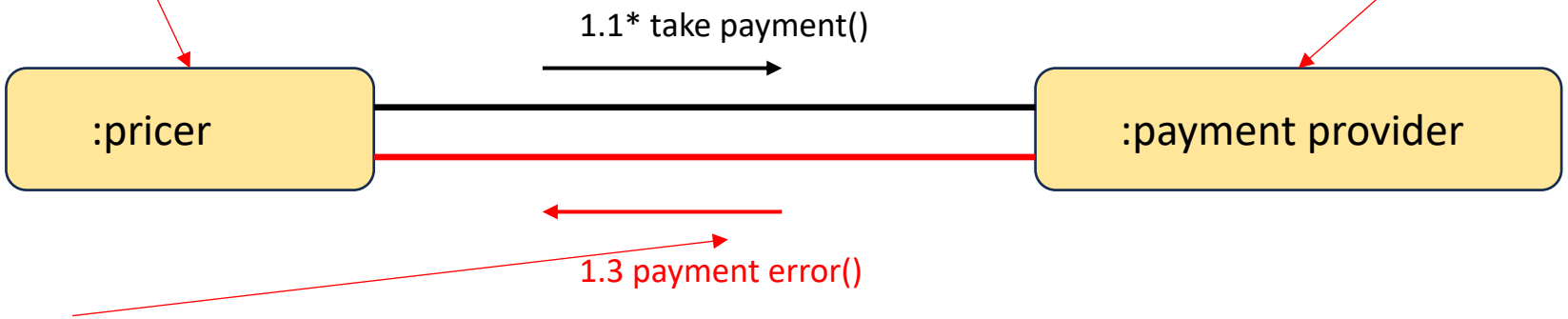
The requestor can handle the error; the error replaces the existing response

Fault Replaces Message (Robust In-Out)

Messaging	Activity	Repair and Clarification	Conversations

Assumption here is that the pricer can orchestrate a new flow, such as asking for an alternate payment method, or cancel the order.

Under Fault Replaces Message pattern the payment **provider** would signal errors taking a card payment back to the pricer



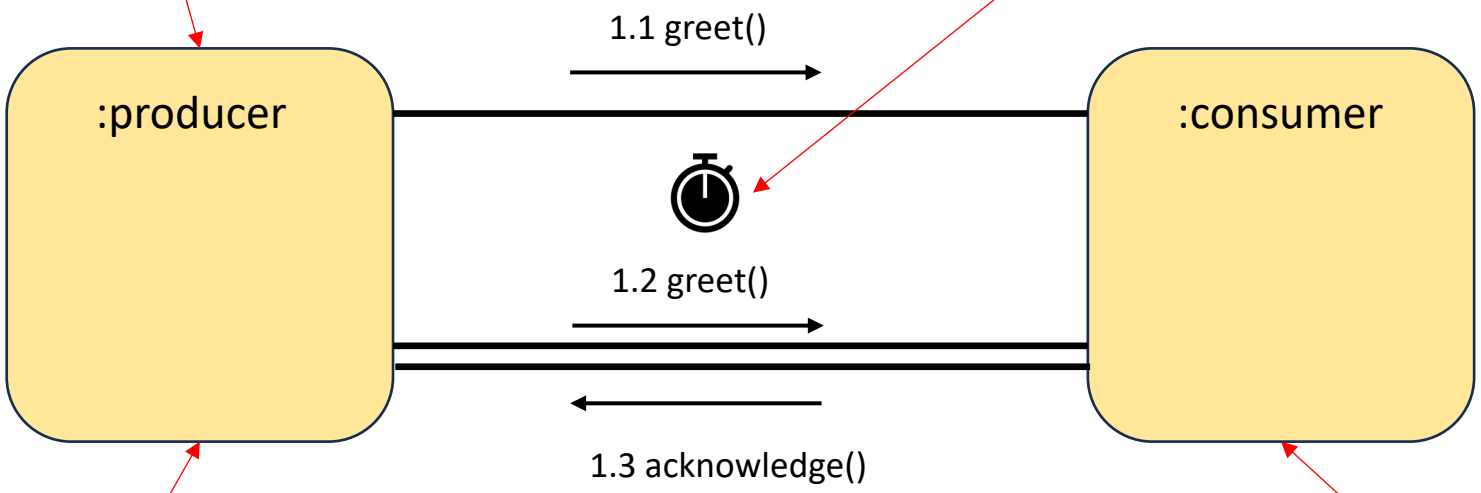
The message should indicate why the payment failed. This might be an issue with the payment provider but it also might be an issue like an invalid card or insufficient funds

In-Out-Retry

Messaging	Activity	Repair and Clarification	Conversations

The producer may not receive an expected response from a consumer. What can it do?

The producer can set a timeout within which to receive a response.

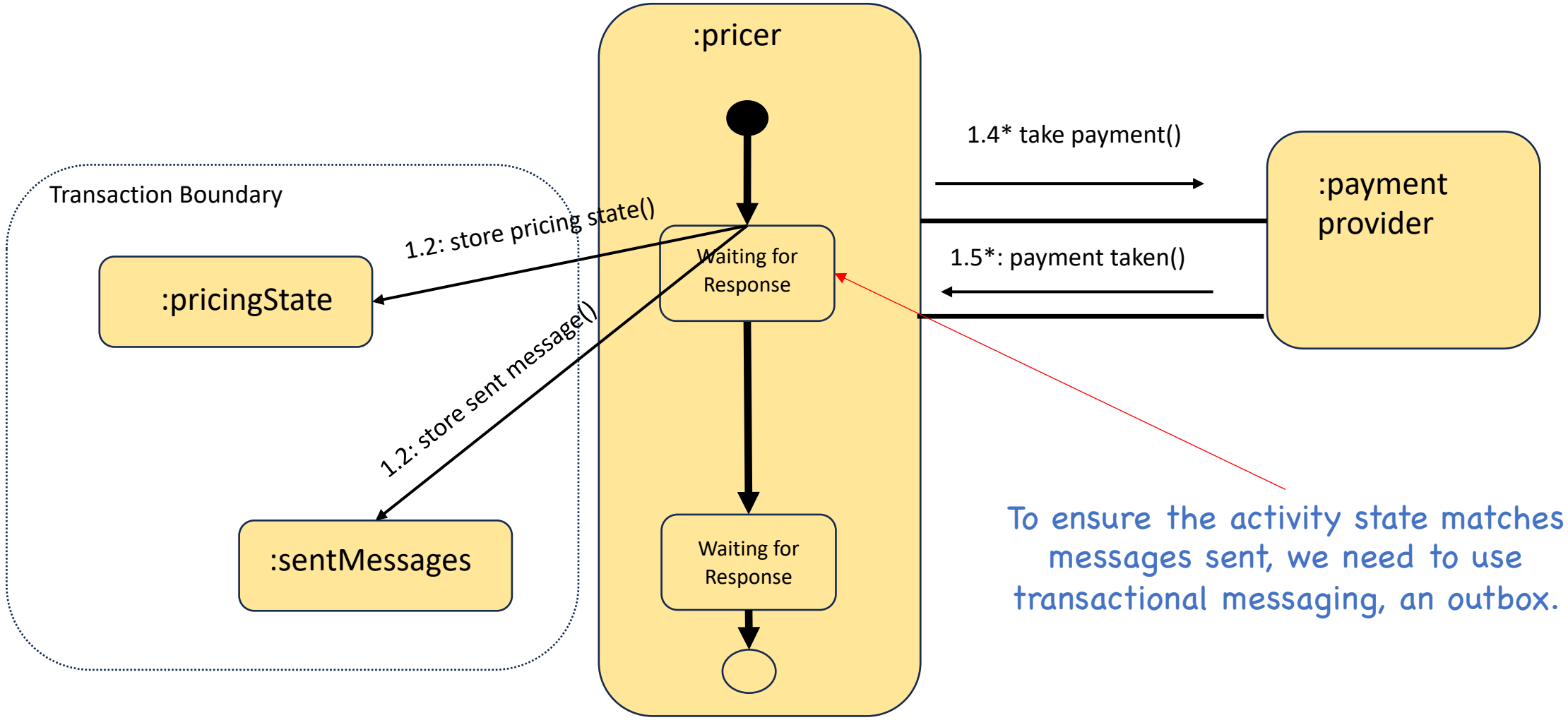


The producer can choose to retry if it does not receive a response within that time window.

Because we might send a message twice, the operation on the consumer must be idempotent, or the consumer must de-duplicate already seen messages

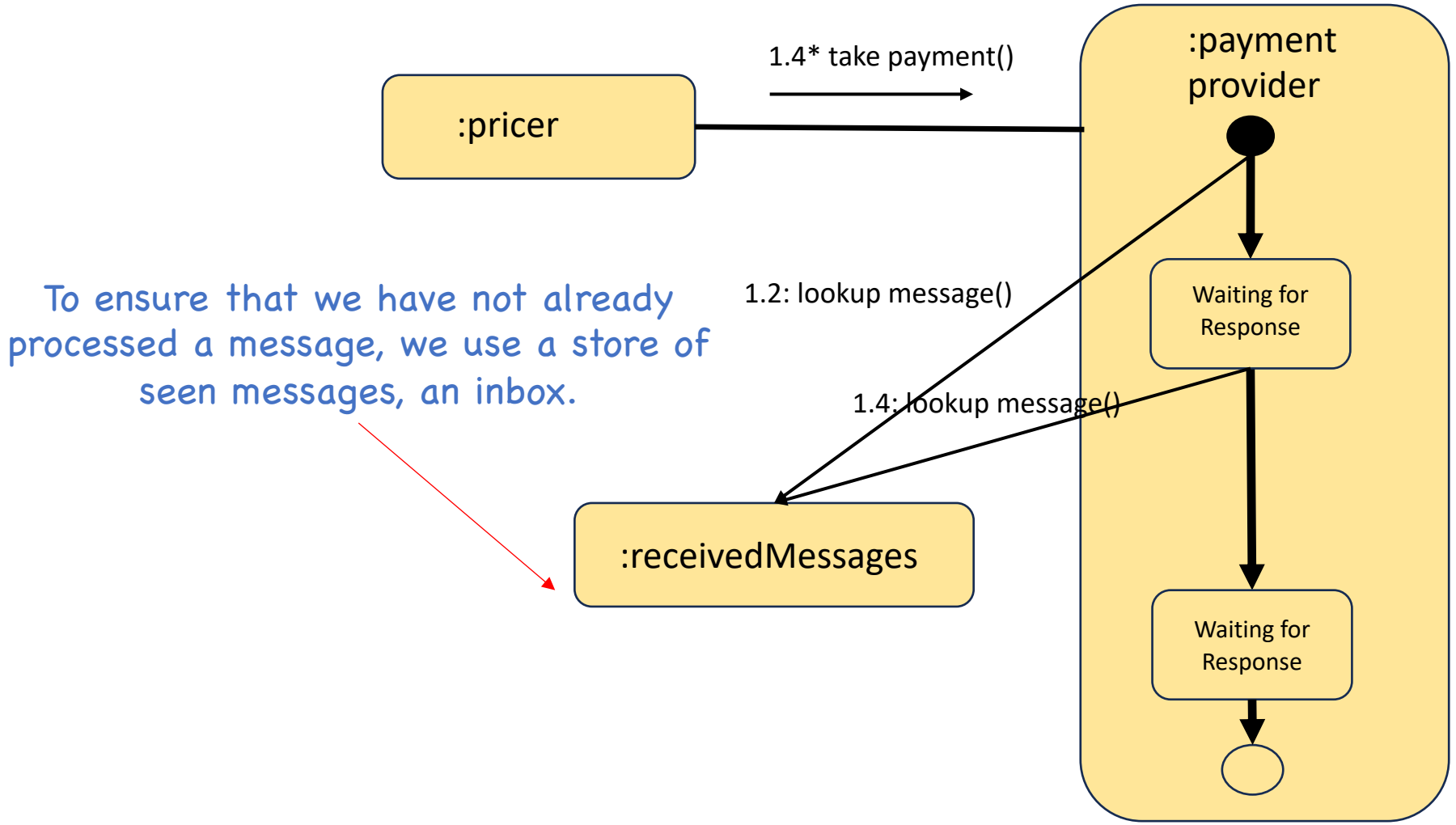
Transactional Messaging (Outbox)

Messaging	Activity	Repair and Clarification	Conversations



Transactional Messaging (Inbox)

Messaging	Activity	Repair and Clarification	Conversations

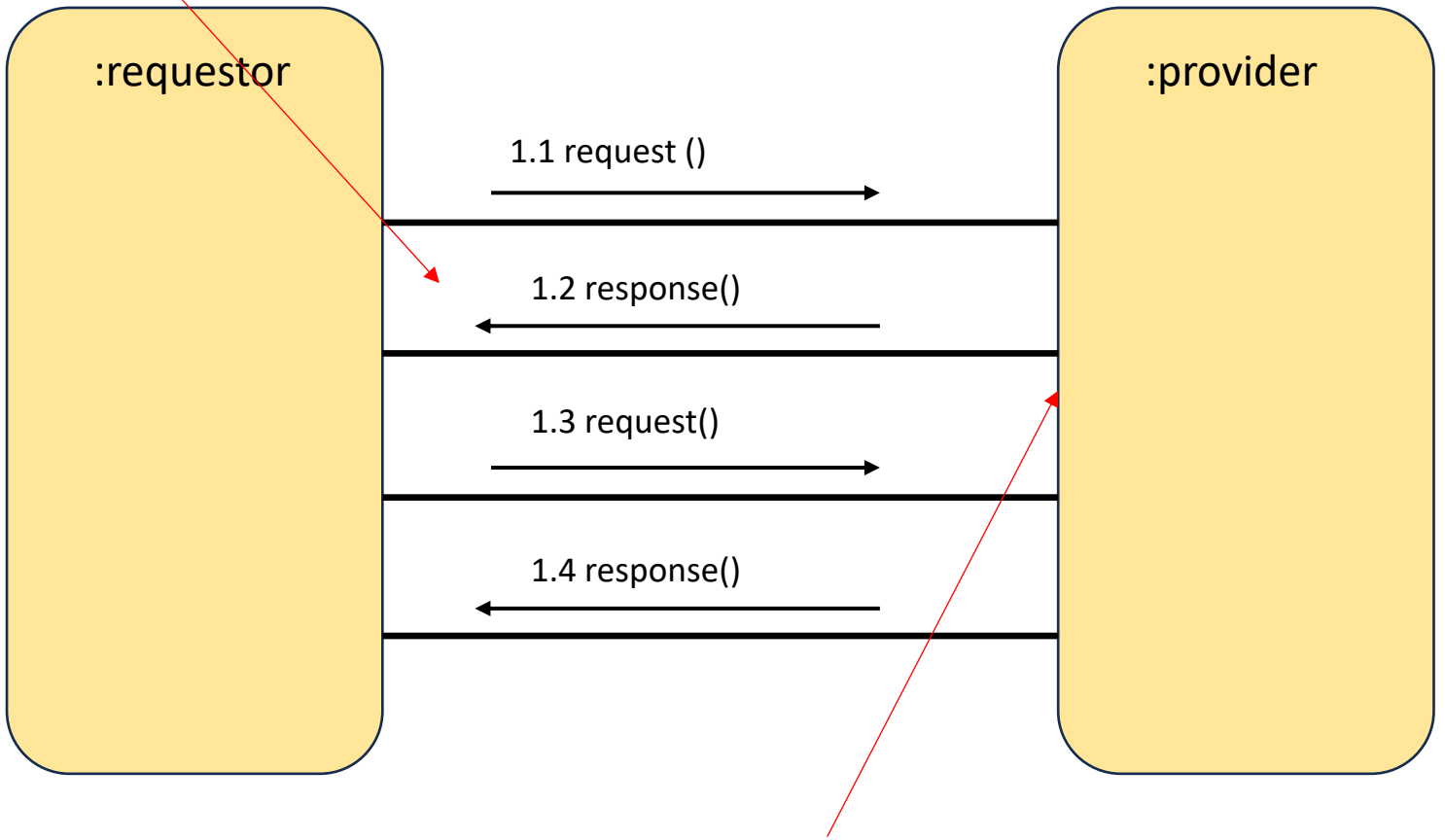


Conversations

Turn Taking

Messaging	Activity	Repair and Clarification	Conversations

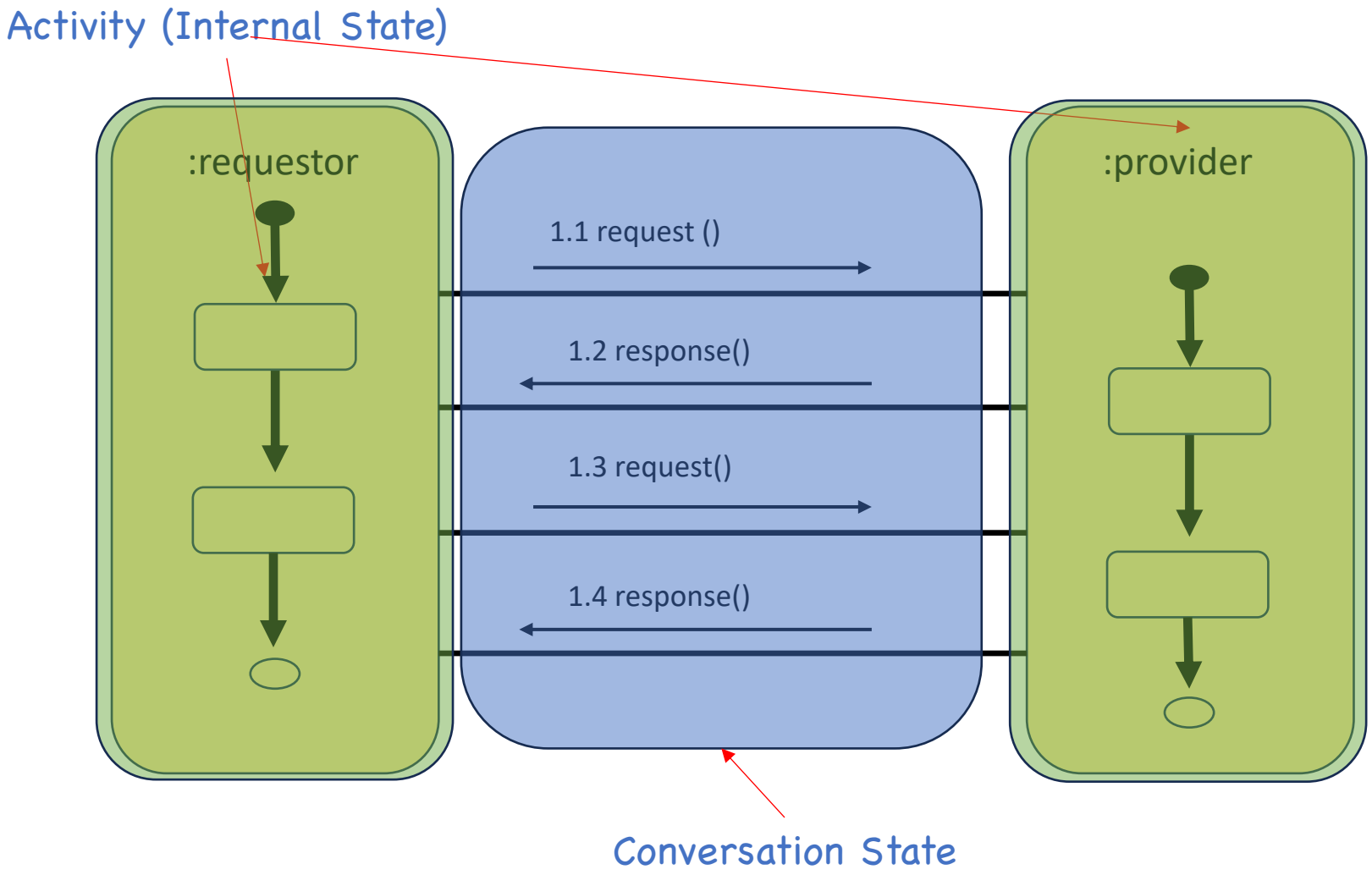
Turn-taking consists of a set of in-out pairs



As each participant takes its turn, it takes control of the flow

Turn Taking

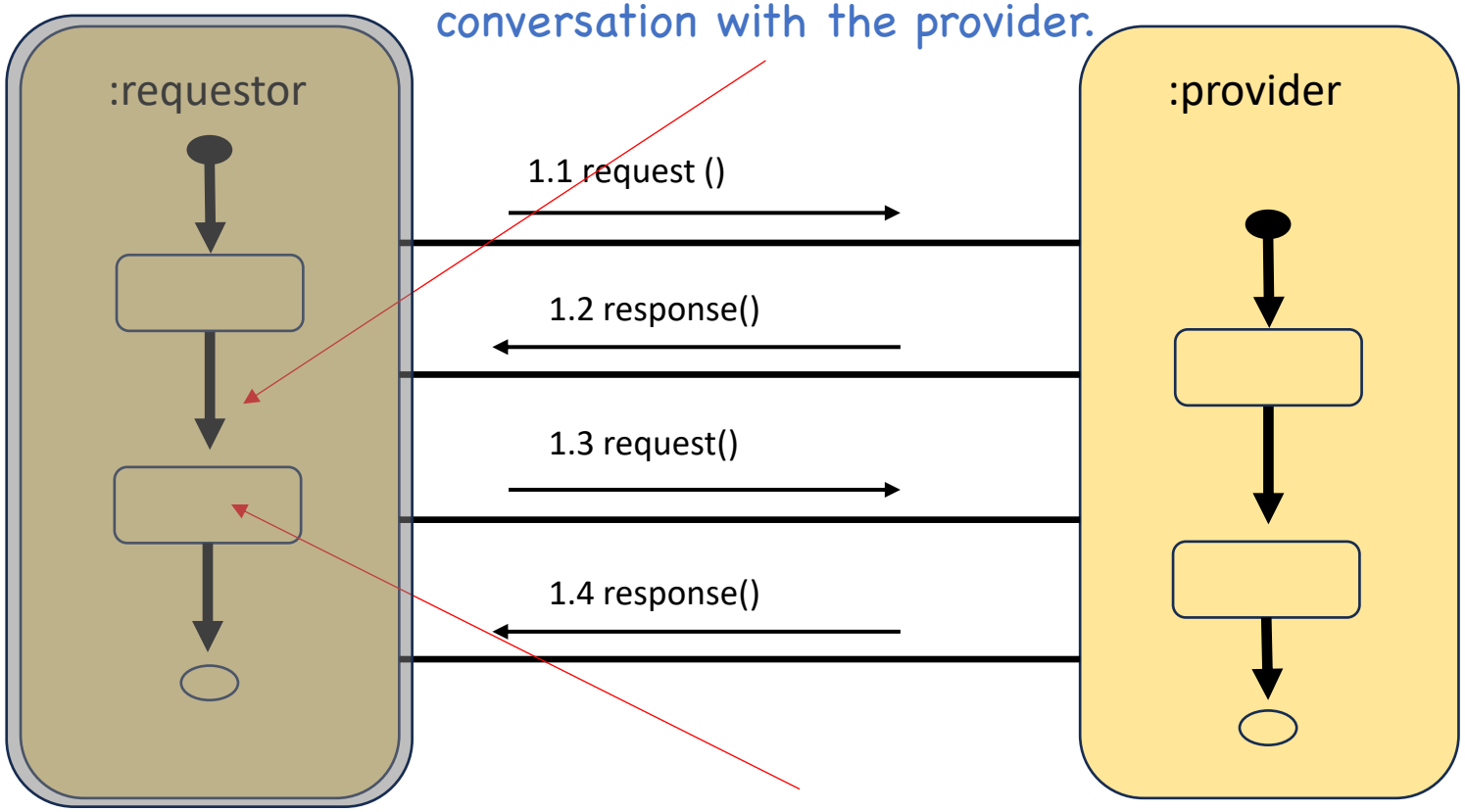
Messaging	Activity	Repair and Clarification	Conversations



Turn Taking

Messaging	Activity	Repair and Clarification	Conversations

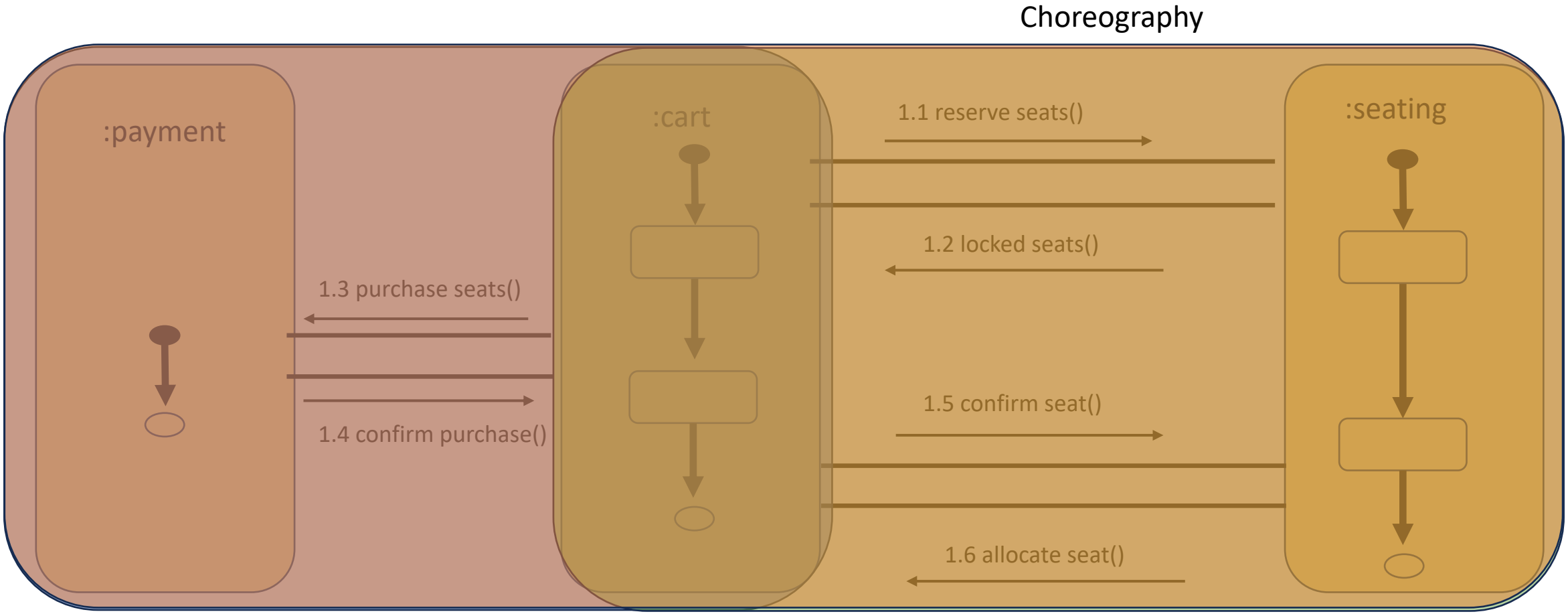
Orchestration: turn taking implies causality, and by implication the requestor's activity orchestrates the conversation with the provider.



Smart Endpoints, Dumb Pipes: we don't need a process manager/saga, as we can use turn-taking instead

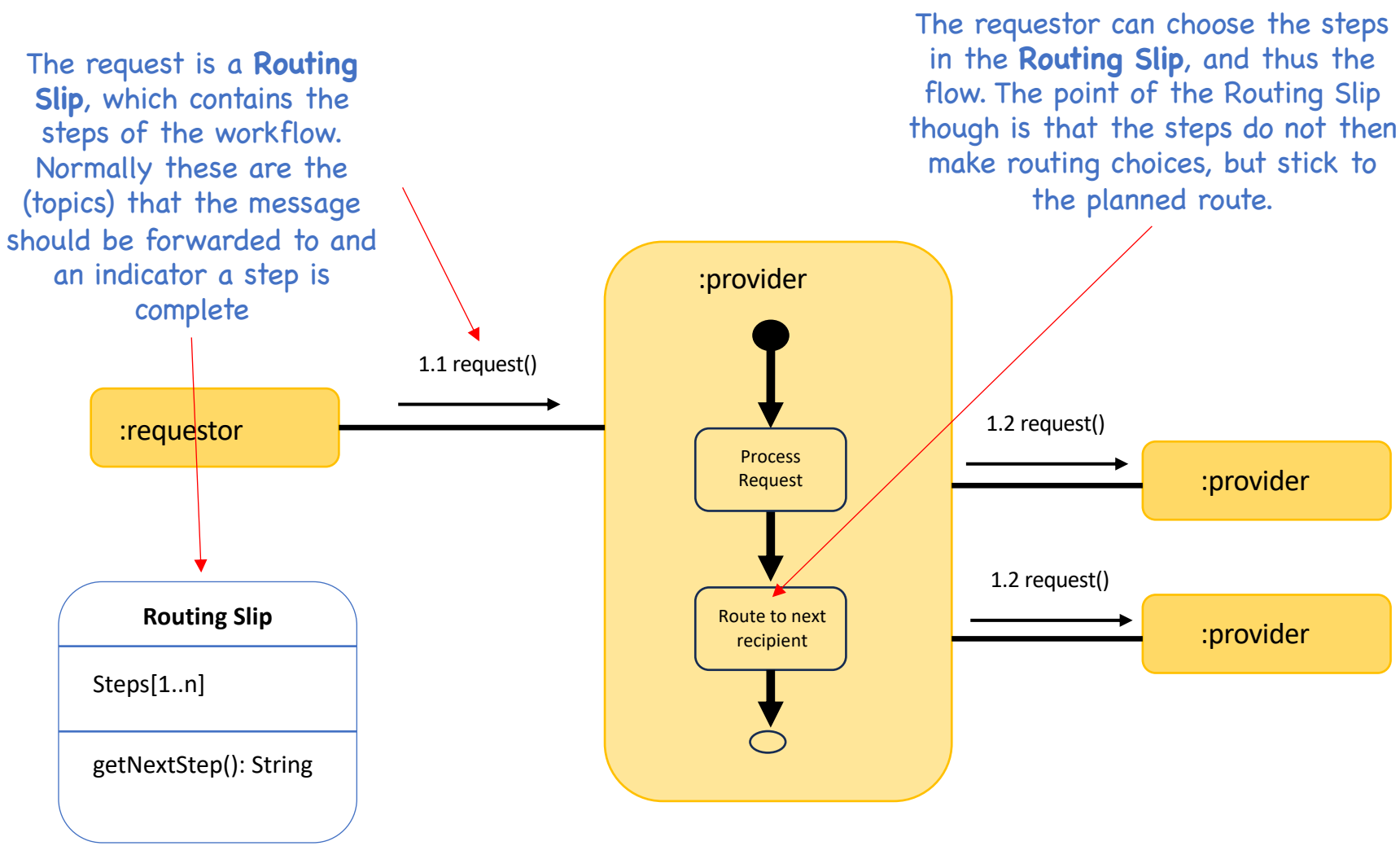
Turn Taking

Conversations	Repair and Clarification	Monologues	Conversations



Routing Slip

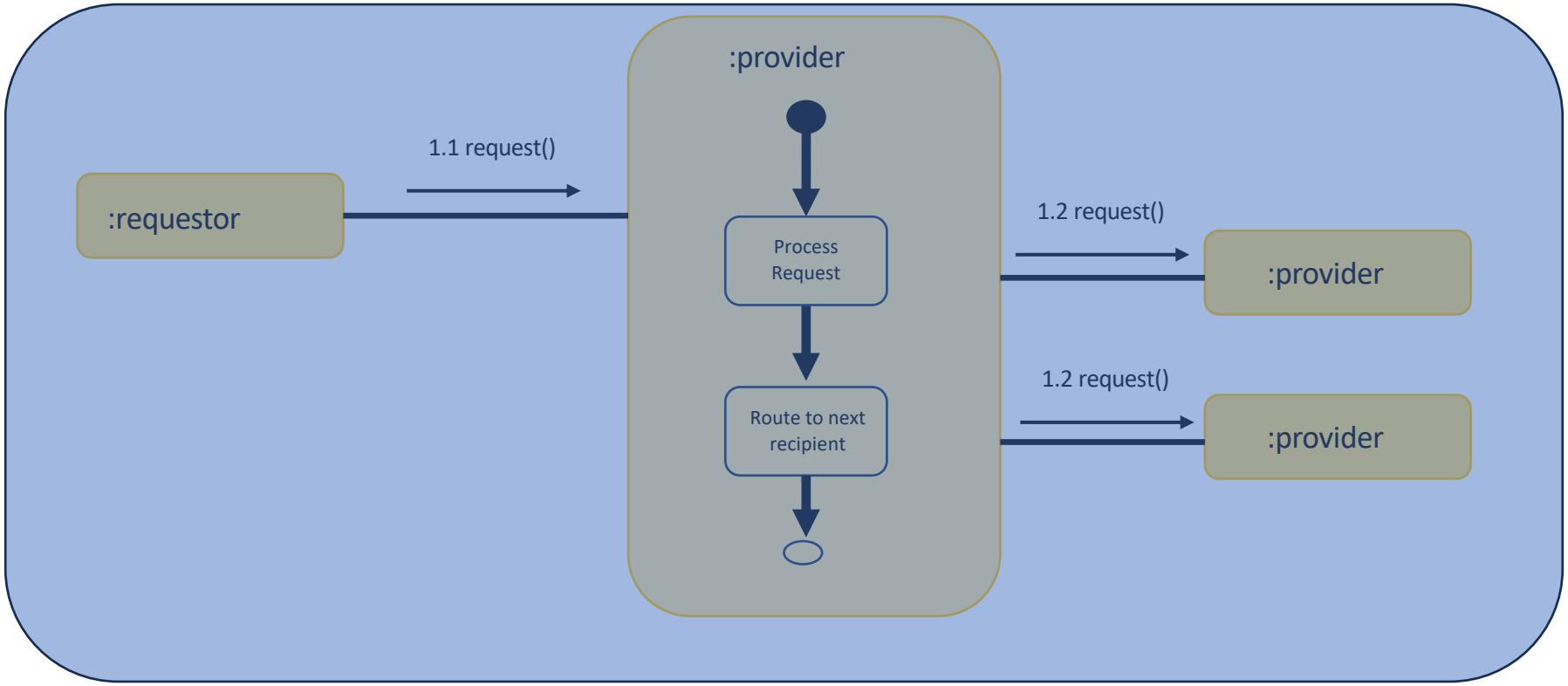
Conversations	Repair and Clarification	Monologues	Conversations



Routing Slip

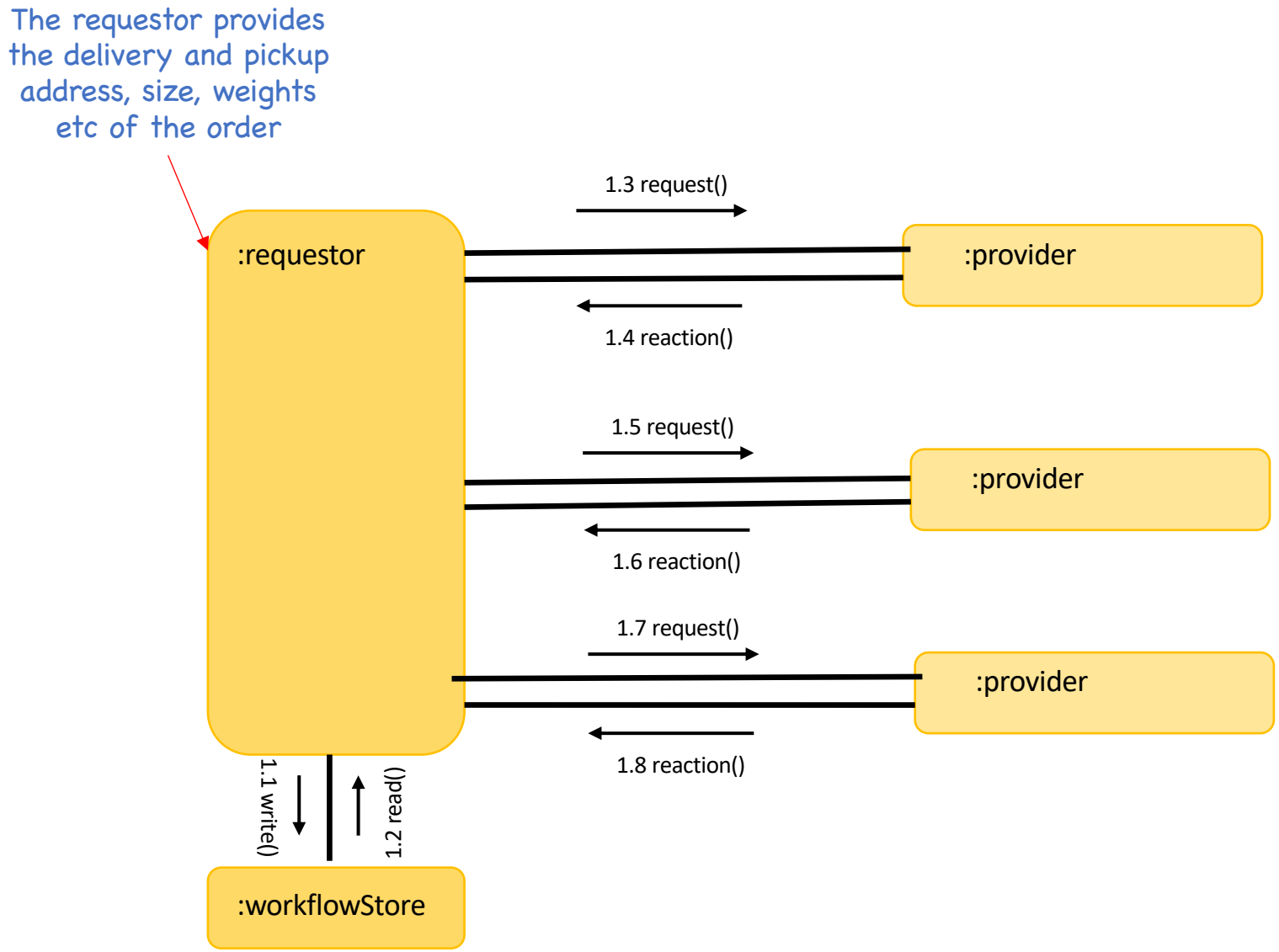
Conversations	Repair and Clarification	Monologues	Conversations

Orchestration



Process Manager (Saga)

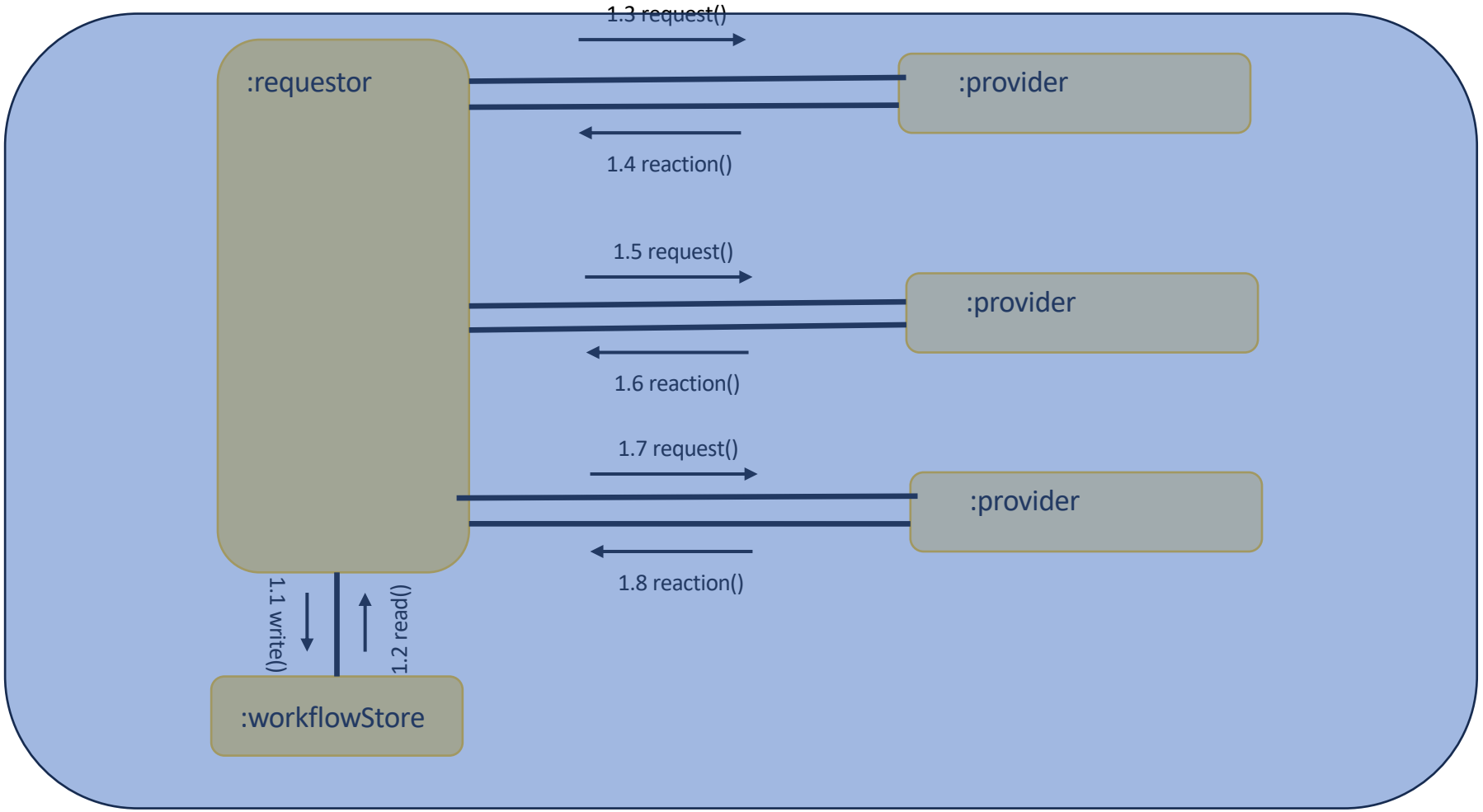
Conversations	Repair and Clarification	Monologues	Conversations



Process Manager (Saga)

Conversations	Repair and Clarification	Monologues	Conversations

Orchestration



Conversation Types

Conversations	Repair and Clarification	Monologues	Conversations

Turn Taking	**Routing Slip**	**Process Manager (Saga)**
Complex Flow	Simple Flow	Complex Flow
Rigid Flow	Dynamic Flow	Dynamic Flow
No central point of failure	No central point of failure	Central point of failure
Distributed	Distributed	Hub-and-Spoke
No central administration or reporting	Central administration but not reporting	Central Administration and Reporting

Q&A