Introducción a los agentes software (parte III)
Ingeniería de Agentes Software y Físicos
Master de Tecnologías de la Información y Telemática Avanzadas

Juan A. Botía

Departamento de Ingeniería de la Información y las Comunicaciones, Universidad de Murcia

Estructura de la clase dedicada a FIPA-IEEE

FIPA-IEEE, primeras ideas

- Organización internacional sin ánimo de lucro
- Dedicada a establecer un marco común y genérico tanto para agentes y sistemas multi-agente
- Su objetivo es "*el permitir la construcción de sistemas que se integren con su entorno de computación particular, mientras que interoperan con sistemas de agentes que residen en entornos heterogeneos, todo con el mínimo esfuerzo*"

Un recorrido rápido por las especificaciones

Divididas en 5 grandes grupos

1. Aplicaciones
2. Arquitectura Abstracta
3. Lenguajes de comunicación
4. Gestión de agentes
5. Transporte de mensajes

La arquitectura de referencia FIPA

La arquitectura de referencia FIPA se denomina "Arquitectura Abstracta" ¿por qué?

- definir los elementos comunes a diferentes sistemas subyacentes
- reunirlos en una especificación común
- instanciar esa arquitectura en forma de implementaciones diferentes pero interoperables !!

La arquitectura de referencia FIPA

Tópicos cubiertos

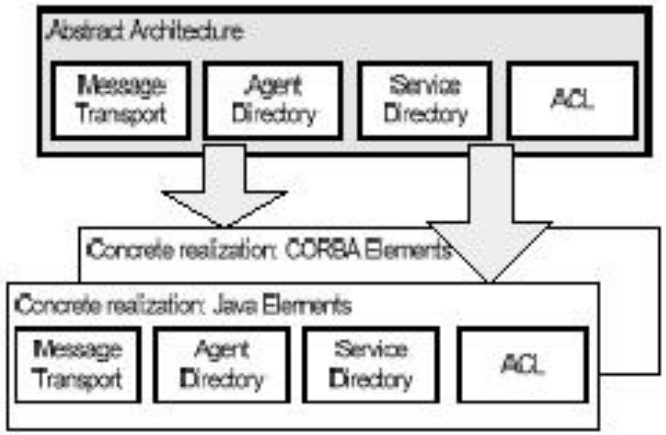
- Definición de un modelo abstracto para el descubrimiento de servicios disponibles para otros agentes y servicios.
- Interoperabilidad entre diferentes medios de transporte de mensajes.
- Soporte para el manejo de diferentes formas de representación de mensajes ACL
- Soporte para el manejo de diferentes lenguajes de contenido
- Soporte para el manejo de diferentes representaciones de servicios de directorio

Tópicos no cubiertos: por ejemplo, la gestión del ciclo de vida de un agente

Realización-materialización de la arquitectura abstracta

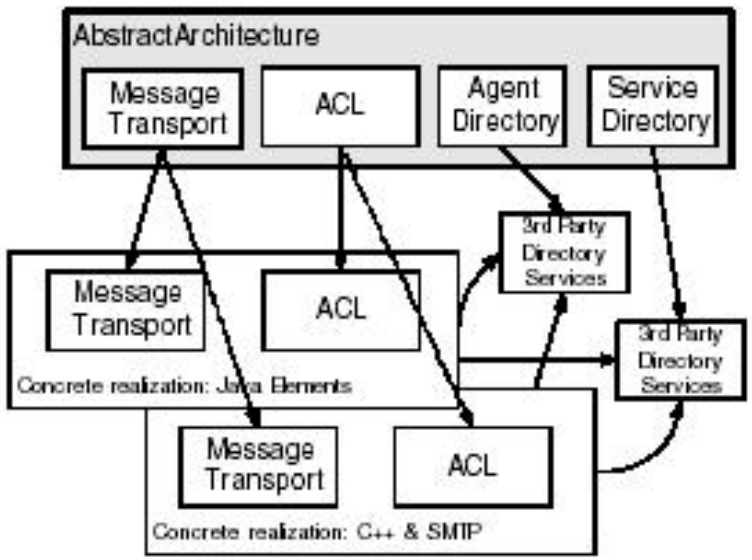
Una arquitectura abstracta no es directamente implementable

- Base para otras especificaciones más concretas (elementos software, lenguajes de programación, protocolos de red, etc)
- El desarrollador tiene libertad para materializar transporte de mensajes, directorio de agentes, directorio de servicios y ACL



Realización-materialización de la arquitectura abstracta

- Una arquitectura abstracta puede incluir elementos adicionales
- una realización puede ser, bien de la arquitectura entera pero también de un único elemento (e.g. directorio de agentes)

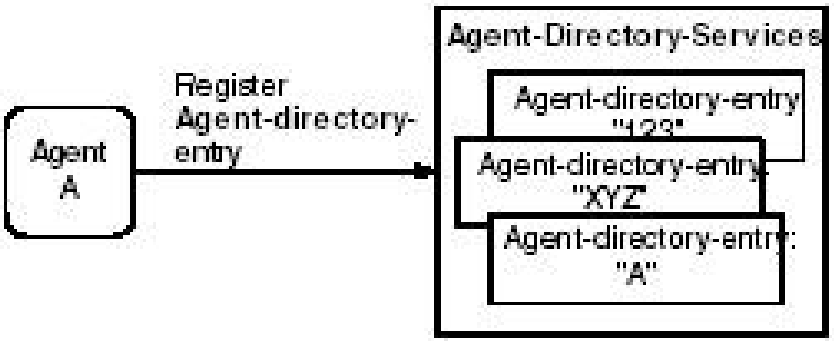


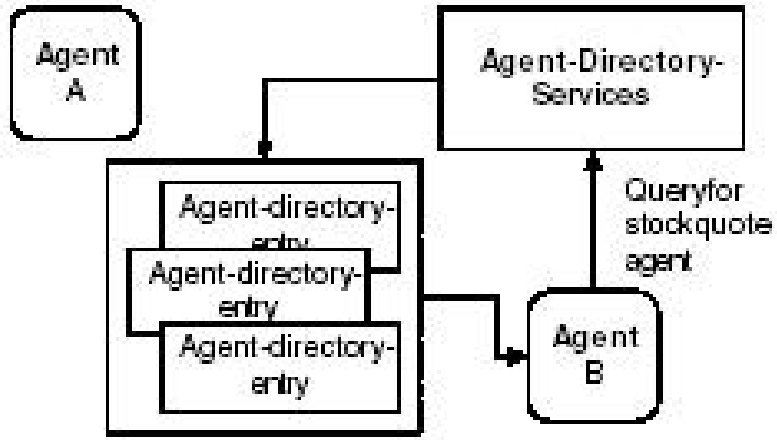
Localizando agentes y servicios en FIPA

- Proporcinar un lugar en donde los agentes registren sus descripciones/servicios de tal forma que otros agentes puedan utilizar ese medio para localizar agentes/servicios con los que deseen interactuar/invocar
- Datos mínimos de un agente
  - AID: nombre único globalmente
  - Localizador: una o más descripciones del transporte que, a su vez, contiene el tipo de transporte (i.e. el protocolo), la dirección de transporte para ese protocolo y cero o más propiedades adicionales
- Cómo anunciarse
  - Primero necesita estar accesible, y por lo tanto se engancha (i.e. hace un *bind*) a una o más direcciones de transporte
  - Luego se da a conocer
    1. crea una entrada de directorio con sus datos de agente
    2. registra la entrada con el servicio de directorio de agentes

Localizando agentes y servicios en FIPA II

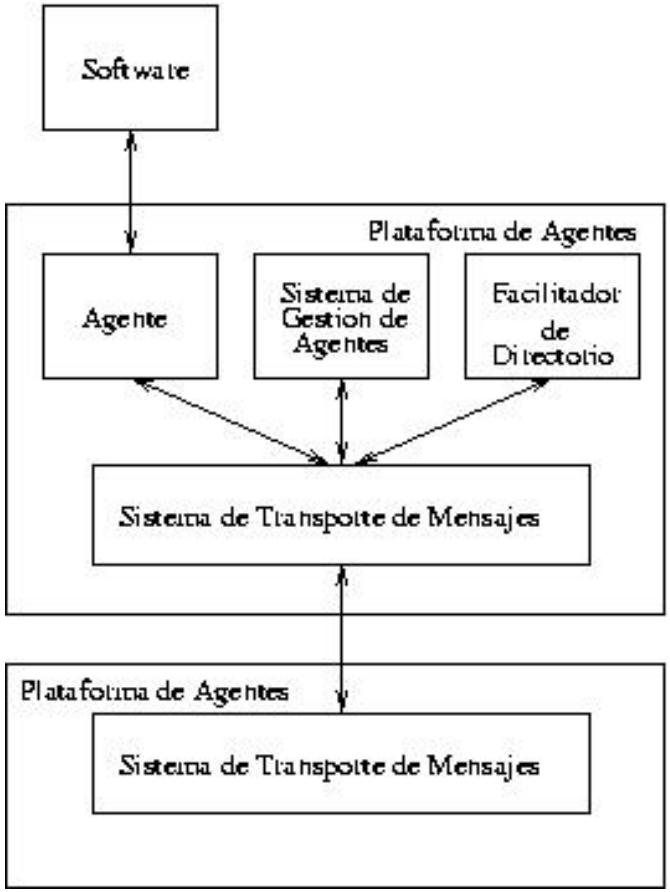
La búsqueda de los agentes es por matching simple de cadenas y pares atributo-valor





El registro y localización de servicios es análogo al de agentes

Panorama global



Elementos del entorno de ejecución

El AMS: habilita el acceso a la plataforma

- Un AMS para cada plataforma de agentes (AP)
- Gestiona la operación la plataforma con respecto a los agentes que viven en ella (ciclo de vida, registro de nombres)
- Operaciones en el directorio: `register`, `deregister`, `modify`, `search` y `get-description`
- Operaciones contra agentes: suspender, terminar, crear, finalizar ejecución, invocar y ejectuar agente

El agente como proceso software en la plataforma

Agente en la plataforma

proceso computacional que implementa la funcionalidad autónoma de comunicación para una aplicación concreta

Identificado mediante el AID (*Agent Identifier*)

- `name`
- `addresses`
- `resolvers`

Facilitador de directorio en la plataforma

Componente opcional

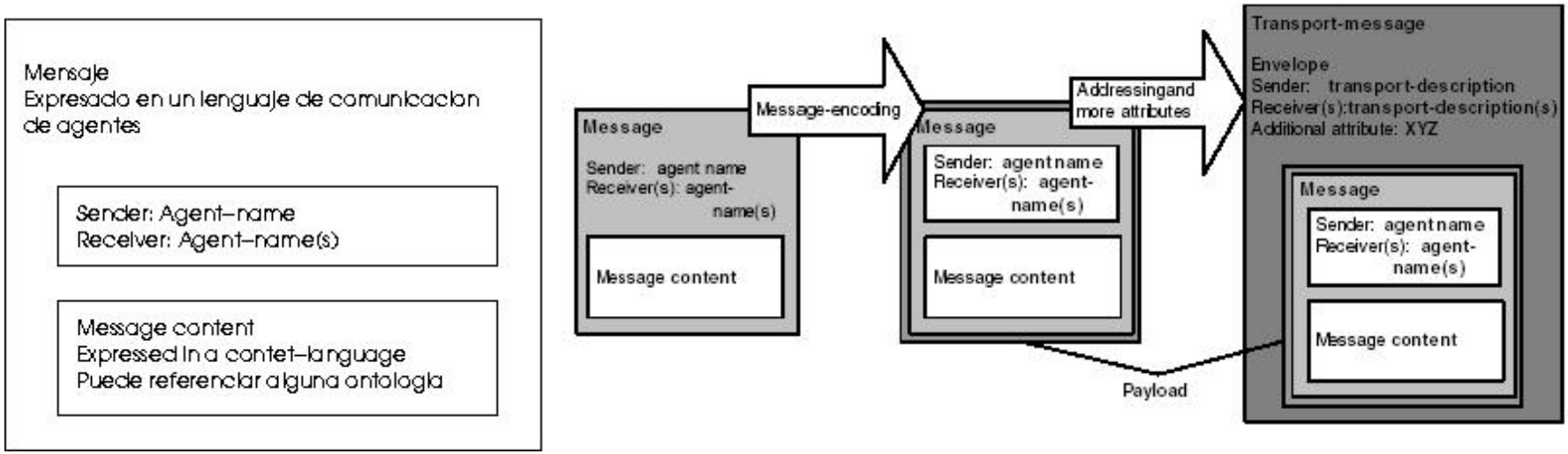
- implementa la funcionalidad de un directorio de servicios (i.e. páginas amarillas) de la arquitectura abstracta
- Si existe, tiene reservado el AID

```
(agent-identifier
   :name df@nombre_plataforma
   :addresses (sequence direcciones_transporte_plataforma))
```

- Posibilita subscripción para escuchar eventos relativos a operaciones (registro, desregistro y modificación) en el directorio de servicios mediante `fipa-subscribe`

Elementos de los mensajes FIPA

Estructura, representación y transporte



Elementos de los mensajes FIPA

- La parte externa del mensaje corresponde al lenguaje de comunicación de agentes
- Si nos referimos a la estandarización ACL de FIPA, este va a consistir nuevamente en una lista de pares atributo-valor

Parameter	Category of Parameters
performative	Type of communicative acts
sender	Participant in communication
receiver	Participant in communication
reply-to	Participant in communication
content	Content of message
language	Description of Content
encoding	Description of Content
ontology	Description of Content
protocol	Control of conversation
conversation-id	Control of conversation
reply-with	Control of conversation
in-reply-to	Control of conversation
reply-by	Control of conversation

El ACL FIPA

- La interoperabilidad se consigue mediante un lenguaje de comunicación de agentes
  - bien definido
  - sin ambigüedades
  - con un aparato formal sólido
- La base de un ACL está compuesta por los actos comunicativos

Actos comunicativos

- Cada uno de las directivas FIPA está definida mediante
  - el resumen en donde se explica resumido el significado del mensaje
  - el contenido del mensaje en donde se detalla qué tipo de contenido debe llevar
  - la descripción que es una explicación detallada del acto comnicativo
  - el modelo formal que es una descripción en SL (*Semantic Language*)
  - un ejemplo de mensaje con el acto comunicativo

Notación para la definición de un AC

Un modelo de acto comunicativo (CA) se representará como sigue:

$$\begin{array}{l} < i, act(j, C) > \\ FP : \phi_1 \\ RE : \phi_2 \end{array} ,$$

en donde i es el agente que ejecuta el CA, j es el receptor, act es el nombre de la performativa, C se refiere al contenido del mensaje (i.e. relativo al dominio de aplicación) y ϕ_1 y ϕ_2 son proposiciones de la lógica. Obsérvese que el mensaje sería

```
(act
  :sender i
  :receiver j
  :content C)
```

El acto comunicativo `inform`

Semántica informal

- Un agente i es capaz de informar a un agente j de que una proposición p es verdad solo si i cree que p es verdad (i.e. $B_i p$)
- El agente i usará `inform` solo si cree que j no conoce p o su negación (i cree que j cree la negación de p, i debería mandarle un `disconfirm`, o un `confirm` si j no está seguro del valor de verdad de p)

La formalización queda como sigue:

$$< i, INFORM(j, \phi) >$$
$$FP : B_i\phi \wedge \neg B_i(Bif_j\phi \vee Uif_j\phi)$$
$$RE : B_j\phi$$

El acto comunicativo `inform`

Ejemplo: el ajente *i* informa al *j* que (es verdad) está lloviendo hoy

```
(inform
   :sender (agent-identifier :name i)
   :receiver (set (agent-identifier :name j))
   :content "weather (today, raining)"
   :language Prolog)
```

El acto comunicativo `request`

Semántica informal

- de entre las FP de la acción a, se cumplen las correspondientes a actitudes del agente i
- el agente i cree que el agente j es el adecuado para ello, y que no tiene a la acción a como objetivo persistente (si así fuera no habría que hacer nada)

Formalizado es:

$$< i, REQUEST(j, a) >$$
$$FP : PF(a)[i \backslash j] \wedge B_i Agent(j, a) \wedge B_i \neg PG_j Done(a)$$
$$RE : Done(a)$$

El acto comunicativo request

Ejemplo: el ajente *i* pide al *j* que abra un fichero

```
(request
  :sender (agent-identifier :name i)
  :receiver (set (agent-identifier :name j))
  :content "open \"db.txt\" for input"
  :language vb)
```

El acto comunicativo `inform-if`

Semántica informal

- Es una acción del tipo macro (se ha de ejecutar realmente un inform)
- Si el agente que recibe la acción cree que la sentencia es verdad, informará de manera afirmativa, si no indicará que es falsa

Formalizado tenemos:

$$< i, INFORM - IF(j, \phi) > \equiv$$
$$< i, INFORM(j, \phi) > \mid < i, INFORM(j, \neg\phi) >$$
$$FP : Bif_i\phi \wedge \neg B_i(Bif_j\phi \vee Uif_j\phi)$$
$$RE : Bif_j\phi$$

Ejemplo de uso del `inform-if`

El agente *i* pide al agente *j* que le informe de que si Lannion está en Normandía con el siguiente mensaje:

```
(request
:sender (agent-identifier :name i)
:receiver
  (set (agent-identifier :name j))
:content
  ``((action
    (agent-identifier :name j)
    (inform-if
    :sender (agent-identifier :name j)
    :receiver (set
      (agent-identifier :name i))
    :content \"in(lannion,normandy)\"
    :language Prolog)))''
:language fipa-sl)
```

y el agente *j* le responde que no

```
(inform
  :sender (agent-identifier :name j)
  :receiver (set
    (agent-identifier :name i))
  :content ``\+ in(lannion, normandy)''
  :language Prolog)
```

El acto comunicativo `refuse`

Semántica informal

- Se realiza cuando el agente emisor no puede cumplir todas las FP de las acciones que se le ha pedido que ejecute, e.g.
  - no se sabe por algo que se pregunta
  - no se tienen privilegios para ejecutar una acción
- Se acompaña de una explicación

Formalmente

$$
\begin{aligned}
&< i, REFUSE(j, < i, act >, \phi) > \equiv \\
&< i, DISCONFIRM(j, Feasible(i, < act >)) >; \\
&< i, INFORM(j, \phi \wedge \neg Done(< i, act >) \wedge \neg I_i Done(< i, act >)) > \\
&\quad FP : B_i \neg Feasible(< i, act >) \wedge B_i(B_j Feasible(< i, act >) \vee \\
&\qquad U_j Feasible(< i, act >)) \wedge B_i \alpha \wedge \neg B_i(Bif_j \alpha \vee Uif_j \alpha) \\
&\quad RE : B_j \neg Feasible(< i, act >) \wedge B_j \alpha
\end{aligned}
$$

siendo $\alpha = \phi \wedge \neg Done(< i, act >) \wedge \neg I_i Done(< i, act >)$

Recomendaciones FIPA para PIs

Protocolos de interacción Los siguientes son estándares:

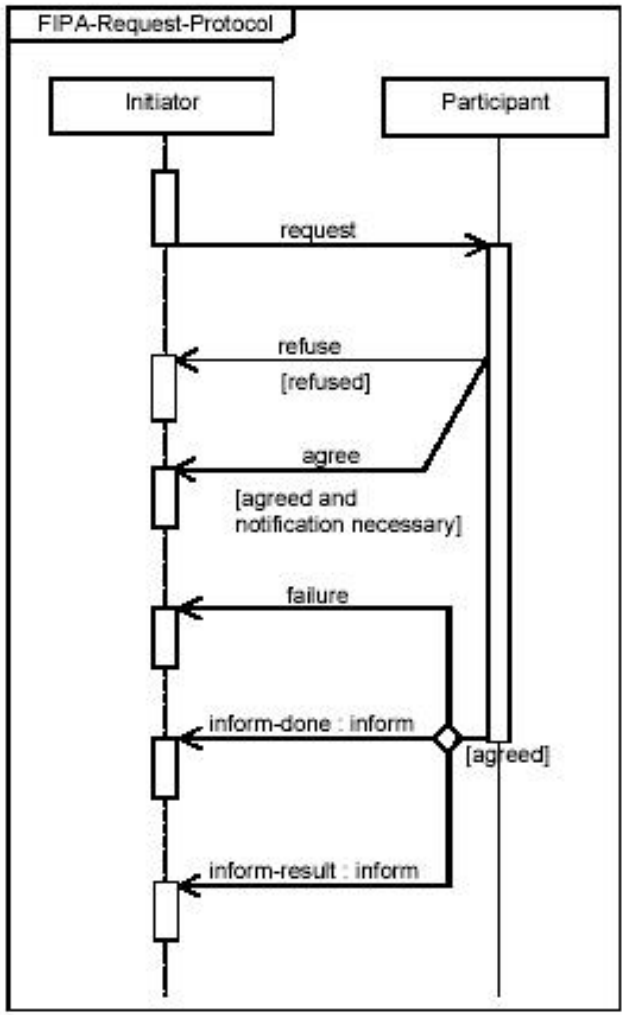
- FIPA Request Interaction Protocol Specification
- FIPA Query Interaction Protocol Specification
- FIPA Request When Interaction Protocol Specification
- FIPA Contract Net Interaction Protocol Specification
- FIPA Iterated Contract Net Interaction Protocol Specification
- FIPA Brokering Interaction Protocol Specification
- FIPA Recruiting Interaction Protocol Specification
- FIPA Subscribe Interaction Protocol Specification
- FIPA Propose Interaction Protocol Specification

y las subastas alemana e inglesa mantienen el estatus de experimental

- FIPA English Auction Interaction Protocol Specification
- FIPA Dutch Auction Interaction Protocol Specification

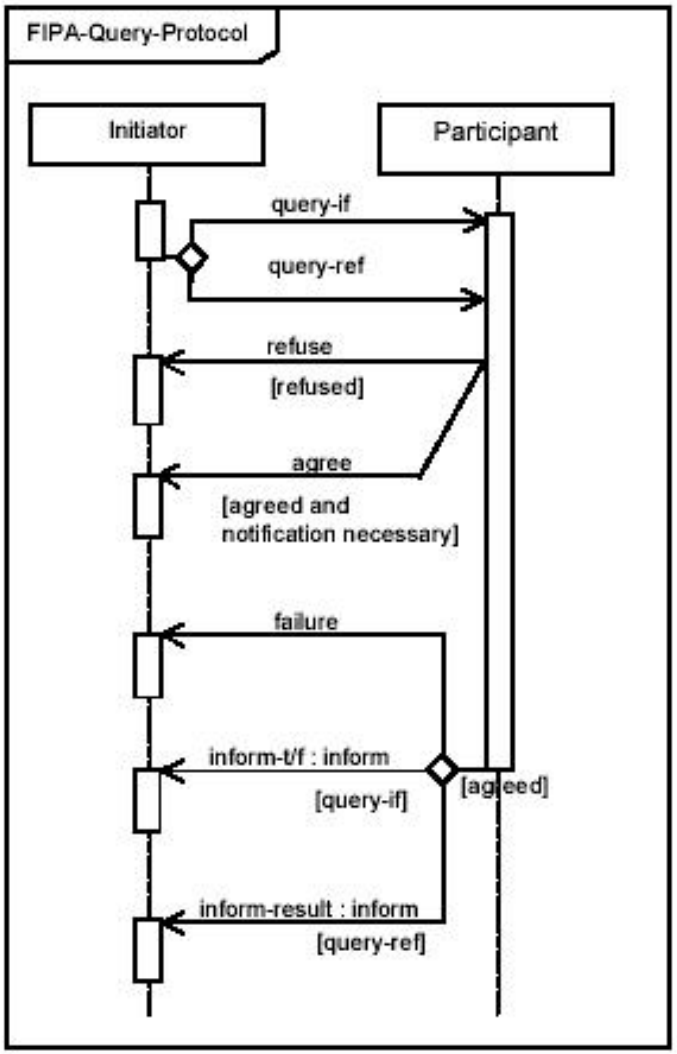
El PI `fipa-request`

El protocolo `fipa-request` permite a un agente pedir a otro que realice una determinada acción.



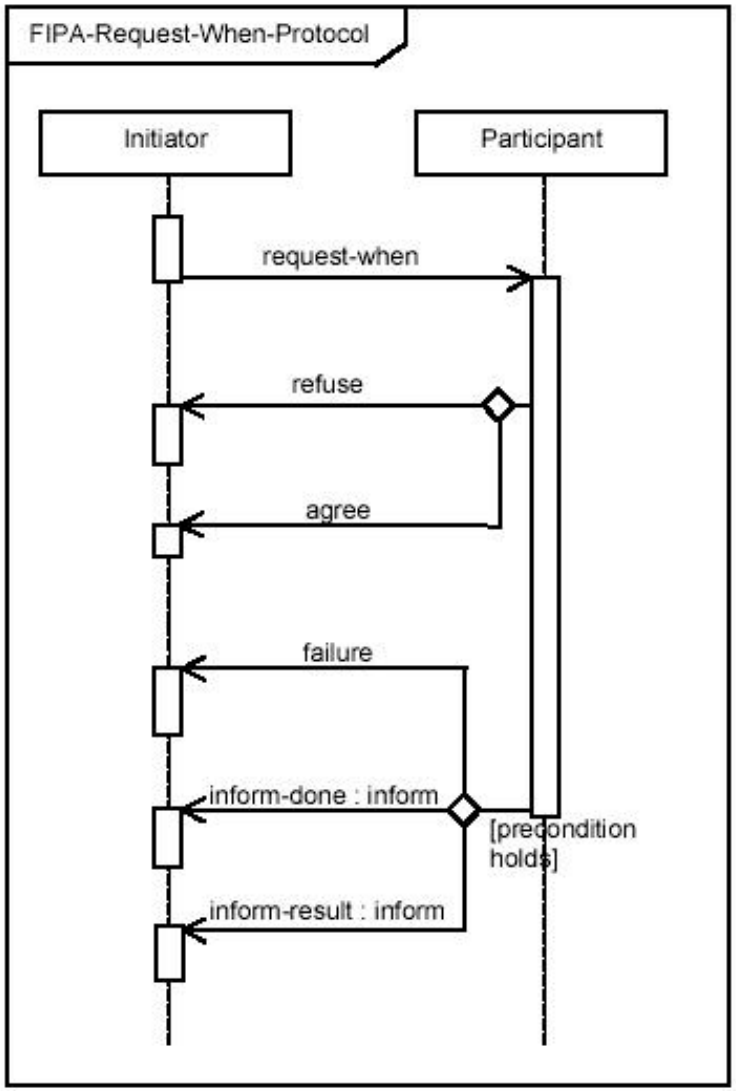
El PI `fipa-query`

El protocolo `fipa-query` es similar al anterior solo que la acción a ejecutar es de un tipo concreto: `inform` para informar sobre algo.



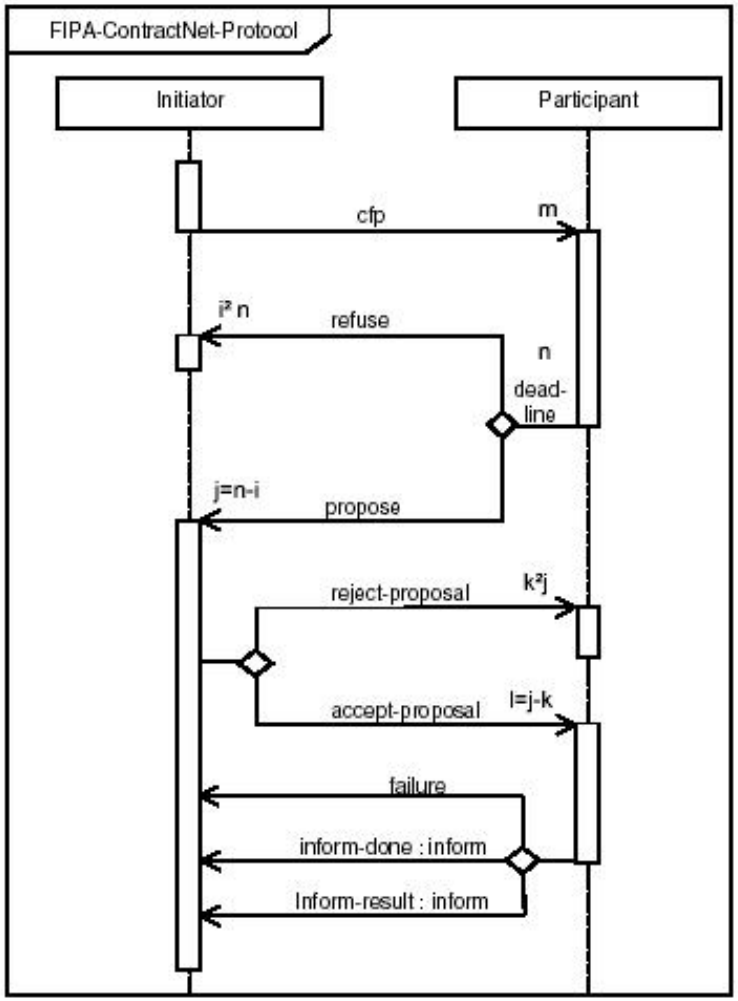
El PI `fipa-request-when`

Es posible indicar a un agente que ejecute una acción de manera diferida, cuando una determinada precondición se haga cierta. Para eso tenemos el protocolo de interacción `fipa-request-when`



El PI `fipa-contract-net`

El fipa-contract-net está inspirado en la propuesta de Smith solo que debido a la autonomía hay que incluir mensajes para confirmación y rechazo.

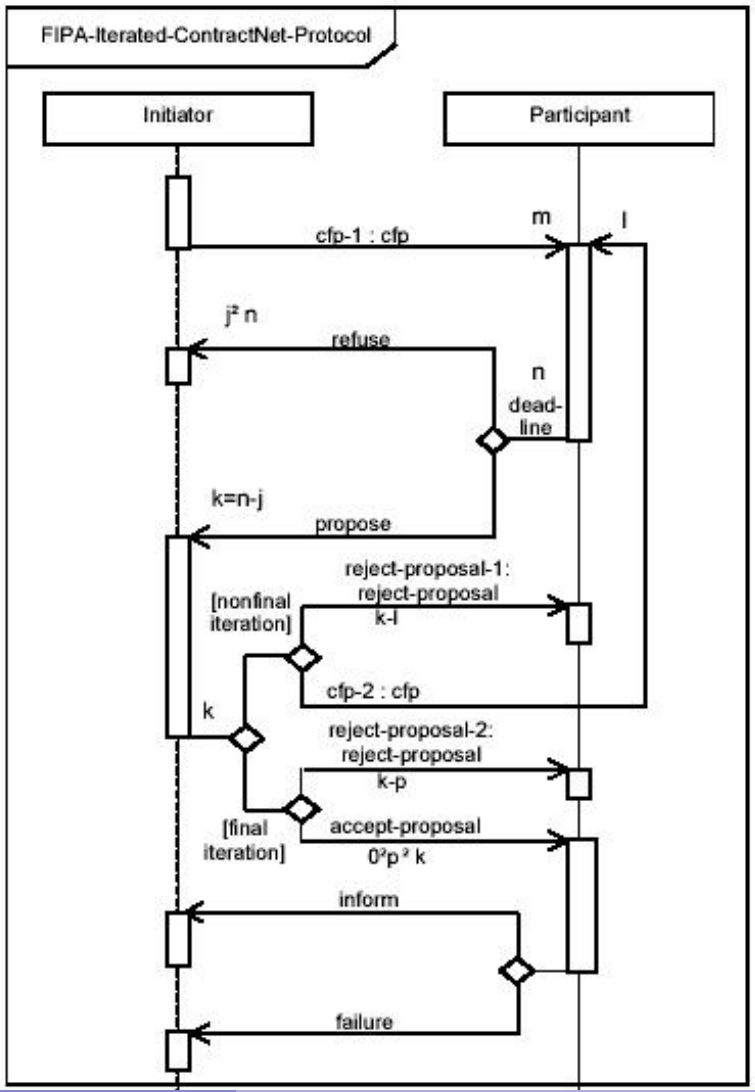


El PI `fipa-contract-net` (y II)

- Inicialmente, el manager, con el rol FIPA de iniciador, genera m mensajes del tipo `cfp` (*call for proposal*) y queda a la espera durante un determinado tiempo, después del cual no recibirá más mensajes (un total de n recibidos)
- Sea i el número de mensajes de tipo `refuse`, Tendremos entonces $j = n - i$ mensajes del tipo `propose`
- Para cada uno de los j mensajes, enviar posteriormente bien un `accept-proposal` o un `reject-proposal`
- Se informa del resultado con un `failure`, con un `inform` sin resultado o con el mismo acompañado del resultado.

El PI `fipa-iterated-contract-net`

- ligera variación del `fipa-contract-net` que permite el número de rondas que sea necesario
- tiene el propósito para el iniciador de la interacción de conseguir mejores ofertas de los interlocutores mediante argumentación o crítica de las ofertas anteriores



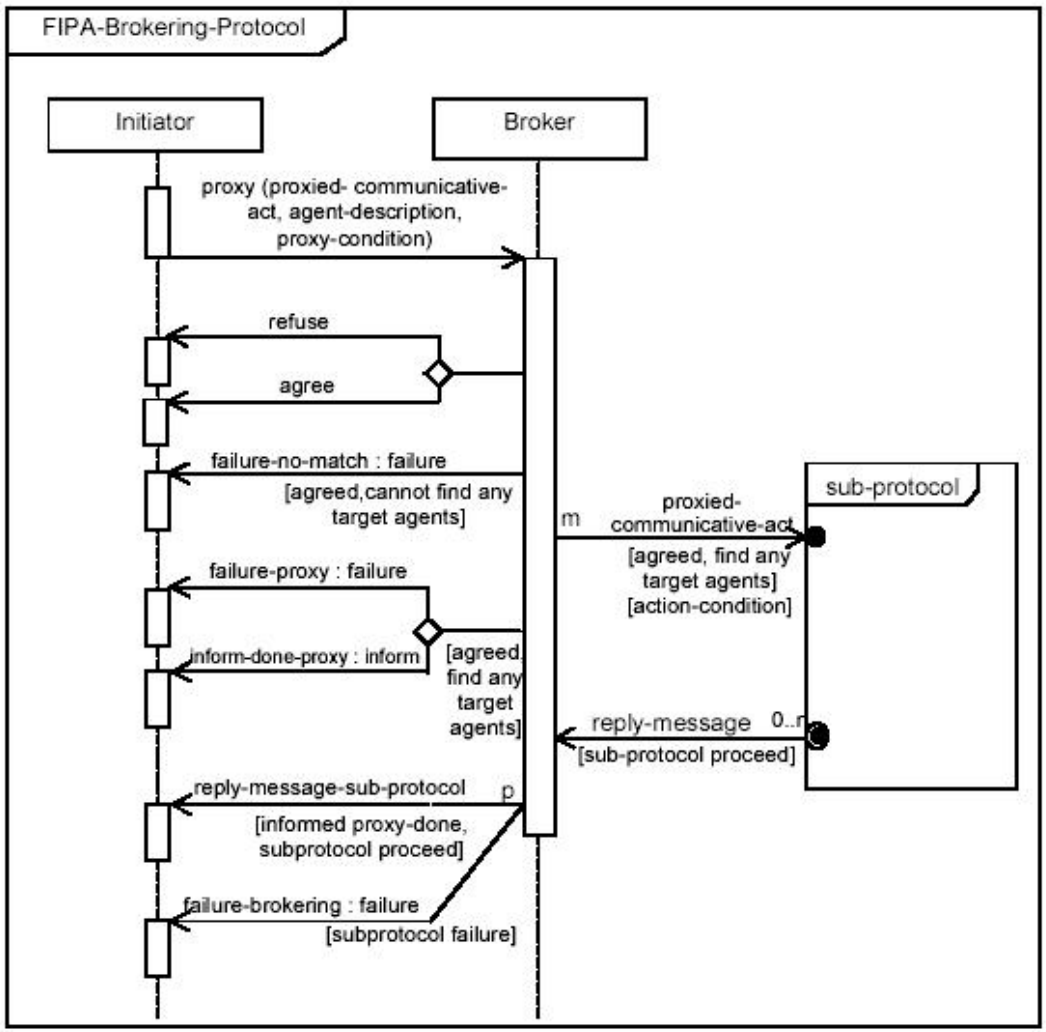
El PI `fipa-iterated-contract-net` (y II)

- Inicialmente, el agente con el rol de iniciador de la conversación genera m mensajes $\widehat{\text{cfp}}$
- Después de un deadline de espera, el iniciador recoge digamos n ofertas
- sean un total de j las que rechazan realizar la tarea mediante `refuse`
- Tenemos entonces $k = n - j$ ofertas de agentes que están dispuestos a realizar la tarea con `prospose`
- Si la iteración no es la última, de entre el total de k ofertas recibidas, se rechazarán algunas directamente, $k - l$, y se aceptarán otras tantas l.
- De entre las aceptadas, se elabora una contraoferta para cada agente y se envuelve en un nuevo `cfp`

El PI `fipa-brokering`

- Tiene como propósito permitir interaccionar con otros agentes a través de un mediador (el broker)
- `proxy` es una macro (incluye otro acto comunicativo que el broker debe hacer llegar al seleccionado o seleccionados)
- El broker devuelve los resultados mediante `reply-message-sub-protocol` (i.e. un `reply` con la respuesta en el cuerpo del mensaje)

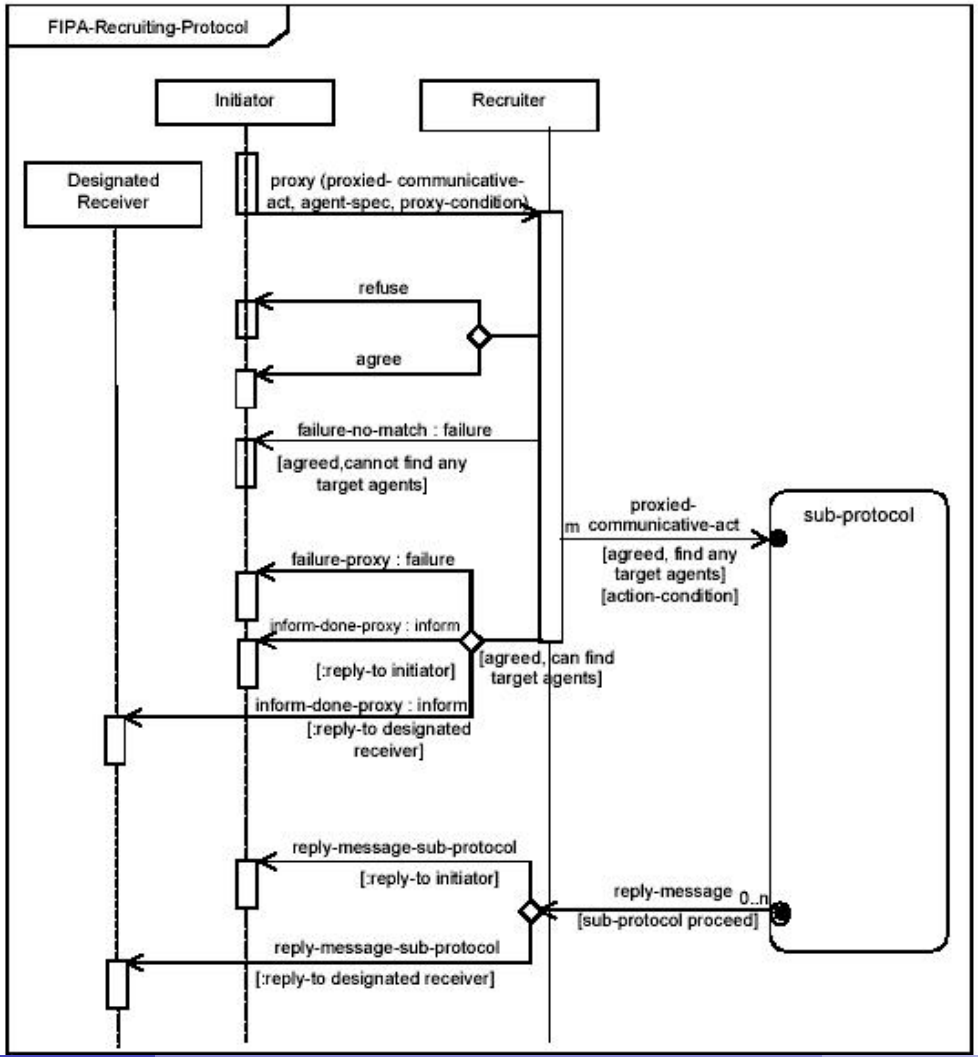
El PI `fipa-brokering` (y II)



El PI `fipa-recruiting`

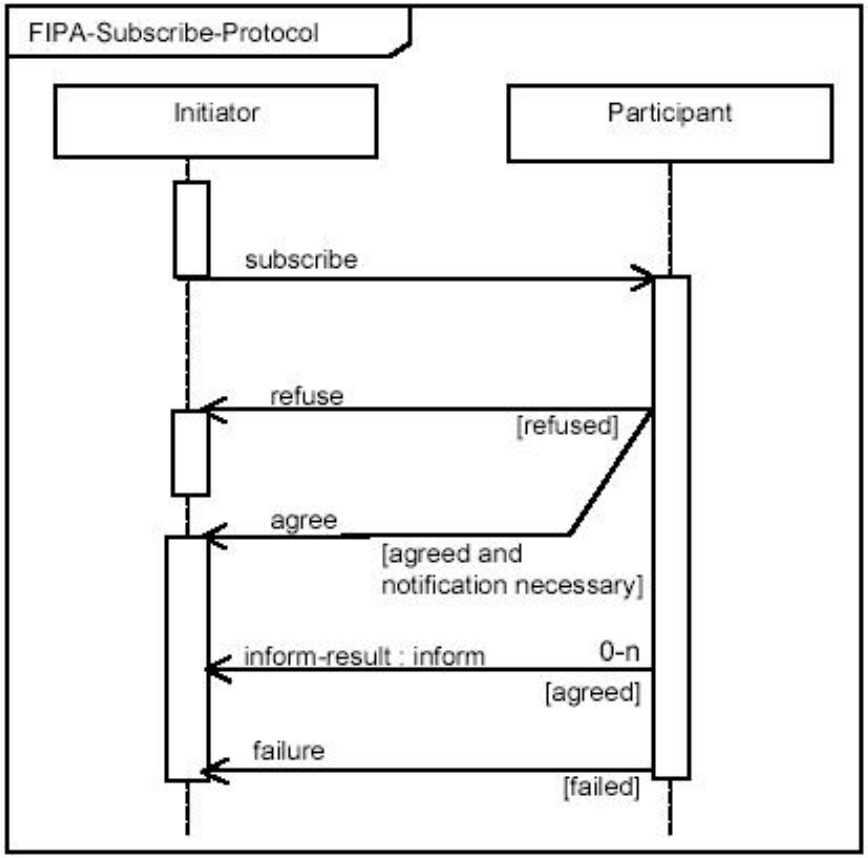
- Tiene como propósito reclutar agentes para interaccionar con ellos posteriormente
- Usado en entornos basados en mediadores
- Se diferencia del anterior en que ya no es el broker el que interactúa con los seleccionados, sino el receptro designado

El PI `fipa-recruiting` (y II)



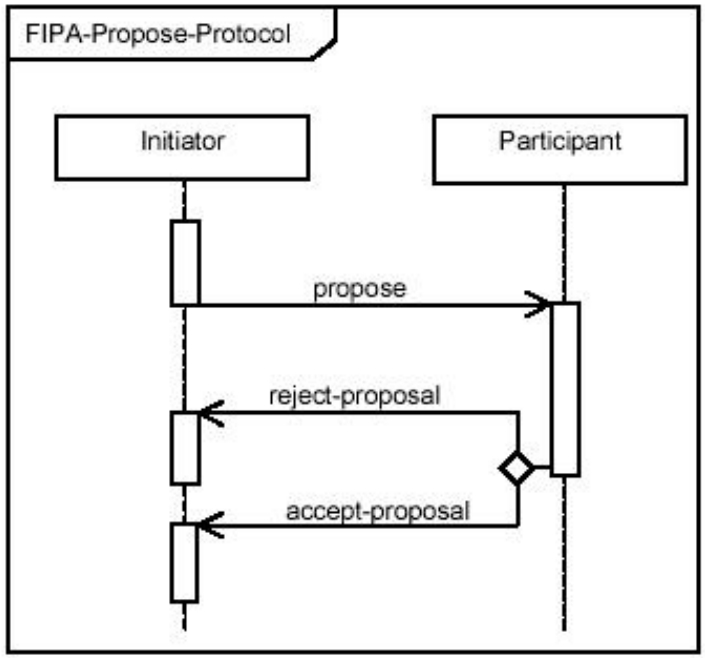
El PI `fipa-subscribe`

De utilidad para subscripción a un servicio de notificación de eventos. El agente participante se compromete (mediante la emisión de un `agree`) a informar mediante `inform` cada vez que el evento se produzca



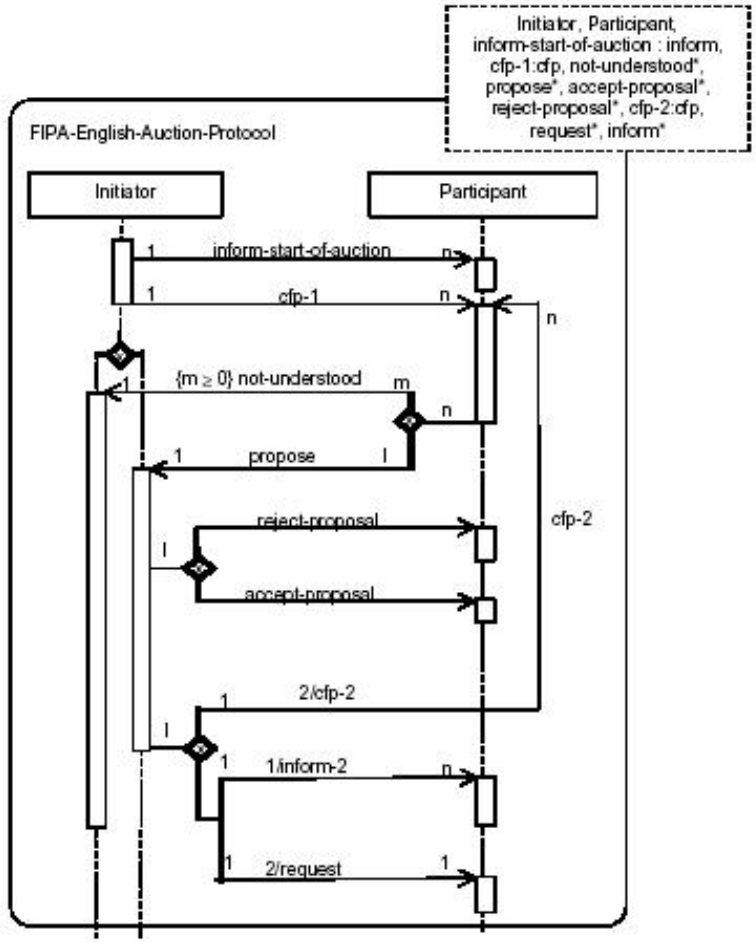
El PI `fipa-propose`

El agente iniciador propone al otro agente particante, que lo ha de supervisar, el realizar una acción



El PI `fipa-english-auction`

El iniciador ofrece un precio. Si ningún posible comprador realiza una oferta, se vuelve a ofrecer otro precio. Se selecciona el mejor de entre los participantes que emiten un `propose`.



Recordatorio de las especificaciones relacionadas

Lenguajes de contenido:

- FIPA SL Content Language Specification
  - El estándar FIPA
- FIPA CCL Content Language Specification
  - Para representar problemas de satisfacción de restricciones
- FIPA KIF Content Language Specification
  - No es posible representar otra cosa que no sea una expresión con valor de verdad o hechos
- FIPA RDF Content Language Specification
  - El más práctico y utilizado

de las cuales, solamente la correspondiente al lenguaje SL tiene consideración de estándar

El estándar SL

- Sintaxis muy parecida a la de LISP
- Sus f.b.f. pueden representar
  - Proposiciones, con un valor de verdad determinado en un contexto concreto. Las proposiciones se usan, por ejemplo, con el acto comunicativo `inform`.
  - Acciones que pueden ser ejecutadas. Estas acciones pueden ser simples o complejas usando para estas últimas operadores de secuenciación o alternancia. Por ejemplo, las acciones son el contenido de mensajes `request`.
  - Una expresión referencial de identificación (IRE), que identifica objetos en el dominio. Este es básicamente el operador referencial que tienen todos los lenguajes lógicos y que se utilizan, por ejemplo, en la performativa `inform-ref`.
- Ver recomendación