Escola Tècnica Superior d'Enginyeria
Universitat Rovira i Virgili

[DEIM]

Lecture 3: Agent Properties and Types

Multi-Agent Systems

Universitat Rovira i Virgili

MESIIA – Master's Degree in Computer Security Engineering and Artificial Intelligence
MAI – Master's Degree in Artificial Intelligence

Outline

1. Agent properties
2. Agent types
    1. Agent classification
    2. Description of some types of agents
    3. Agentification mechanisms

1. Agent properties

- The usual properties that agents may exhibit are:
  1. Flexibility
  2. Reactivity
  3. Proactiveness
  4. Social Ability
  5. Rationality
  6. Reasoning
  7. Learning
  8. Autonomy
  9. Temporal continuity
  10. Mobility

1. Flexibility

- An intelligent agent is a computer system capable of flexible action in some dynamic complex environment
- By flexible, we mean:
  - Adaptive
  - Proactive
  - Social

2. Reactivity

- A *reactive* system is one that maintains an ongoing interaction with its environment, and responds to changes that occur in it (in time for the response to be useful)

- If a program's environment is guaranteed to be fixed, the program can just execute blindly
- The real world is not like that: things change, information is incomplete. Most interesting environments are *dynamic*

Ways to achieve reactivity

[Recall last lecture]

- Reactive architectures
  - [Situation – Action] rules
  - Layered, behaviour-based architectures
- Deliberative architectures
  - Symbolic world model, long-term goals
  - Reasoning, planning
- Hybrid architectures
  - Reactive layer + deliberative layer [+ social layer]

Balancing Reactive and Goal-Oriented Behaviour

- We want our agents to be reactive, responding to changing conditions in an appropriate (timely) fashion
- We want our agents to systematically work towards long-term goals
- These two approaches are complementary
- Designing an agent that can balance the two remains an open research problem

[recall hybrid architectures]

3. Proactiveness

- Reacting to an environment is relatively easy

  (e.g., stimulus → response rules)
- But we generally want agents to do things for us, to act on our behalf
- Hence they must exhibit goal-directed behaviour
- Agents should be proactive

Aspects of Proactiveness

- Generating and attempting to achieve goals
- Behaviour not driven solely by events
- Taking the initiative when appropriate
- Executing actions/giving advice/making recommendations/making suggestions without an explicit user request
- Recognizing opportunities on the fly
  - Available resources
  - Chances of cooperation

Examples of Proactiveness

- Set of agents embedded in the home of an old or disabled person
- Detects the movement of the person around the house and the actions he/she performs
- Learns the usual daily patterns of behaviour
- Can detect abnormal situations, and proactively send warnings/alarms to family/health services
  - F.i., too much time in the same position, long time in the bathroom, whole day without going into the kitchen, …
- Demo. CONFIDENCE project

4. Social Ability

- The real world is a *multi*-agent environment: we cannot go around attempting to achieve goals without taking others into account
- Some goals can only be achieved with the cooperation of others
- Social ability in agents is the ability to interact with other agents (and possibly humans) via some kind of agent-communication language

Requirements for communication

- Agent communication language
  - FIPA-ACL
  - Message types
  - Message attributes
- Agent communication protocols
- Languages to represent the content of the messages between agents
- Shared ontologies
- World-wide standards

[Lecture 5: Agent communication]

High-level activities

- Communication is a first step towards sophisticated activities:
  - Coordination
    - How to divide a task between a group of agents
    - Distributed planning
  - Cooperation
    - Share intermediate results
    - Share resources
  - Negotiation [e-commerce]
    - Agree conditions in an economic transaction
    - Find the agent that can provide a service with the best conditions (matchmaking)

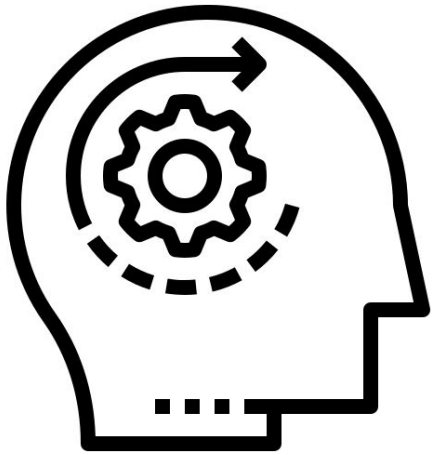
Second part of the course

Other aspects related to communication

- Security issues
  - Authentication
  - Encryption
- Trust
  - To what extent can we trust the other agents of the system?
  - Reputation models

5. Rationality

- An agent will act in order to achieve its goals
- It will not act in such a way as to prevent its goals from being achieved
  - At least insofar as its beliefs permit
- For instance, it will not apply deductive procedures (blind triggering of rules) without a purpose

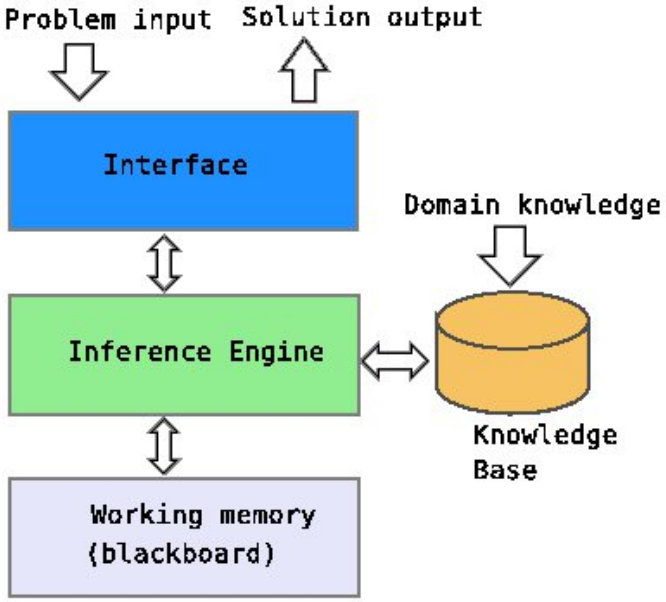


6. Reasoning capabilities

- Essential aspect for intelligent/rational behaviour
- Knowledge base with beliefs on the world
- Ability to infer and extrapolate based on current knowledge and experiences
- Capacity to make plans
- This is the characteristic that distinguishes an intelligent agent from a more “robotic” reactive-like agent

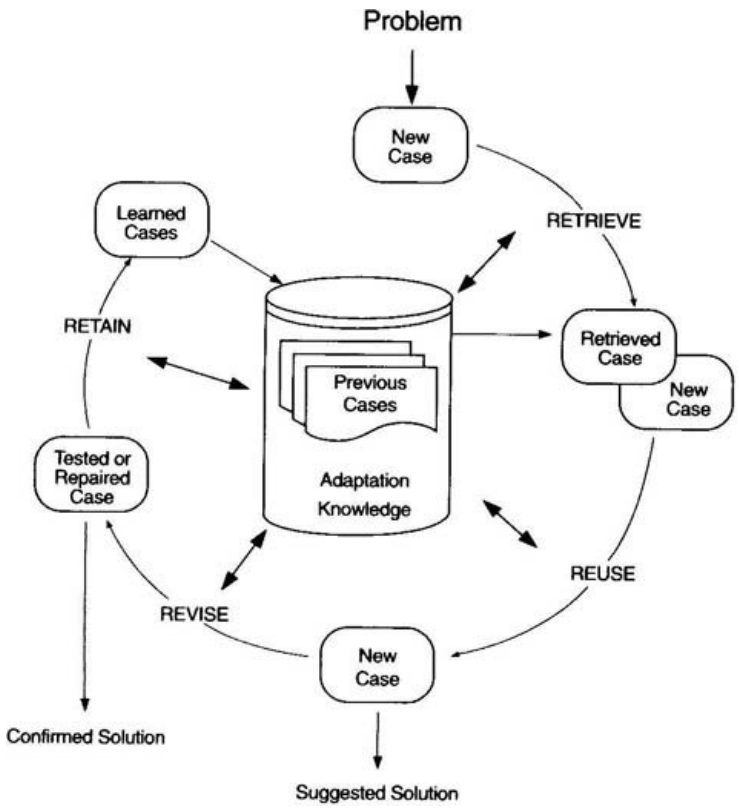
Kinds of reasoning in AI

- Knowledge-based systems / expert systems
  - Reasoning techniques specialised in the system's domain
  - Forward-chaining, backward-chaining, ML, DL, etc
- Rule-based systems
  - Knowledge is represented as a set of rules
  - Detect – Select – Apply execution cycle



Kinds of reasoning in AI

- Case-based reasoning
  - Using similarity to solved problems
- Approximate reasoning
  - Fuzzy logic, Bayesian networks, probabilities, etc [PAR]



7. Learning

- Basic component for an intelligent behaviour
- It can be considered as the [automatic] improvement of the performance of the agent over time
- Machine/Deep Learning: large area within AI

Learning for improving

- Make less mistakes
- Do not repeat computations performed in the past
- Find solutions more quickly
- Find better solutions
- Solve a wider range of problems
- Learn user preferences and adapt the behaviour accordingly

Advantages of learning systems

- Can adapt better to a dynamic environment, or to unknown situations
  - Without the need of an exhaustive set of rules defined at design time
- Can leverage previous positive/negative experiences to act more intelligently in the future

8. Autonomy

- Very important difference between agents and traditional programs
- Ability to pursue goals in an autonomous way, without direct continuous interaction/commands from the user
- Given a vague/imprecise goal, the agent must determine the best way to attain it

Autonomous decisions

- Given a certain goal …
  - Which actions should I perform?
  - How should I perform these actions?
  - Should I seek/request/(buy!) help/collaboration from other agents?

- Less work for the human user!!!

Autonomy requirements

- To have autonomy, it is necessary for an agent…
  - To have a control on its own actions
    - An agent cannot be obliged to do anything
  - To have a control on its internal state
    - The agent's state cannot be externally modified by another agent
  - To have the appropriate access to the resources and capabilities needed to perform its tasks
    - E.g., access to Internet, communication channels with other agents

Autonomy limitations

- Sometimes the user may restrict the autonomy of the agent
  - For instance, the agent could have autonomy to search in Internet for the best place to buy a given book …
  - … but the agent could not have the autonomy to actually *buy* the book, using the credit card details of the user

Autonomy issues

- Autonomy also raises complex issues:
  - Legal issues
    - Who is the responsible of the agent's actions?
  - Ethical issues
    - To what extent should decisions be delegated to computational agents?
- E.g., agents in medical decision support systems

9. Temporal continuity

- Agents are continuously running processes
  - Running active in the foreground or sleeping/passive in the background until a certain message arrives
- Not once-only computations or scripts that map a single input to a single output and then terminate

10. Mobility

- Mobile agents can be executing in a given computer and, at some point in time, move physically through a network (e.g., Internet) to another computer, and continue their execution there
- In most applications the idea is to go somewhere to perform a given task and then come back to the initial host with the obtained results

Example: Access to a DB

- Imagine there is a DB in Australia with thousands of images, and we need to select some images with specific properties
- We have to make some computations on the images to decide whether to select them or not

Option 1: remote request

- The agent in our computer makes hundreds of requests to the agent managing the DB in Australia
  - Continuous connection required
  - Heavy use of the bandwidth
  - All computations made on our computer

Option 2: local access

- Establish connection
- Send a specialised agent to the Australian computer holding the DB

  <connection can be dismissed here>
- The agent makes local accesses to the DB, analysing the images there

  <re-establish connection>
- Our agent comes back with the selected images

Problems of mobile agents

- Security
  - How can I accept mobile agents in my computer? (virus!!!)
- Privacy
  - Is it secure to send an agent with the details of my credit card, or with my personal preferences?
- Technical management
  - Each computer has to be able to “pack” an agent, send it to another machine, receive agents from other machines, “validate” them, and let them execute locally

Kinds of mobility

- Do not confuse
  - Mobile agents
    - Agents that can move from one computer to another
  - Agents running in mobile devices
    - Agents executing in portable devices such as PDAs, Tablet PCs, portable computers or mobile phones

11. Other properties

- Benevolence
  - An agent will always try to do what is asked of it
- Veracity
  - An agent will not knowingly communicate false information
- Character
  - Agents must seem honest, trustable, …
- Emotion
  - Agents must exhibit emotional states, such as happiness, sadness, frustration, …

Relationships between properties

- More learning => more flexibility
- More learning => more autonomy
- More reasoning => more rationality
- More reasoning => more proactivity
- Less autonomy => less proactivity

Conclusions on properties

- It is almost impossible for an agent to have all those properties!!!
- Most basic properties:
  - Autonomy
  - Reactiveness
  - Reasoning and learning
  - Communication

2. Agent types

1. Agent classification
2. Description of some types of agents
3. Agentification mechanisms

How to classify agents? By properties

- There may be different classifications of agents, depending on the employed criteria
- An initial classification was made based on the properties that are emphasized in a particular agent:
  - Reasoning
  - Mobility
  - Flexibility
  - …

  but there are multiple combinations of properties

How to classify agents? By “purpose”

- Collaborative agents
- Interface agents
- Information agents
- Facilitator agents
- Translators

Agent types: Collaborative Agents

- Emphasize *communication* and *cooperation* with other agents
- Typically operate in open multi-agent environments (MAS)
- *Negotiate* with to reach mutually acceptable agreements during cooperative problem solving
- Collaborative agents are usually *deliberative* agents (BDI model), with some *reasoning* capabilities

Agent types: Collaborative Agents - Rationale

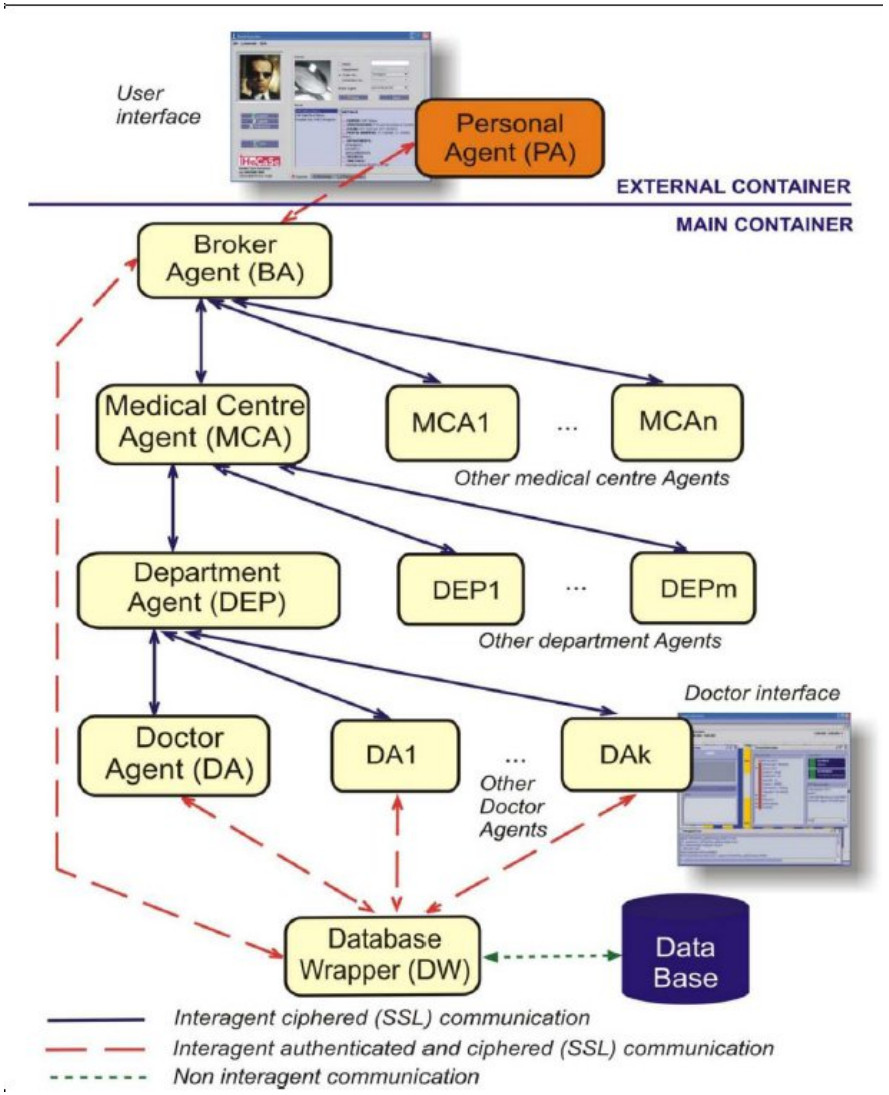
- To solve problems that are too large for a single centralised agent
- To create a system that functions beyond the capabilities of any of its members
- To allow for the interconnection and inter-operation of existing legacy systems
- To provide solutions to inherently distributed problems

Agent types: Collaborative Agents - Applications

- Provide solutions to physically distributed problems
  - Air-traffic control, management of a team of robots
- Provide solutions to problems with distributed data sources
  - Different offices of a multi-national business
- Provide solutions that need distributed expertise
  - Health care provision (family doctors, nurses, specialists, laboratory analysis, ...)

Example: HeCaSe Architecture

HeCaSe: An Agent-Based System to Provide Personalised Medical Services, David Isern, David Sánchez, Antonio Moreno, Aïda Valls (URV, 2003-2009)



Example: HeCaSe Architecture

HeCaSe: An Agent-Based System to Provide Personalised Medical Services, David Isern, David Sánchez, Antonio Moreno, Aïda Valls (URV, 2003-2009)

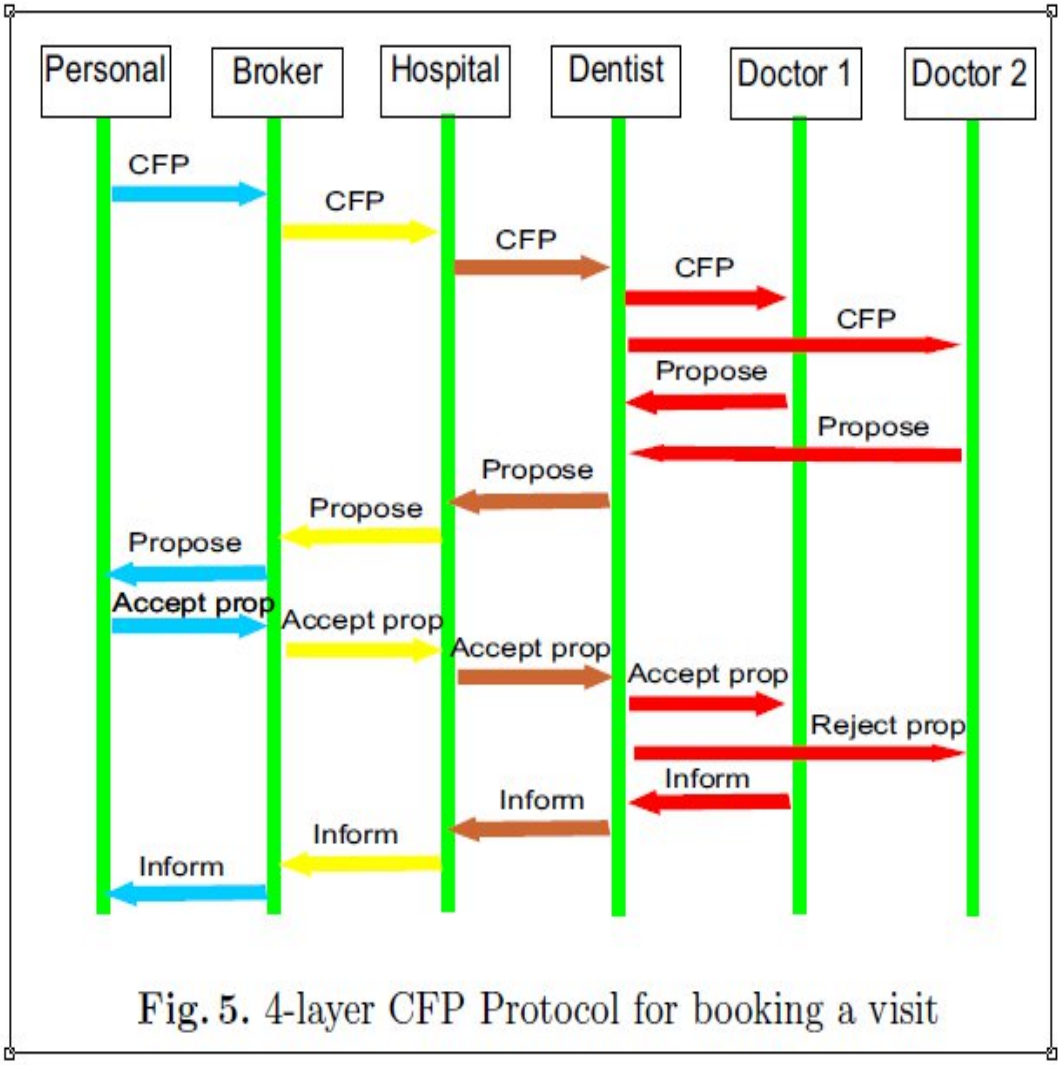
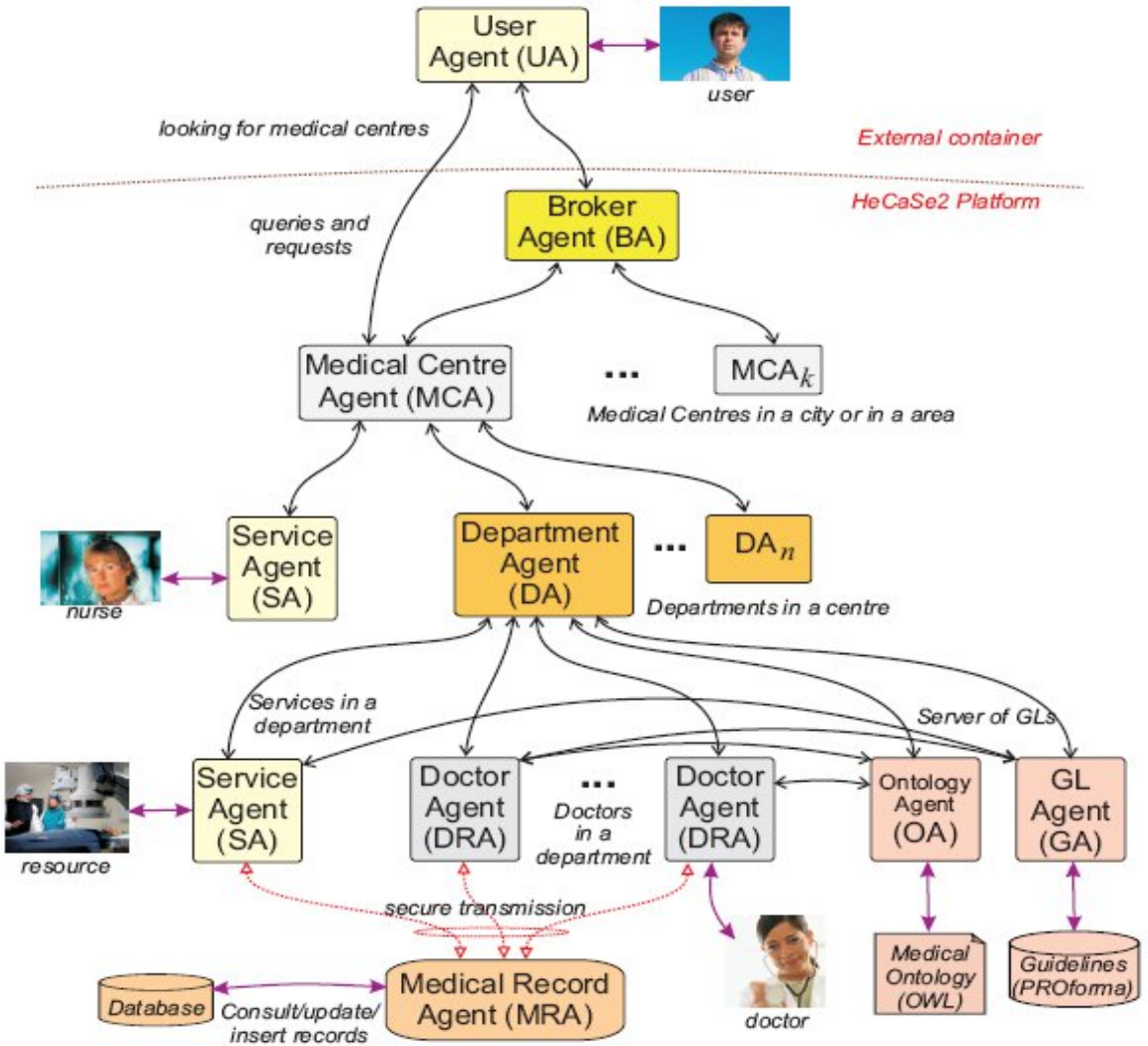


Fig. 5. 4-layer CFP Protocol for booking a visit

Example: HeCaSe2 Architecture

HeCaSe2: A Multi-Agent System that Automates the Application of Clinical Guidelines, David Isern and Antonio Moreno (URV, 2010)



Agent types: Interface (or User) Agents

- Emphasise *autonomy* and *learning* in order to perform tasks for their owners
- Support and provide *proactive* assistance to a human that is using a particular application or solving a certain problem
  - Anticipate user needs
  - Make suggestions
  - Provide advice

  … without explicit user requests

Agent types: Interface Agents

- Limited cooperation with other agents
- Limited reasoning and planning capabilities
- Interface agent = personal assistant

Agent types: Interface Agents – Personal Assistant

- Initially, a personal assistant is not very familiar with the habits and preferences of his employer
  - It may not be very helpful
  - It may even give extra work !
- With every experience, the assistant *learns* by
  - Watching how the employer performs tasks
  - Receiving instructions from the employer
  - Learning from other more experienced assistants
- Gradually, more tasks that were initially directly performed by the employer can be taken care of by the assistant

Agent types: Interface Agents – Applications

- Less work for the end user and for the application developer
  - Automation of routine activities
  - Automation of tasks that would take a long time to a human user
- Examples:
  - Mail management
  - Scheduling meetings
  - Internet browsing
  - News filtering agent

Example: Mail management assistant

- Delete uninteresting or potentially harmful messages
- Prioritize the messages according to their relevance
- Sort the incoming messages in the appropriate folders
- Warn the user when a very important message arrives
- Forward a message to another user

Example: Meeting scheduler

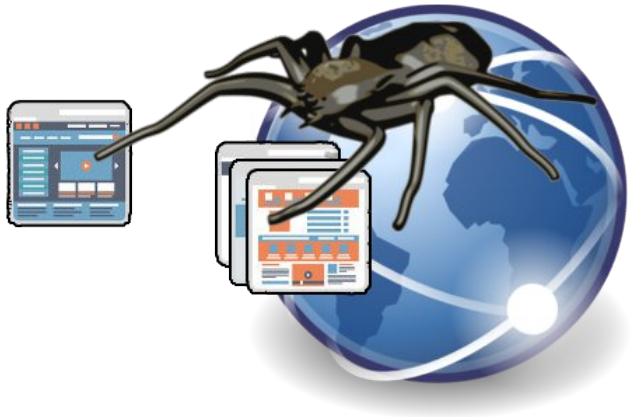
- Learn user preferences on different kinds of meetings
- Make a meeting proposal
- Accept a meeting proposal
- Reject a meeting proposal
- Reschedule a previously agreed meeting
- Negotiate a meeting time with other agents

Agent types: Interface Agents – Problems

- Slow learning curve
  - Agents require many examples before they can make accurate predictions (especially if they cannot be directly trained)
  - No useful assistance during the learning process
- Learning from scratch
  - Each agent has to learn on its own, even if there is a bunch of agents dealing with a team of people with similar interests
- Trust: How can we guarantee the user feels comfortable delegating tasks to an agent?

Agent types: Information Agents

- Software agents that manage the access to multiple, heterogeneous and geographically distributed information sources.
- Information agents = *Internet agents*
- Main task: proactive acquisition, mediation and maintenance of relevant information for a user/other agents



Agent types: Information Agents - Interest

- Commercial benefits
  - *Proactive*, *dynamic*, *adaptive* and *cooperative* WWW browser-embedded information manager
- User benefits
  - Time and effort to access and analyse data
  - Improve productivity (more time, better data)

Agent types: Information Agents - Tasks

- Information acquisition and management
  - Provide transparent access to different information sources
  - Retrieve, extract, analyse, summarize and filter data
  - Monitor information sources
  - Update relevant information on behalf of the user
- Examples: DBs, web pages, purchase of information from providers on electronic marketplaces, ...

Agent types: Information Agents - Tasks

- Information synthesis
  - Fuse, merge heterogeneous data
    - Handle conflicts, contradictions, repetitions
  - Provide unified, multi-dimensional views on relevant information to the user
    - Not just a mere list of data from different sources
- Personalised presentation of information
  - Learn automatically user preferences
  - Adapt dynamically to changes in user preferences

Example: Biological data

- Biological Data Search and Integration (2019)
  - *A major problem for the integration of biological data is the structure and format of the data. They are not always standardized and data access is not centralized, making it extremely difficult and time-consuming to search for results in all databases.*

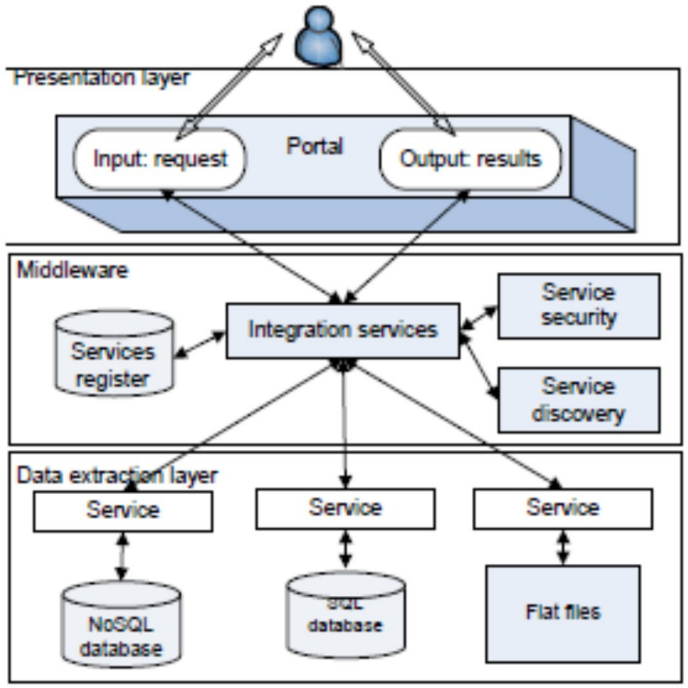


Fig. 1 Conceptual architecture for retrieval and integration of biological data based on SOA

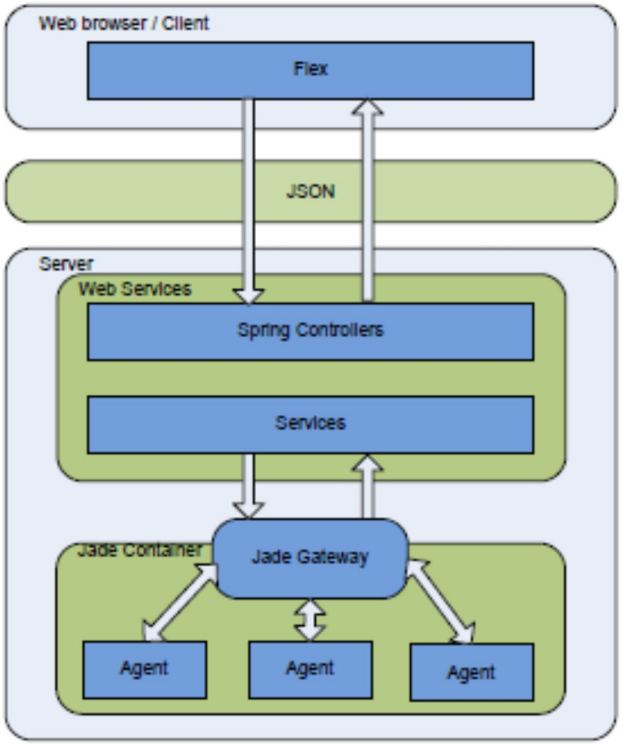


Fig. 3 Multilayer architecture of system for biological data searching

Example: Information Foraging

- A Collaborative Multi-Agent Approach to Web Information Foraging (2017)
  - *Exploratory Search, and which aims at discovering paths leading to relevant information on the Web is Information Foraging. Aim: discover in an automatic way the surfing paths that could be useful to users in order to help them to learn, and augment their knowledge on a specific topic.*

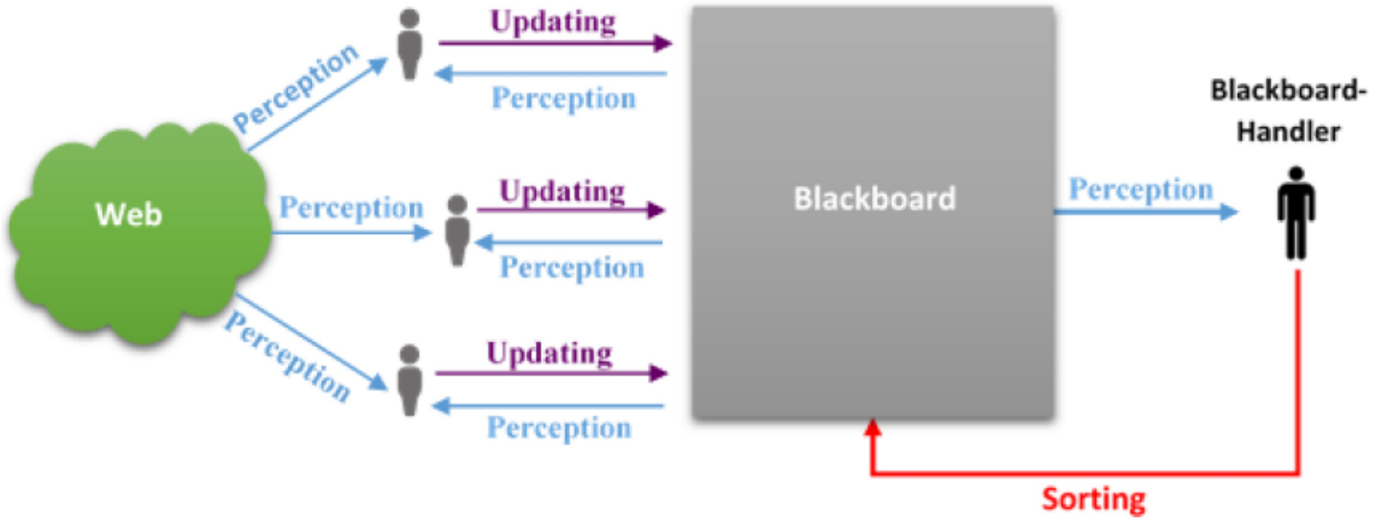


Fig. 1. Architecture of the Web Information Foraging Multi-Agent System.

Agent types: Information Agents

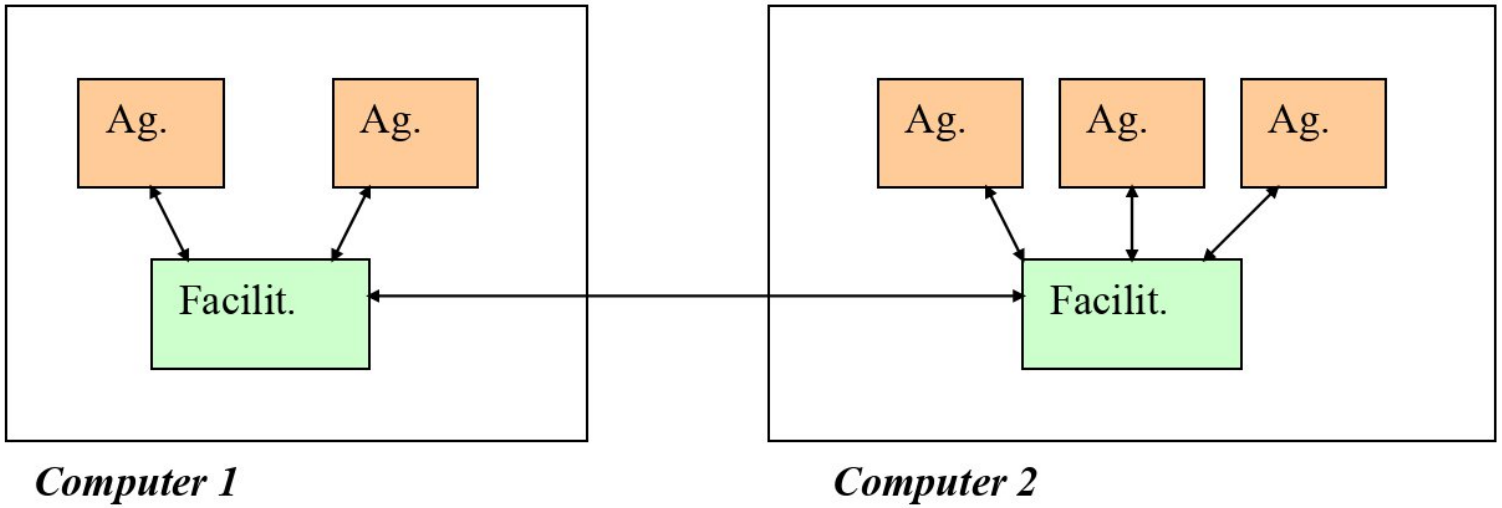
- Information agents must help in the task of information retrieval
- Advanced models use a conversational paradigm
  - User makes a question
  - System returns the subset of documents that are considered relevant to the query
  - User evaluates the returned information items
  - User refines the initial question

Agent types: Facilitator Agents

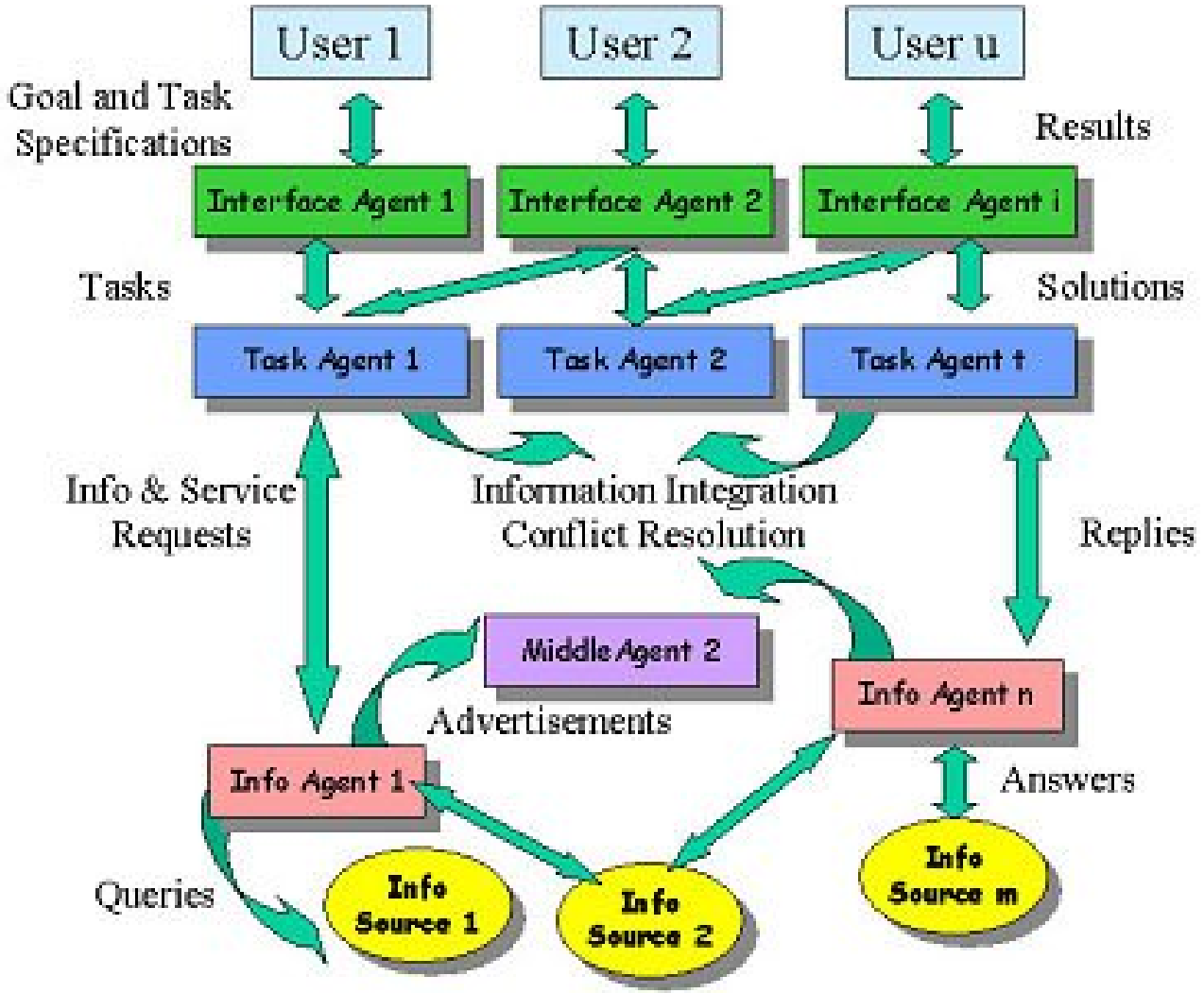
- Flat system
  - Each agent can talk directly to any other agent
- Federated system
  - There are special agents (facilitators), that manage
    - The connection of the system with the users, and
    - The communication between the agents

Agent types: Facilitator Agents – Federated systems

- Agents do not communicate directly, but through facilitators
- Facilitators help agents to find other agents in the system that provide some services
- Without having to ask all other agents



Example: RETSINA



Example: RETSINA - Agents

- Interface agents: input/output, learn from user actions
- Task agents: encapsulate task-specific knowledge, used to perform/request services to/from other agents/humans
  - They can coordinate their activities
- Middle agents: infrastructure service (e.g., they know the services provided by other agents)
- Information agents: monitor and access one or more information sources

Agent types: Facilitator Agents

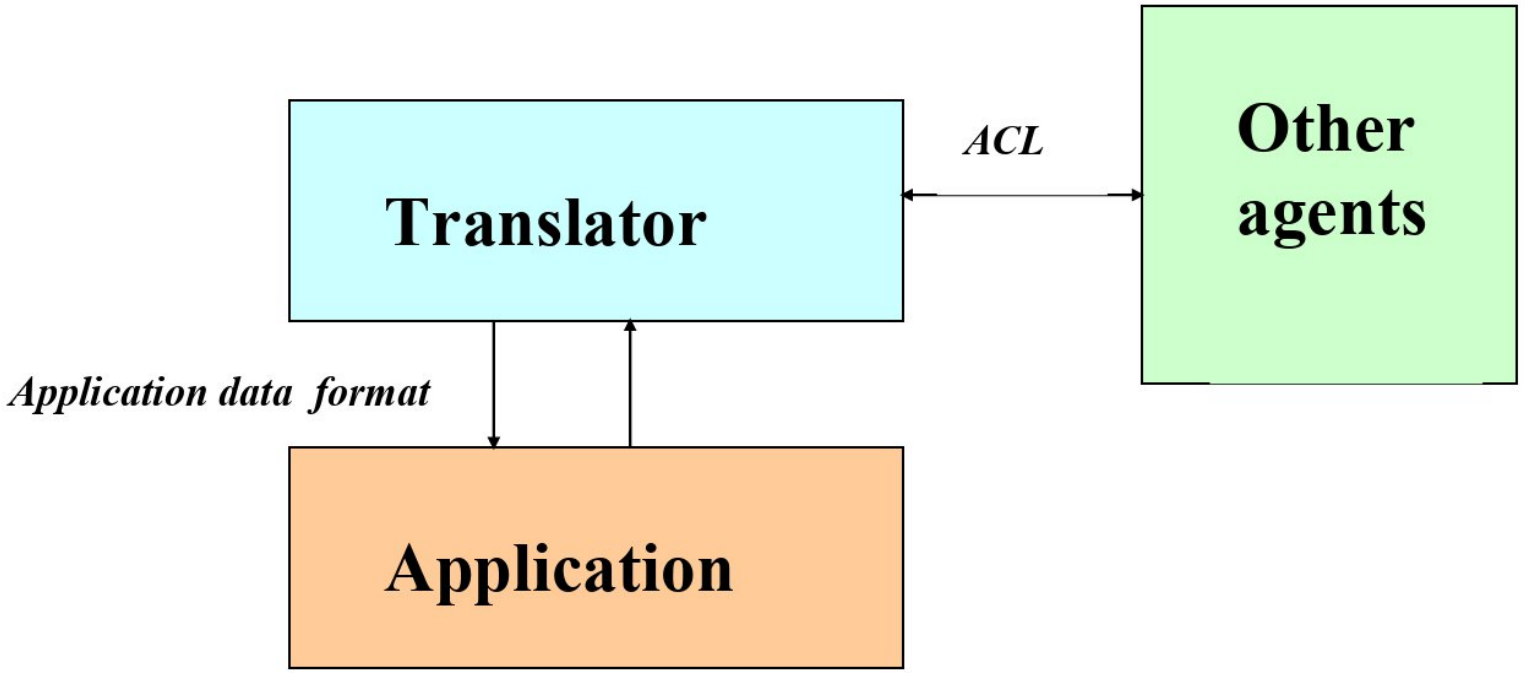
- Broker agents: in addition to facilitate the connection with users or providers, they can:
  - Store data about the resources available (cache)
  - Select which agents should be contacted
  - Collect, merge and filter answers
  - Coordinate with other brokers
  - …

Agentification mechanisms

- We can re-use an existing application (not necessarily based on agent technology)
  - *Legacy* systems
  - No need to re-program all the applications to integrate them within a MAS
  - The union can give an added value
- Agentification is the way to transform a standard application into an "*agent*" that can be integrated and participate in a multi-agent system of collaborative agents

Agentification mechanisms: Translator

- "Bridge" between the application and the other agents
- Takes the messages of other agents, and translates them to the program communication protocol (and

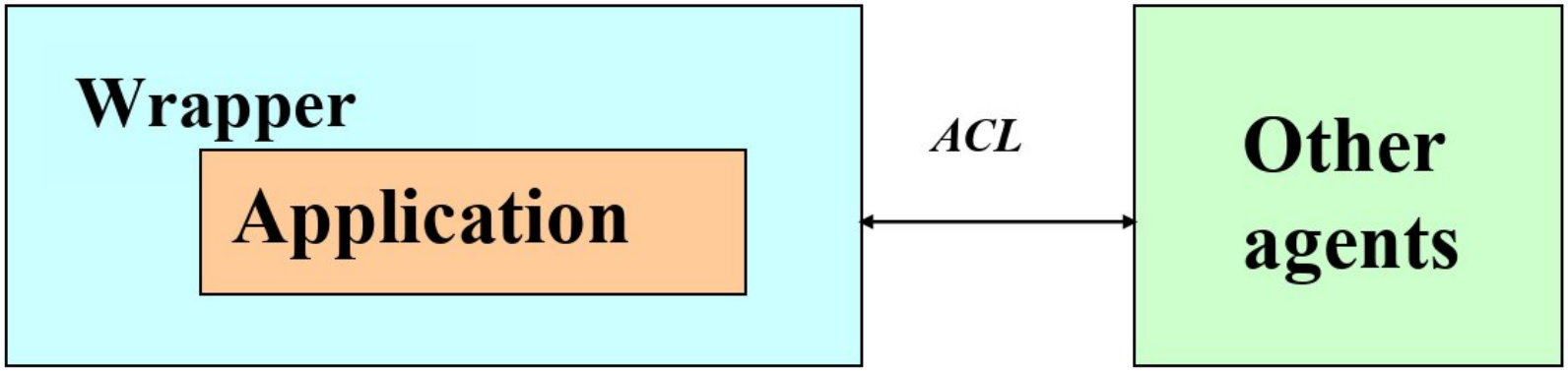


Agentification mechanisms: Translator - Advantages

- The code of the application does not have to be modified
- We only need to know the inputs/outputs of the application
- The same translator can be used to agentify different applications/resources
  - Access to an Oracle database

Agentification mechanisms: Wrapper

- Add code to the application so that it can communicate with the agents of the MAS
- Implies a direct manipulation of the internal data structures of the application

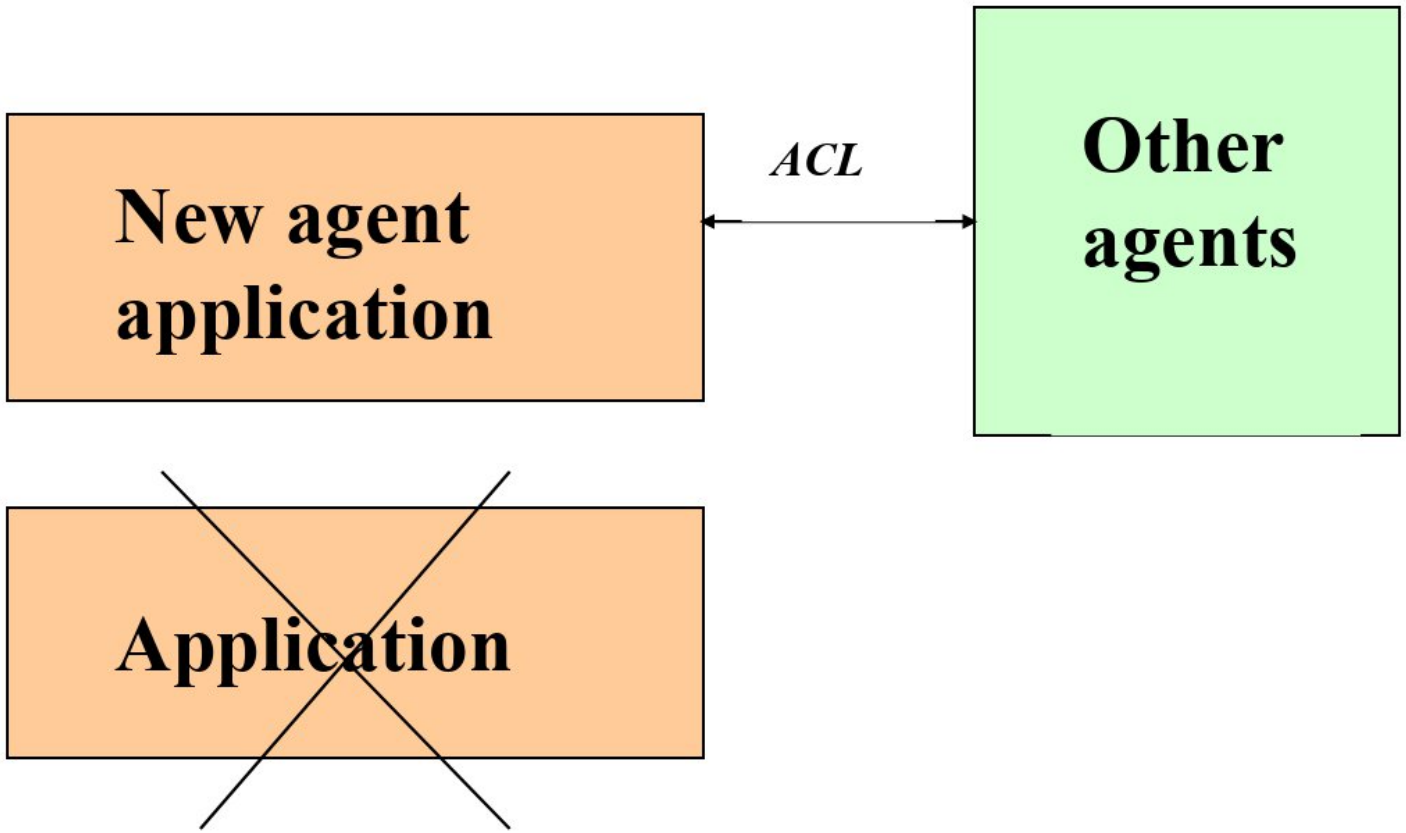


Agentification mechanisms: Wrappers - Comments

- Positive aspects
  - More computationally efficient than translators
- Negative aspects
  - We need access to the application source code
  - It may not be easy to understand/modify the code of a complex application
  - A wrapper is not reusable as a translator
    - One wrapper for each program to be agentified

Agentification mechanisms: Re-code the application

- Re-write the original program as an agent



Agentification mechanisms: Re-code the application - Comments

- Negative aspects
  - Much more work than the previous options
  - It does not seem really an "agentifying" mechanism
- Positive aspect
  - The resulting agent could be designed to work in a much more efficient way, without external/internal translations between formats

Agent types: Final comments

- It is usual to have different types of agents working together in a multi-agent system
- Not all types are needed in all applications

Proposed materials

- M. Wooldridge: An introduction to Multi-Agent Systems – beginning chapter 2, section 10.3
- [Video] Agent properties – M. Wooldridge

- Papers with examples of different types of agents
  - RETSINA and HeCaSe show collaborative agents, together with simple personal agents
  - Two papers show different uses of information agents