Java Agent Development Framework

what it is and what it is next

http://sharon.cselt.it/projects/jade/

Fabio Bellifemine

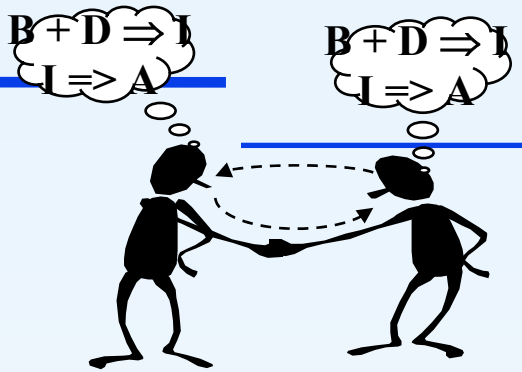
Telecom Italia Lab - Torino (Italy)

ETAPS 2001, 7th Apr. 2001

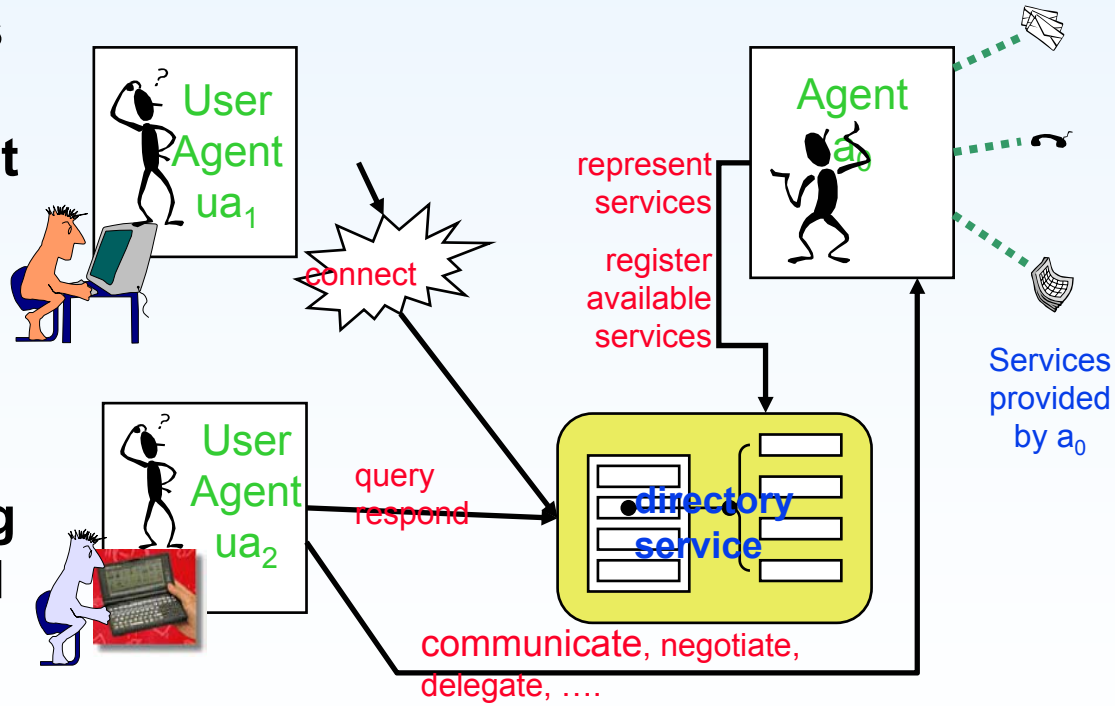
Overview

- **No standard, no agents**
  - FIPA – Foundation for Intelligent Physical Agents
- **No software tools, no agents**
  - JADE
    - what it is
      - *main features, architectural choices, sub-systems*
    - what it is next
- **Standards and Software tools, agents?**
  - Some conclusions

No standards, no agents



- **Maximum of potential is expressed by high level interaction (e.g. contract net, auction)**
  - agents from several designers, several vendors, several organizations
- **Standard is the enabling factor for openness and heterogeneity**



Shift from making better components
to making better the way the components cooperate

FIPA

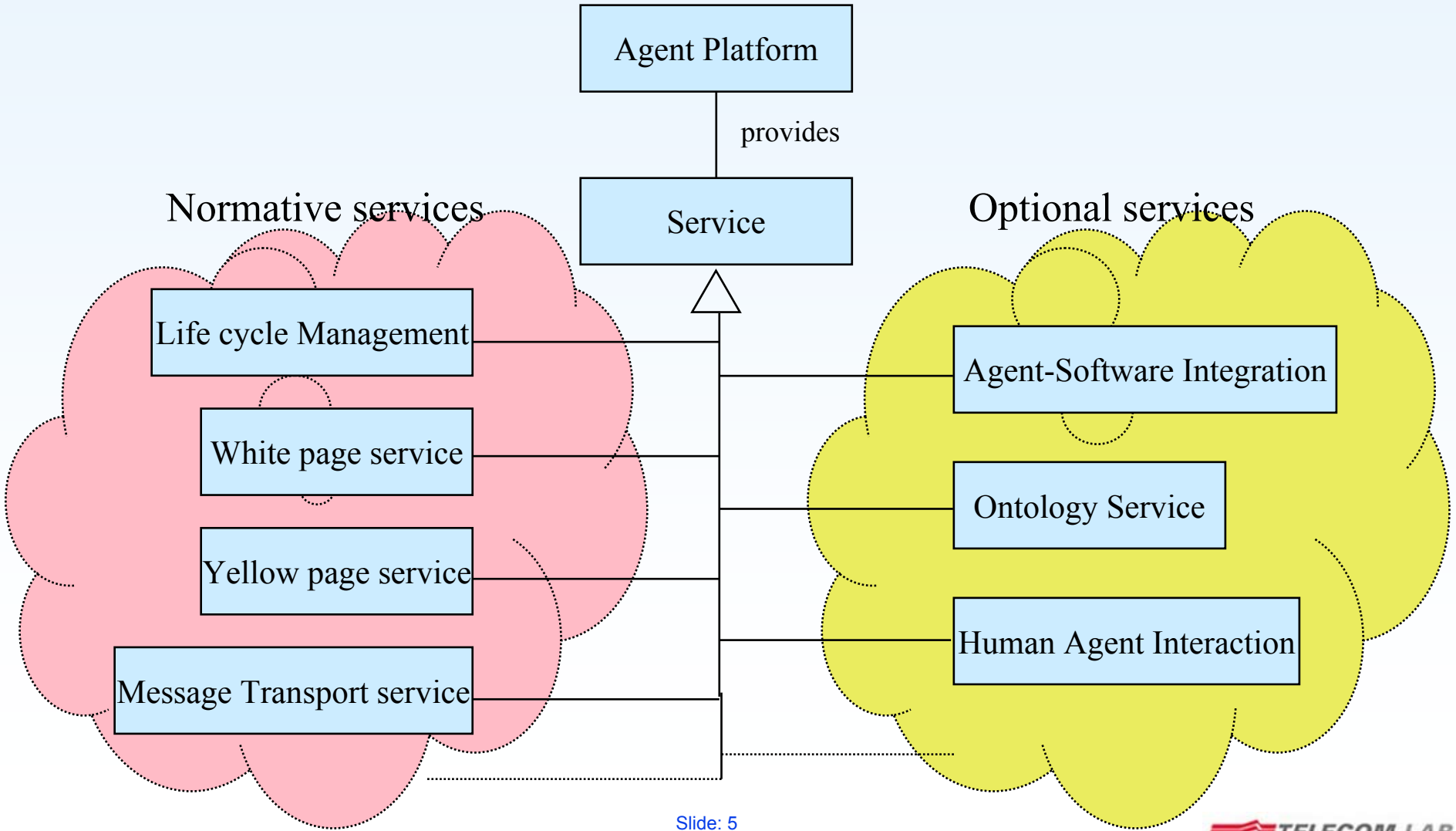
Foundation for Intelligent Physical Agents

http://www.fipa.org/

- is a non-profit association of companies, founded in 1996
- its purpose is to promote the success of emerging agent-based applications, services, and equipments
  - to facilitate the end-to-end interworking
    - *for heterogeneous and interacting agents and agent-based systems*
    - *developed by different companies and organisations*
- the goal is pursued by producing specifications
  - Agent Management and platform services
  - Agent Communication Model and Language
  - set of common Interaction Protocols
- today FIPA counts about 65 member companies
- the FIPA's core message
  - through a combination of speech acts, predicate logic and public ontologies, standard ways of interpreting communication between agents can be achieved that respect the intended meaning of the communication

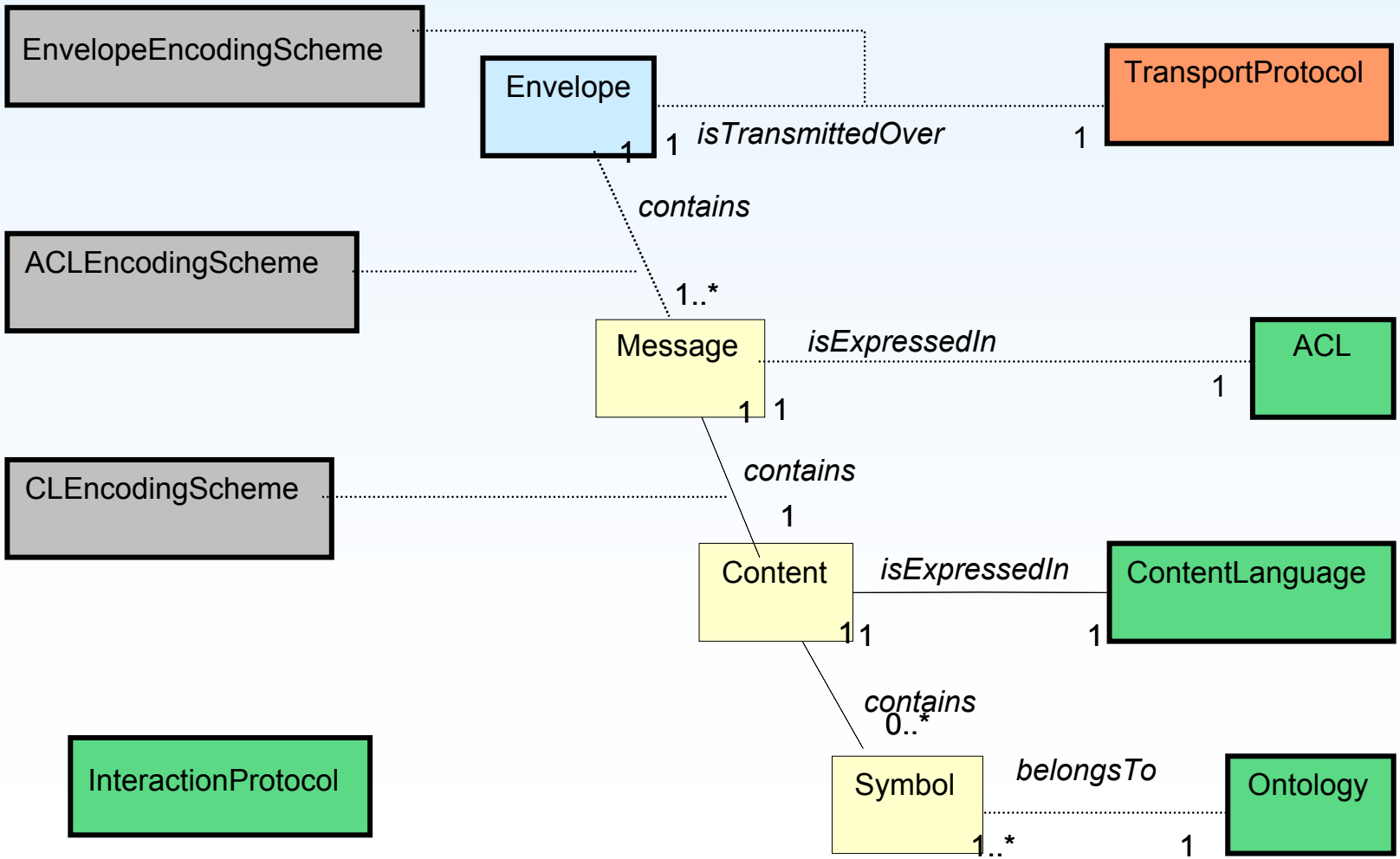
FIPA

Conceptual model of an Agent Platform



FIPA

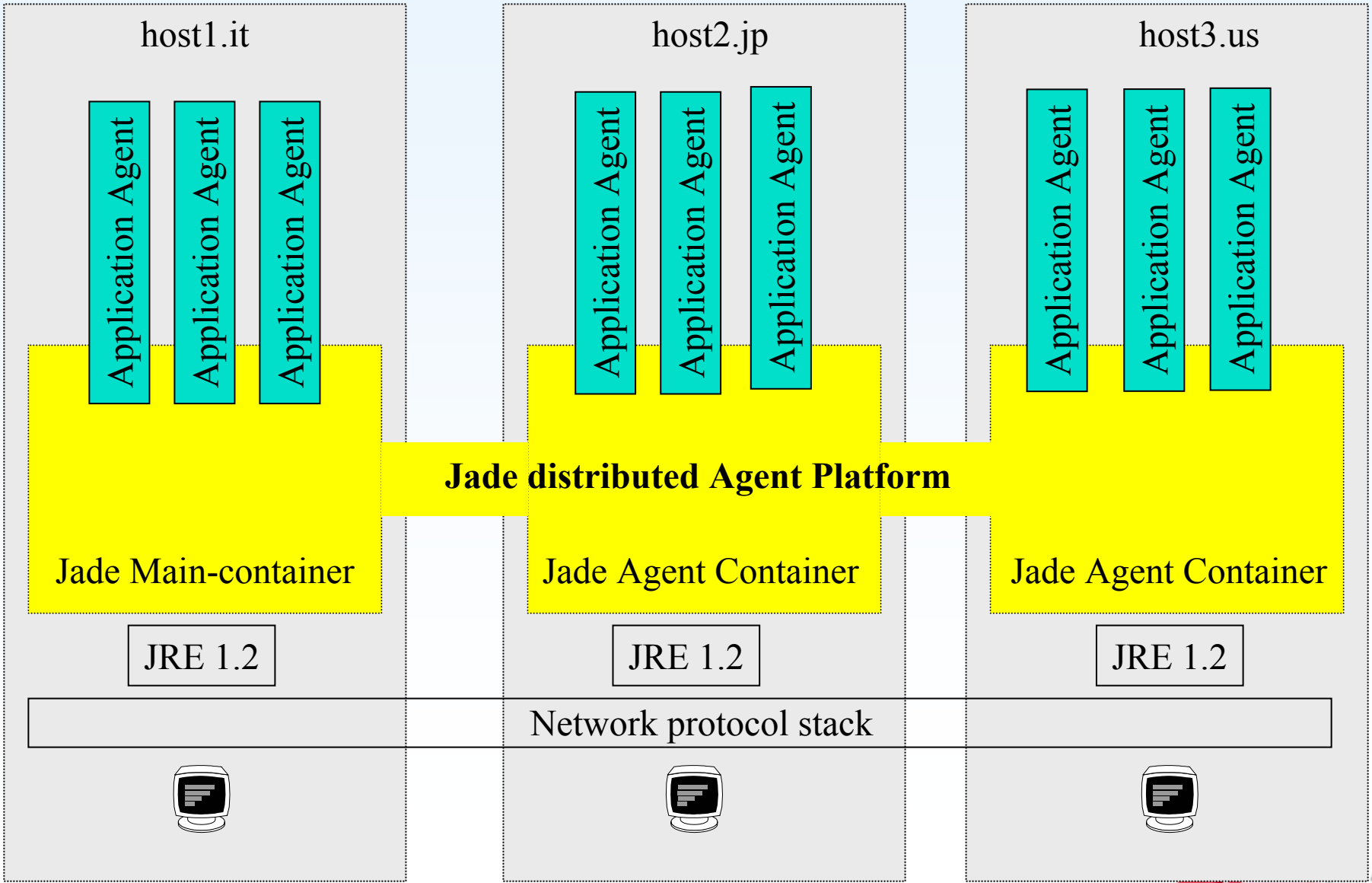
conceptual model of agent communication



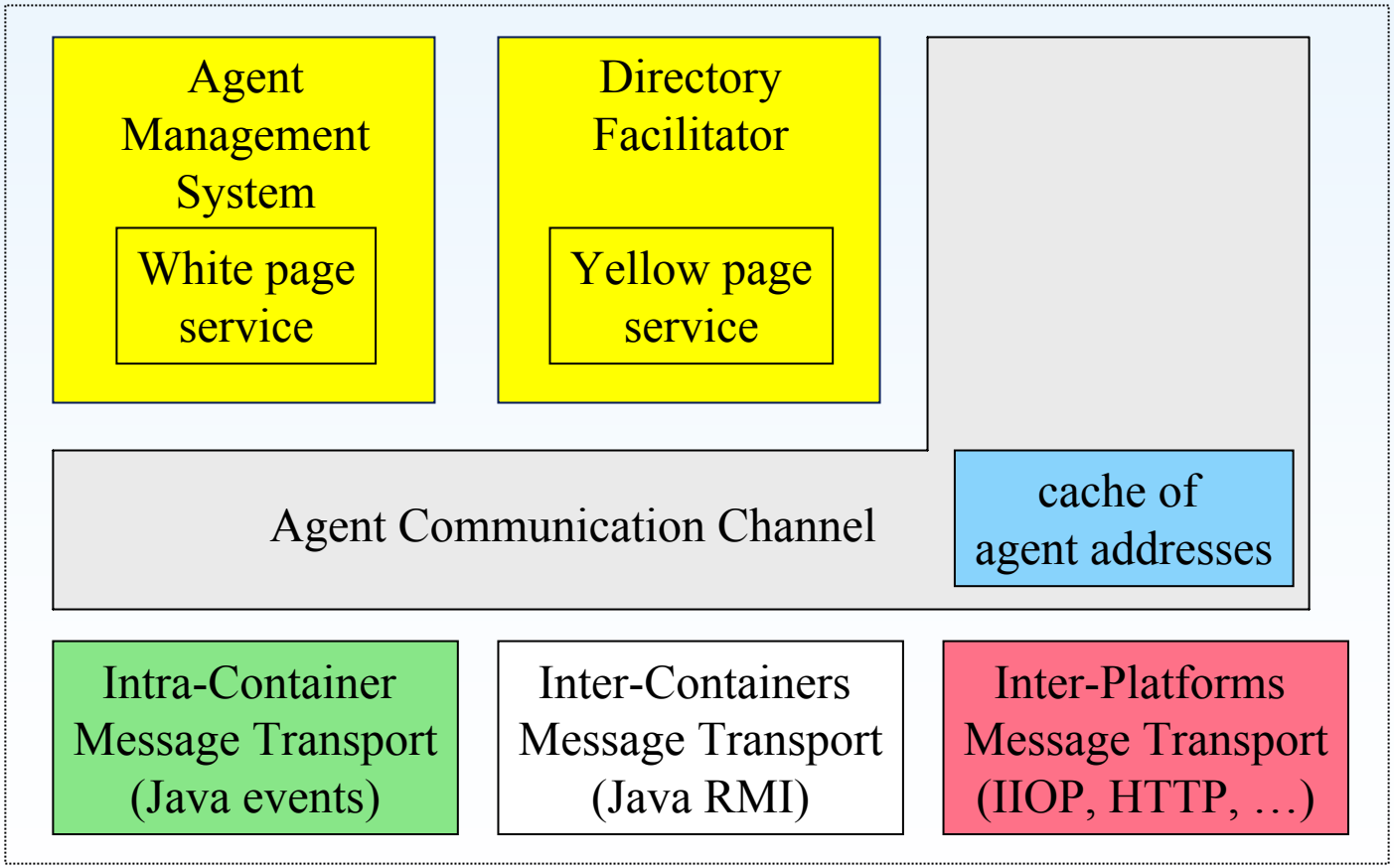
No software tools, no agents - JADE

- **is the middle-ware for MAS**
  - target users: agent programmers for MAS
  - agent services
    - life-cycle, white-page, yellow-page, message transport, message encoding
  - tools to support debugging phase
    - remote monitoring agent, dummy agent, sniffer agent, introspector agent
  - designed to support scalability
    - from debugging to deployment
    - from small scale to large scale
  - claims to comply with the FIPA specifications
- **fully implemented in Java**
- **distributed in Open Source under LPGL**

Distributed architecture of a JADE platform



Internal architecture of the JADE run-time



Note: The internal architecture of a JADE container is similar, but it does not contain the AMS, and the DF.
Within the scope of the LEAP project, dependency on Java RMI has been removed

JADE - Message Transport Service

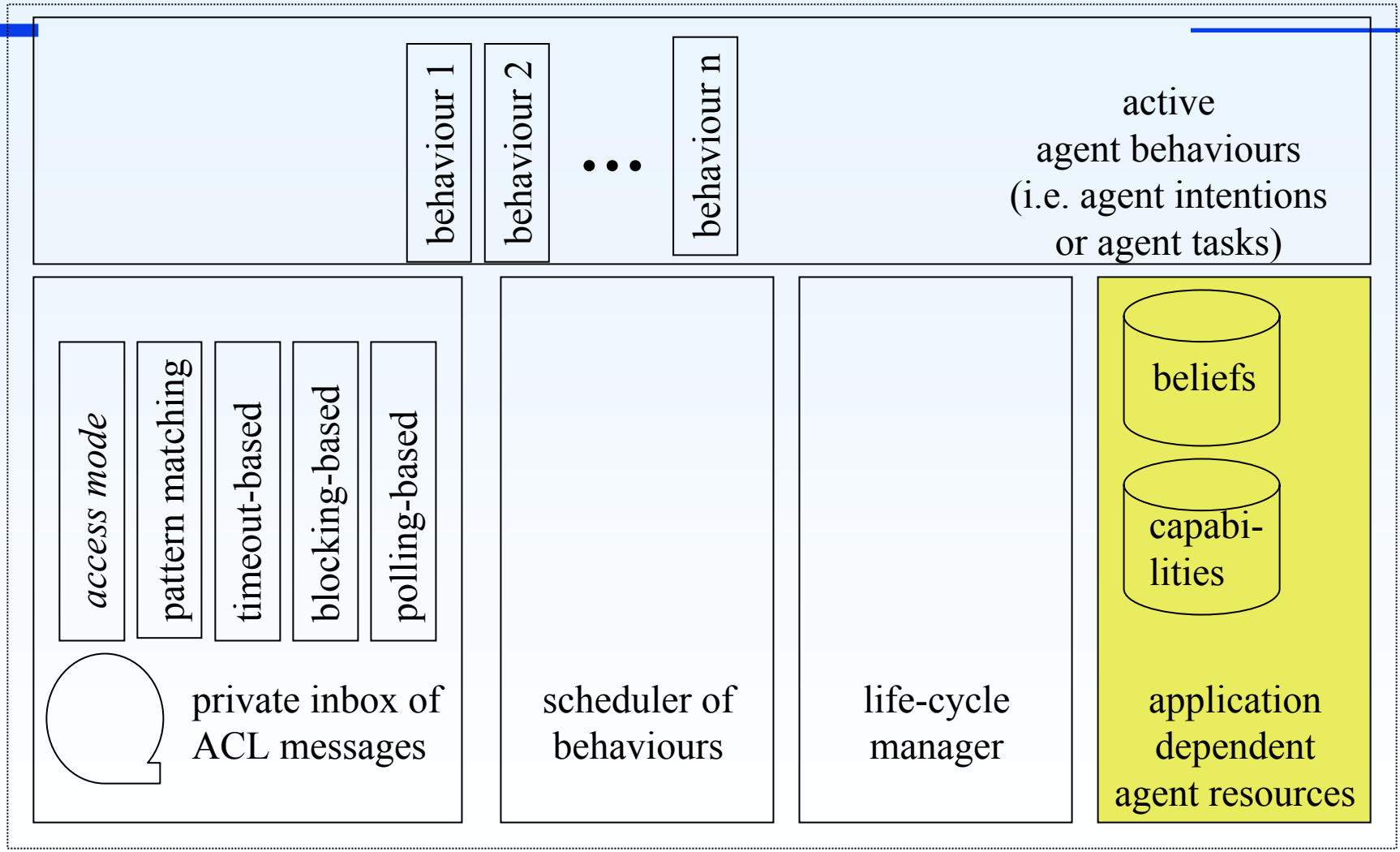
- **controls the agent's private queue of ACL messages**
- **designed as a chameleon**
  - the transport mechanism is selected according to the situation
    - to achieve the lowest cost for message passing
    - the overheads depend on the receiver's location and the cache status
- **distributed Agent Communication Channel**
  - the main container is not a bottle-neck, thanks to the distributed caches
  - Message Transport Protocols (MTP) can be activated/deactivated at run-time on any container via the GUI
    - IIOP based on the ORB implementation of Sun
    - IIOP based on the ORBacus implementation
      - *allows to make persistent the object reference*
      - *allows a more friendly URL-format corbaloc:iiop:hostname:port/name*
    - HTTP MTP provided by EPFL under LGPL
  - multiple ACL encodings have been implemented
    - String-based, XML-based (EPFL), bit-efficient (Sonera)

Agent Execution Model

implications on the JADE implementation

- **agent is autonomous**
  - it completely controls its thread of execution
    - has a private proxy of the life-cycle manager
  - decides itself when to read messages and which messages to read
    - the transport mechanism fills a private queue but it does not call the agent code (no automatic callback)
- **agent needs concurrency**
  - can engage multiple simultaneous conversations
  - can execute several concurrent tasks
    - Java multi-thread or/and
    - JADE behaviours with cooperative scheduling
      - *one thread-per-agent rather than one thread-per-task/conversation.*
- **Programming Model**
  - A JADE agent is mapped onto an user defined Java class, that must subclass `Agent` class in `jade.core` package.
  - Agent tasks are mapped onto user defined subclasses of `Behaviour` class in `jade.core.behaviours` package.

Int. architect. of a generic JADE agent



The JADE framework includes a library of interaction protocols and generic agent behaviours, that must be customized for the specific application needs in order to create the agent capabilities

JADE library of interaction protocols and of generic agent behaviours

JADE – Agent Communication Model

- **Agents send/receive Java objects, that represent ACLMessages, within the scope of interaction protocols**
  - JADE hides all message coding (encoding/parsing)
    - Envelope level
      - *String-based, XML-based*
    - Agent Communication Language level
      - *String-based, XML-based, bit-efficient*
    - Content Language level
      - *FIPA SL-0 + API to register user-defined content languages*
      - *support for Base64-encoded direct Java object serialization*
    - Ontology level
      - *FIPA-Agent Management; JADE Agent Management*
      - *API to register user-defined content languages*
    - the framework can be extended by users
      - *all levels provide APIs to implement/register new codecs*
      - *work is in progress to improve CL and ontology level extendibility*
  - JADE provides a library of common interaction protocols
    - users just need to implement the handle methods
    - users can compose agent tasks like super-states of FSM

JADE – Agent mobility

- **JADE supports intra-platform mobility and cloning**
  - A platform can be distributed across multiple hosts
    - each host is an agent container
  - Agents can migrate between containers
  - Agents can clone across containers
  - Self-initiated
    - doMove(Location) / doClone(Location, String)
    - before/afterMove() before/afterClone()
  - Requested to the platform (via the AMS)
    - Fipa-request interaction protocol
    - jade.domain.MobilityOntology defines all the concepts and actions needed to support agent mobility and cloning

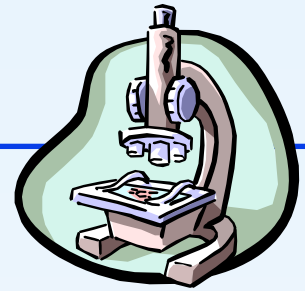
JADE - Some graphical tools to support development

- **RMA (Remote Monitoring Agent)**
  - to browse the white-page service
  - to control the agent life-cycle (e.g. remote creation, agent migration, …)
  - to activate/deactivate MTPs on containers
  - to browse white-page services of remote agent platforms
- **DF GUI**
  - to browse the yellow-page service
  - to make DF federations and browse remote DF's
- **DummyAgent**
  - send/receive store/save ACLMessages
- **Sniffer Agent**
  - to sniff, debug, save to file, multi-agent conversations
- **Introspector Agent**
  - to debug an agent: queue of sent/received messages, queue of behaviours

Scalability in JADE

- **Configuration of a platform**
  - from one MAS on a single host
    - single-host platform
  - to one agent on a single host
    - agent platform on a cluster of hosts
  - configuration can be changed at run-time
    - hot restarting is possible thanks to the local caches
      - *agent is referred by name => no need to get new reference*
- **Is the main-container a bottle-neck?**
  - the Agent Communication Channel is distributed
  - the main container is involved only when strictly necessary

How much of FIPA is hidden by JADE to the programmer?



- **no need to implement the Agent Platform**
  - AMS, DF, and ACC automatically launched at start-up
- **no need to implement agent-management ontology and functionalities**
  - an agent is registered with the AP by the Java constructor itself
    - it is given a name and an address
  - the Agent class provides a simplified interface to access the services of the DF (registration, searching, …)
- **no need to implement Message Transport and Parsing**
  - automatically (and possibly efficiently) done by the framework when sending/receiving messages
- **no need to implement Interaction Protocols**
  - they must only be extended via handle methods

JADE – what is next

- **JADE 2.2 (the taste of the community's power)**
  - HTTP MTP and XML-based ACLCodec(EPFL's contribution)
  - bit-efficient ACL codec (Sonera's contribution)
  - improvements to the SnifferAgent GUI (HP Palo Alto contribution)
  - GUI support for remote platforms, *introspector agent*, FSM Behaviour
- **JADE 2.3 (expected in summer)**
  - improving integration with Web technologies: applets, servlets, JSP, XML
  - improving support for persistence and restartability
  - improving support for interaction protocols
- **soon more …**
  - integrating the LEAP core (JADE/LEAP run-time for mobile terminals)
    - No API modifications. JADE agents will run also on mobile devices
  - security and multi-users support
  - improving support for JESS users
  - adding plans, i.e. JADE behaviours as production rule systems
  - … more contributions of the user community

The JADE license

open source does not just mean access to the source code (O'Reilly)

- **JADE is OSS since Feb. 2000 (version 1.3) under LGPL**
  - keep all contributors to the same level relative to each other
  - Assure right to
    - make and distribute copies of JADE
    - have access to the software's source code
    - make improvements to the program
    - incorporate JADE into a proprietary program
    - **continue the JADE experience even if we stopped it !!**
      - ***which will not happen, because we will not stop JADE so easily***
  - Mandate duty to
    - not keep modifications private
    - not change the license of JADE and its modifications

Some conclusions (1/2)

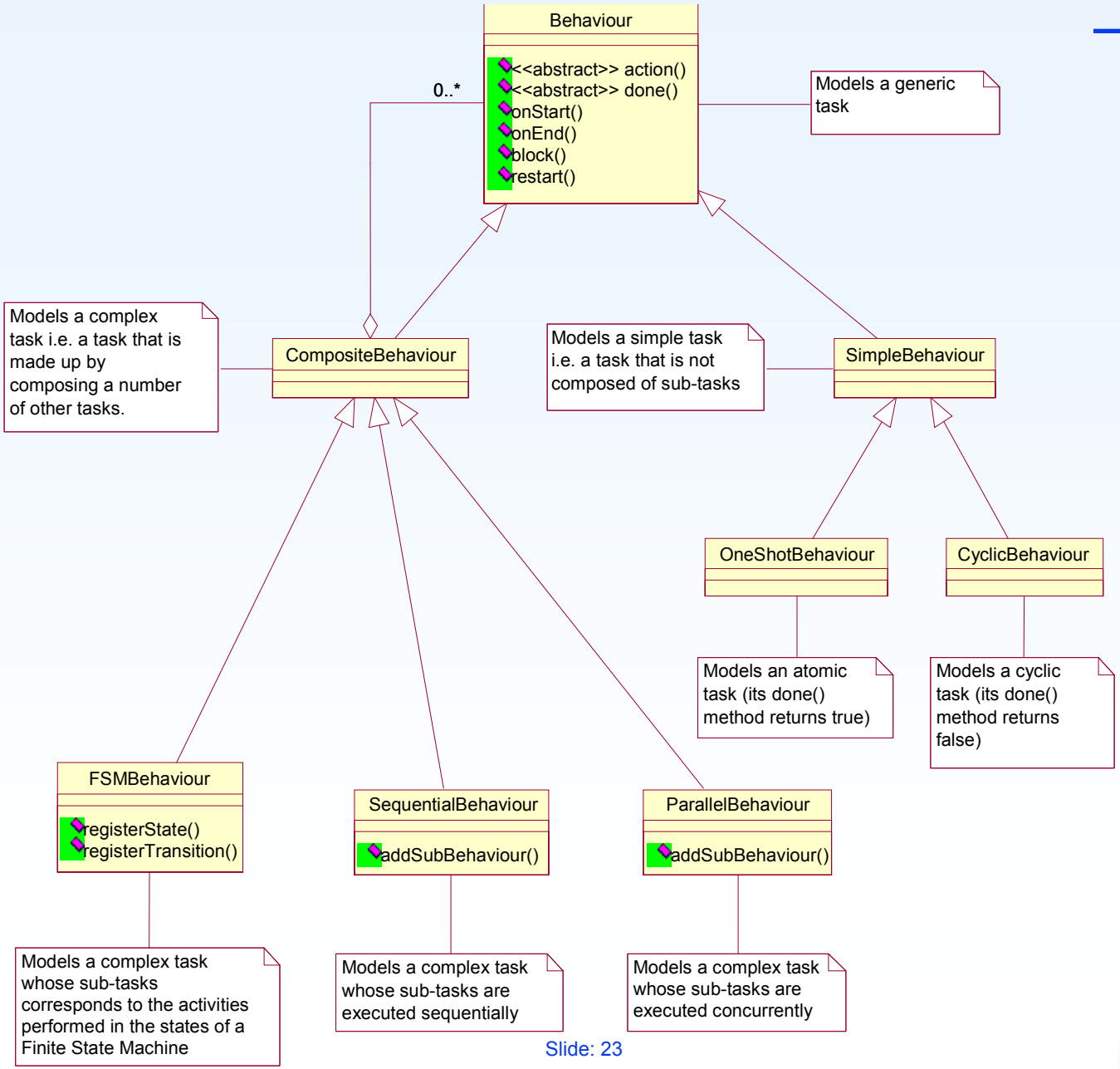
- **does JADE really comply with FIPA?**
  - strong active participation of the JADE team in FIPA (FAB, TC chair, …)
  - successfully participated in the 1st FIPA interop. tests in Seoul on Jan. 99
  - successfully participated in the 2nd FIPA interop. tests in London on Apr. 01
- **does JADE really simplifies the development of MAS?**
  - pay-as-you-go philosophy
  - How much of FIPA is hidden to programmers?
    - is FIPA for agent developers or for agent platform developers?
- **Is JADE too low-level?**
  - JADE provides the communication layer and the task scheduling layer
  - JADE can be easily integrated with a reasoning layer (JESS, PARADE, …)
- **LGPL license**
  - respect and protect both the users and the authors

Some conclusions (2/2)

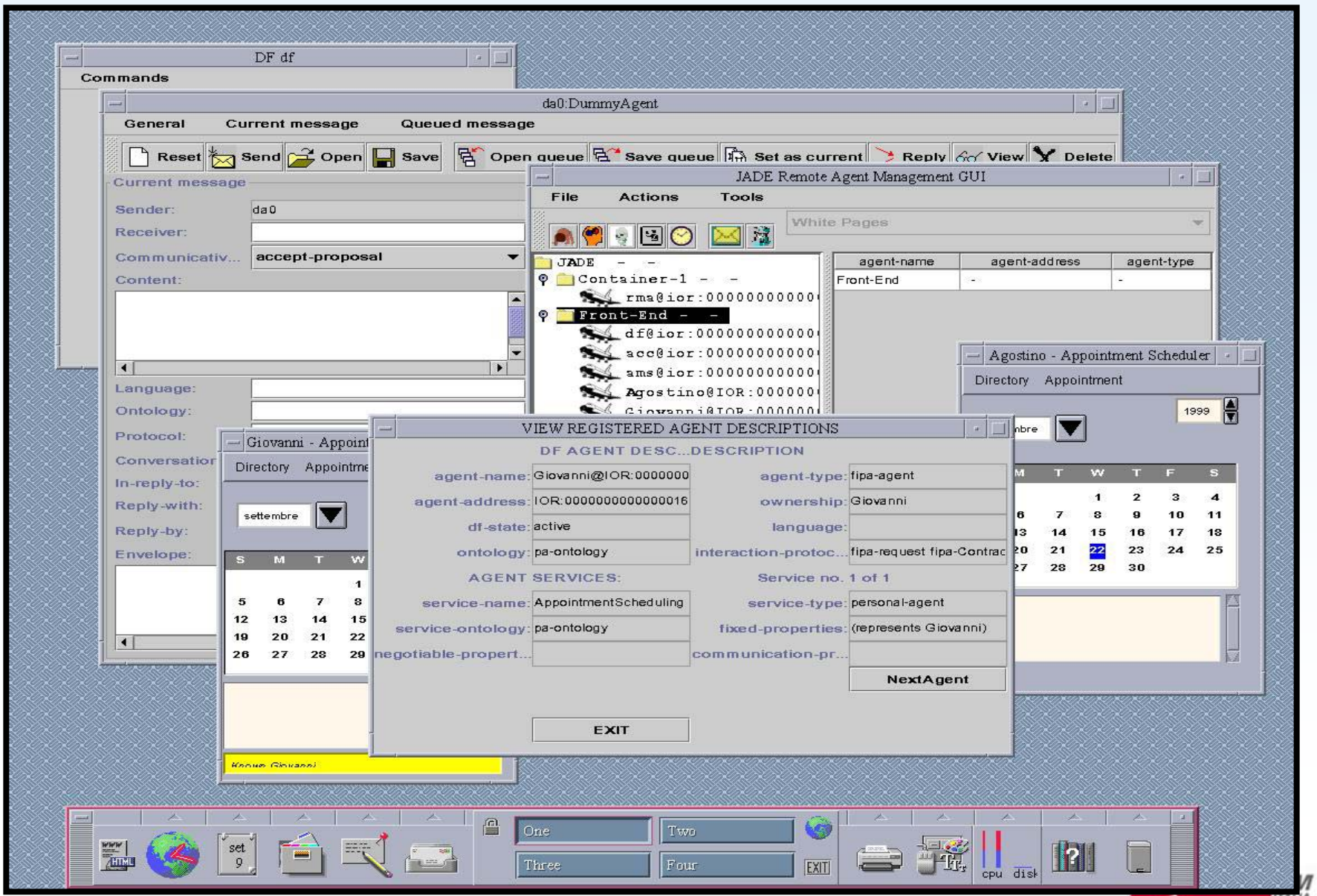
- **Some numbers (useless)**
  - > 3000 downloads since Feb. 2000
  - > 600 users registered on the mailing list **jade-develop@sharon.cselt.it**
  - > 700 exchanged and archived e-mails
- **Some credits (useful)**
  - JADE is the result of a joint collaboration with the University of Parma
  - thanks to all the external contributors:
    - Daniel Le Berre, Kaveh Kamyab, Bernard Burg and his team at HP Palo Alto, EPFL, …
- **Standards and Software tools, Agents?**
  - standard and sw tools are enabling components but
    - the technology needs a scope (agents to do what?)
    - from multi-agent systems/platforms to multi-agent applications
    - killer applications of mainstream technologies
    - pragmatic view of the theory (OO theory vs. OO usage)

Spare slides

Package jade.core.behaviours



JADE - Some graphical tools to support development



Concurrency in JADE

- **different containers on the same platform**
  - 1 JVM per container
- **different agents on the same container**
  - run in a preemptive multi-threaded environment scheduled by the JAVA Virtual Machine
- **different behaviours on the same agent**
  - scheduled cooperatively
    - every behaviour must release the control to allow the other behaviours to be executed
    - no stack to be saved, more effort to the programmer
    - JADE scheduler carries out a round-robin non-preemptive policy among all behaviours in the ready queue
  - Behaviours can be composed into a tree
    - every Behaviour is a Finite State Machine
      - *one state per execution time slot*

Agents as a system building paradigm
THE Problem: Interoperability

Methodologies, technologies, and tools currently available allow to develop software more and more powerful and complex

BUT

All the produced software is not able to interoperate with other software if no specific provision was made at the design time

THEREFORE

It is necessary to define new software development methodologies and technologies that allow to transfer this interoperability burden from the designers to the software program itself

- **Shift from making better components to making better the way the components cooperate**

FIPA and Compliance Testing

- **FIPA is not only ACL**
  - FIPA is also Agent Platform and Agent Services
- **MAS and the MAS environment are very dynamic in nature**
  - static compliance testing vs statistical compliance testing
- **compliance tests based upon access to internal of agents might not be viable in commercial applications**
- **the delegation pattern does not apply to standards**
  - if you need standards, you must work for that
- **compliance test activity has kicked-off in FIPA**
  - a workplan has been submitted with goals and milestones