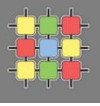
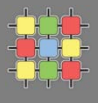


Architecture of software systems

Course 14: Intelligent systems, multi-agent systems

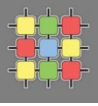
David Šišlák
david.sislak@fel.cvut.cz

Intelligent systems



» intelligent system integrates concepts of **artificial intelligence** such as:
  - knowledge reprepresentation and expert systems
  - advanced search methods
  - mathematical reasoning methods
  - **nature inspired computing**: artificial neural networks and genetic algorithms
  - **agent-based computing**: distributed artificial intelligence and multi-agent systems

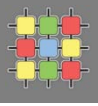
Motivation/vision thing



» in computer science and artificial intelligence there are 5 important trends:
- **ubiquity**: computation, internet access, agents to be everywhere
- **interconnection**: computation to be distributed, networked
- **intelligence**: in order to master very complex problems
- **delegation**: computers will be given control to imitate humans decisions
- **human orientation**: adapt to the way how we do reasoning

- how can **cooperation** emerge in societies of self-interested agents?
- what **language** can the agents use to communicate their beliefs and aspiration both to **agents** and to **people**?
- how can the agents **recognized** that their beliefs are in **conflict**?
- how can the **self-interested agents reach an agreement** with one another?
- how can they **coordinate activities** so that they achieve the goal **collaboratively**?

Introduction to agents #1

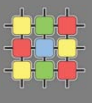


» **autonomous agents and multi-agent systems** (also referred to as agent-based computing):

- a specific sub-field of computer science and artificial intelligence,
- investigates the concepts of autonomous decision making, communication and coordination, distributed planning and distributed learning but also game-theoretic aspects of competitive behavior or logical formalization of higher level knowledge structures representing interaction attitude of actors in multi-actor environment.

A multi-agent system is a decentralized computational (software) system, often distributed (or at least open to distribution across hardware platforms) whose behavior is defined and implemented by means of complex, peer-to-peer interaction among autonomous, rational and deliberative units – agents.

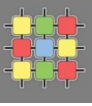
Introduction to agents #2



An **agent** is an encapsulated computational (or physical, even human) system, that is situated in some environment, and that is capable of flexible, autonomous behavior in order to meet its design objective (Wooldridge, 2000). The agent can exists on its own but often is a component of a multi-agent system.

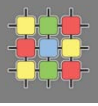
Agent technology provides a set of tools, algorithms and methodologies for development of distributed, asynchronous intelligent software applications that leverage the above listed theories.

Key properties of an autonomous intelligent agent



- **Autonomy** – the agent is accountable for execution of its own actions and is not controlled from outside. Often the agent's reasoning mechanism that selects the action to be executed is unknown from outside of the agent (unlike e.g. objects).
- **Reactivity** – the agent is able react qucikly to the events in the environment and to the requests from other agents, it is able to reconsider its activity according to the change of the environment in timely fashion. Often the longest reasoning cycle of an agent needs to perform faster than the fastest change in the environment (calculative rationality).
- **Intentionality** – the agents is able to maintain its long term intention encoded by the agent's designer and is capable to consider both the long term intentions and immediate reactive inputs when selecting the next action.
- **Social capability** – the agent is able to interact, collaborate, form teams but also to perform different levels of reasoning about the other agents.

Agent concepts



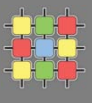
Agents use models:

- **agents as design metaphore**, providing the designers and developers with a way of structuring an application around autonomous, communicative elements;
- **source of technologies**, providing specific techniques and algorithms for dealing with interactions in dynamic and open environments and
- **simulation models**, providing strong models for representing complex and dynamic real-world environments.

Agents design levels:

- **organization-level**: related to the agent communities as a whole (organizational structure, trust, norms, obligations, self-organization, etc.);
- **interaction-level**: concern communication among agents (languages, interaction protocols, negotiations, resource allocation mechanisms);
- **agent-level**: concern individual agents (agent architecture, reasoning, learning, local processing of social knowledge).

Similar concepts



objects - computational entity with its encapsulated state, ability to perform methods on the state and communicating with the other objects via message passing

» lesser **degree of autonomy** - possibility to have a public method

» joint goal is set-up at the **design-time**

» multi-agent systems are inherently **multi-threaded**

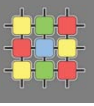
expert systems - the most important technology of the 1980's

» expert systems are **disembodied** from the environment

» expert systems are not capable of **reactive** and **proactive behavior**

» expert systems are not equipped with the **social ability**

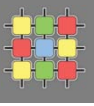
Research challenges and applications of agent-based computing #1



- **Coordination** - list of agent techniques (based mainly on dedicated coordination protocols and various collaboration enforcement mechanisms) that facilitates coordinated behavior between autonomous, while collaborative agents. Coordination usually supports conflict resolution and collision avoidance, resource sharing, plan merging, and various collective kinds of behavior.

- **Negotiation** - list of various negotiation and auctioning techniques that facilitate an agreement about a joint decision among several self-interested actors or agents. Here we emphasize mainly negotiation protocols and mechanisms how individual actors shall act and what strategies shall they impose to optimize their individual utility.

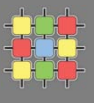
Research challenges and applications of agent-based computing #2



- **Simulation** - techniques that allow inspection of collective behavior of the interactive actors, provided that the models of the individual agents are known. Here we count on the versatile simulation frameworks that allow long-run complex simulation and various ”what-if” analyses of different problems. If distributed hardware system is modeled, agent-based simulation enables a close linkage between simulation and the real hardware machinery.

- **Interoperability** - set of techniques for achieving high level interoperability among software components developed by different designers, especially in the situation where the source code and complete models of behavior are not shared. Interoperability is studied on the level of physical connections via interaction protocols but also semantics of communication.

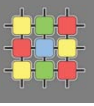
Research challenges and applications of agent-based computing #3



» **Adjustable Autonomy and Policies** - set of techniques and methods for specifying and dynamic adjustment of decision making autonomy of the individual actors in a multi-agent system. Various formal frameworks for specifying policies have been proposed and numerous policy management systems have been designed by the agent community.

» **Organization** - techniques supporting agents in ability to organize autonomously in permanent or temporal interaction and collaboration structures (virtual organizations), assign roles, establish and follow norms, or comply with electronic institutions.

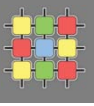
Research challenges and applications of agent-based computing #4



» **Multi-agent Learning** - in the multi-agent community there are various methods allowing an agent to form hypothesis about available agents. These methods work mainly with the logs of communication or past behavior of agents. Agent community also provides techniques for collaborative (distributed) learning, where agents may share learnt hypothesis or observed data. A typical application domain is distributed diagnostics.

» **Multi-agent Planning** - specific methods of collaboration and sharing information while planning operation among autonomous collaborating agents. Agent community provides methods for knowledge sharing, negotiation and collaboration during the 5 phases of distributed planning (Durfee, 1999): task decomposition, resource allocation, conflict resolution, individual planning, and plan merging. These methods are particularly suitable for the situations when the knowledge needed for planning is not available centrally.

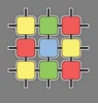
Research challenges and applications of agent-based computing #5



- **Knowledge Sharing** - techniques assisting in sharing knowledge and understanding different types of knowledge among collaborative parties as well as methods allowing partial knowledge sharing in semi-trusted agent communities (closely linked with distributed learning and distributed planning).

- **Trust and Reputation** - methods allowing each agent to build a trust model and share reputation information about agents. Trust and reputation is used in non-collaborative scenario where agents may perform non-trusted and deceptive behavior.

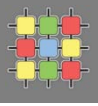
Agent applications #1



» **Decentralized scenarios**: Particularly suitable are the domains where the data and knowledge required for computation are not or cannot be available centrally or the process physical system control needs to be distributed.

- **Geographical distribution** of knowledge and control (e.g., logistics, collaborative exploration, mobile and collective robotics, pervasive systems) or the environments with partial or temporary communication inaccessibility.
- **Competitive domains**, with the restrictions on the information sharing (e.g., e-commerce applications, supply-chain management, and e-business)
- Domains with the requirements for **time-critical response** and **high robustness** in distributed environment (e.g., time-critical (soft-and/or hard-realtime) manufacturing or industrial systems control, with re-planning, or fast local reconfiguration)

Agent applications #2

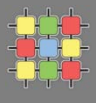


- **Simulation and modeling scenarios**: Using agents for simulation purposes has been very common, while the right justification was often missing. Agents shall be deployed in simulation exercises where we require, e.g., an easy migration from the simulation to deployment in real environment.

- **Open systems scenarios**: In scenarios requiring integration and interoperability among software systems that are not known a priori and whose source code may not be available - here the use of agent technologies, especially agent communication languages and interoperability standards is advisable.

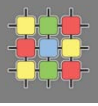
- **Complex systems scenarios**: In scenarios requiring modeling, controlling or engineering of complex systems. Decomposition of the decision making into separate agents' reasoning and solving problems by means of negotiation represents a novel software development paradigm.

Agent applications #3



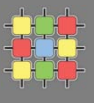
» **Autonomy oriented aspects** of agency is appropriate in application domains with high requirements for systems with decision making autonomy, when the user delegates the substantial amount of decision making authority to the system and when the system is expected to cope independently with unexpected situations.

Agent domains



- **manufacturing**: planning highly complex production, control of dynamic, unpredictable, unstable processes, diagnostics, repair, reconfiguration/replanning.
- **virtual enterprises**: forming business alliances, forming long-term/short-term deals, managing supply chains.
- **internet agents**: mainly for intelligent shopping and auctioning, information retrieval and searching, remote access to information and remote system control.
- **transport**: intelligent car, public transport, logistic and material handling, but also peace-keeping missions, military maneuvers, etc.
- **collective robotics operations**: cooperation and autonomy in the group of robotic entities (UAS, ground vehicles, unattended sensors), replacement of teleoperation with autonomous decision making
- **utility networks**: energy distribution networks, mobile operators networks, cable provider networks - simulation and predication of alarm situations, prevention to black-out and overload, intrusion detection.

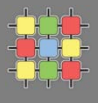
Agent architectures #1



» **Reactive agents** are agents that contain no symbolic knowledge representation (ie: no state, no representation of the environment, no representation of the other agents, ...). Their behaviour is defined by a set of perception-action rules.

$$\text{rules} \times \text{percept} \rightarrow \text{action}$$

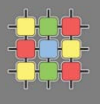
Agent architectures #2



The classical approach to building agents is to view them as a particular type of knowledge-based system, and bring all the associated methodologies of such systems to bear. We define a **deliberative agent** or SR agent architecture to be one that:

- contains an explicitly represented, **symbolic model of the world**;
- makes decisions (e.g. about what actions to perform) via **symbolic reasoning**.

Agent architectures #3



Belief-desire-intention (**BDI**) model is framework for reasoning about formal abstract models of mental states (based on Theory of Practical Reasoning).

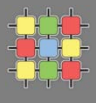
» contains representations (as objects, data structures, or whatever) of:

- **beliefs**, which constitute its knowledge of the state of its environment (and perhaps also some internal state),
- **desires**, which determine its motivation what it is trying to bring about, maintain, find out, etc.,
- **intentions**, which capture its decisions about how to act in order to fulfill its desires (committed desires)

» **intention** is something between the agents' **state of mind** (belief) and the immediate action to be performed

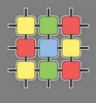
» unlike **desire/goal** an intention may be seen as agents immediate **commitment** to implementing an action.

Multi-agent system from developer perspective



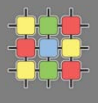
- Software agents which interact to achieve a common goal
- Heterogeneous agents
- Different execution environments - PCs, embedded systems, industrial control systems
- Message-based interaction between agents

Examples of software agent applications



- User Interface Agents
  - Personal Assistants
  - Health-care Assistants
- Information Management Systems
  - Resource Discovery Agent
  - Travel Assistants
- Business Applications
  - Energy management
  - Logistics
  - Supply chain management
  - Traffic control management
- Simulations

Multi-agent platforms #1



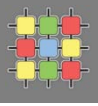
Basic features

- Simplify and speed-up a development process
- Ensure safe and efficient execution environment
- Allow seamless interaction between individual agents

Various levels of provided services

- Toolkits for creating agent applications (UML-like tools), model-driven approaches
- Agent programming languages, interpreters
- Full platforms - self-contained systems that provide runtime libraries, APIs for developers

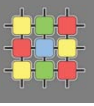
Multi-agent platforms #2



From software development point of view

- Middleware which provides developers with execution environment and programming API
- Agent life-cycle management
- Communication infrastructure
- Set of basic services which can be utilized by agents residing on platform (like service discovery)
- Security and mobility services
- Monitoring and logging components
- Additional optional modules - advanced reasoning, machine learning, interaction protocols

Interaction schemes

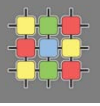


From the point of view of interaction among agents we distinguish:

» Open systems
- Interaction among various types of agents from different developers
- Common understanding of messages is necessary – specification of message structure
- May act as self-interested and try to harm others/whole system
  - security issues
- Strong emphasis on interoperability

» Closed systems
- Interact with a predefined set of agents known to the developer in advance
- Proprietary data formats can be used
- Interoperability can be sacrificed for the sake of performance optimizations

FIPA specifications #1



Foundation for Intelligent Physical Agents - FIPA

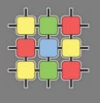
» Founded to create specification that will ensure interoperability among agents

» Complete set of specifications from different categories:

- agent communication
- agent transport
- agent management
- abstract architecture and applications

» Most significant for agent interoperability is agent communication and transport

FIPA specifications #2



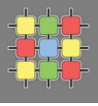
In order to be FIPA compliant, concrete architectural specifications must have certain properties

- Mechanisms for agent registration
- Agent/service discovery service
- Inter-agent message transfer

Defined by FIPA Abstract Architecture specification together provide support for:

- concurrent execution of agents - multiple agents running in parallel on the same host
- distributed execution of agents - agents distributed over multiple physical hosts

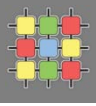
Agent interoperability



- Software created by different developers and at different times works together in seamless manner
- Limitations of current software interoperability
  - Lack of the ability to communicate definitions, theorems and assumptions
  - No general way of resolving inconsistencies in the use of syntax and vocabulary

- ACL by Genesereth and Fikes, 1992 - three cornerstones
  1. Vocabularies (ontologies)
  2. Knowledge Interchange Format (KIF)
  3. Knowledge Query Manipulation Language (KQML)

Agent communication language #1



Categories of requirements for an ACL

» Form - it should be declarative, syntactically simple and easily readable

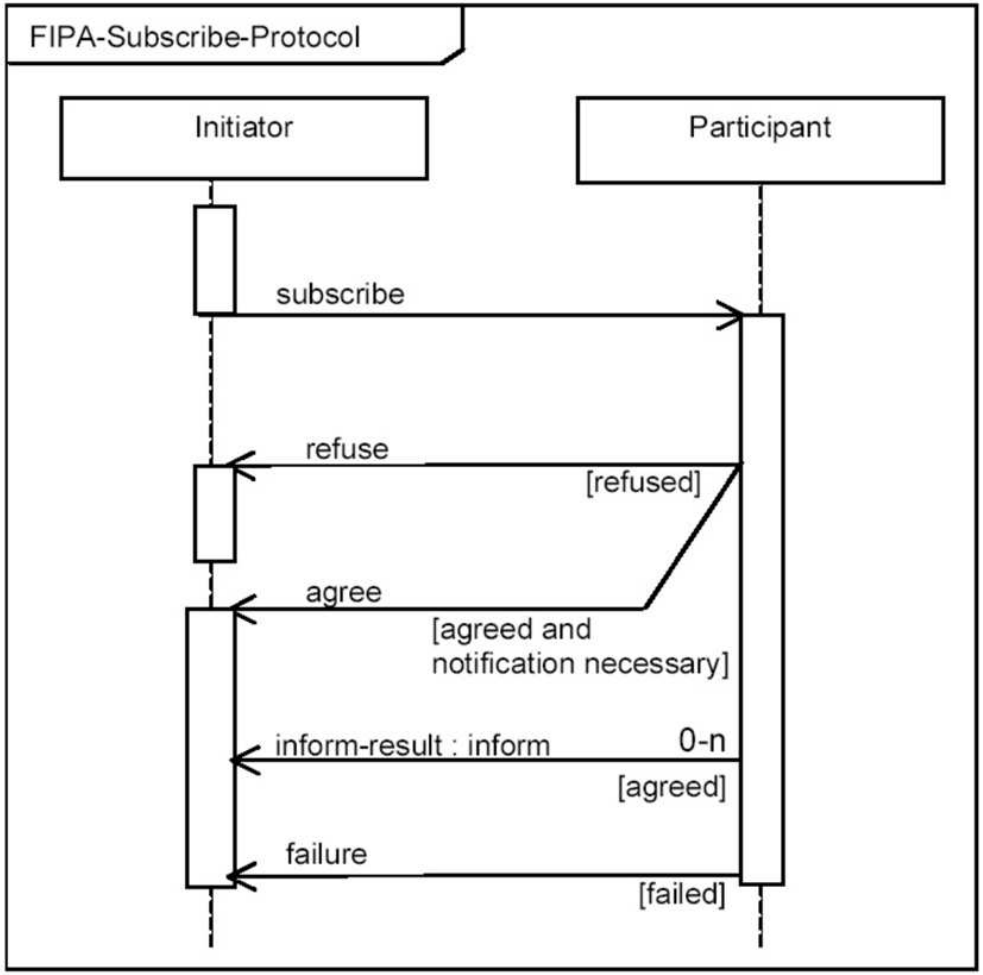
» Content - a distinction between the languages that express communicative acts and the language that conveys the content of the message

» Semantics - should exhibit properties expected of the semantics of any other language

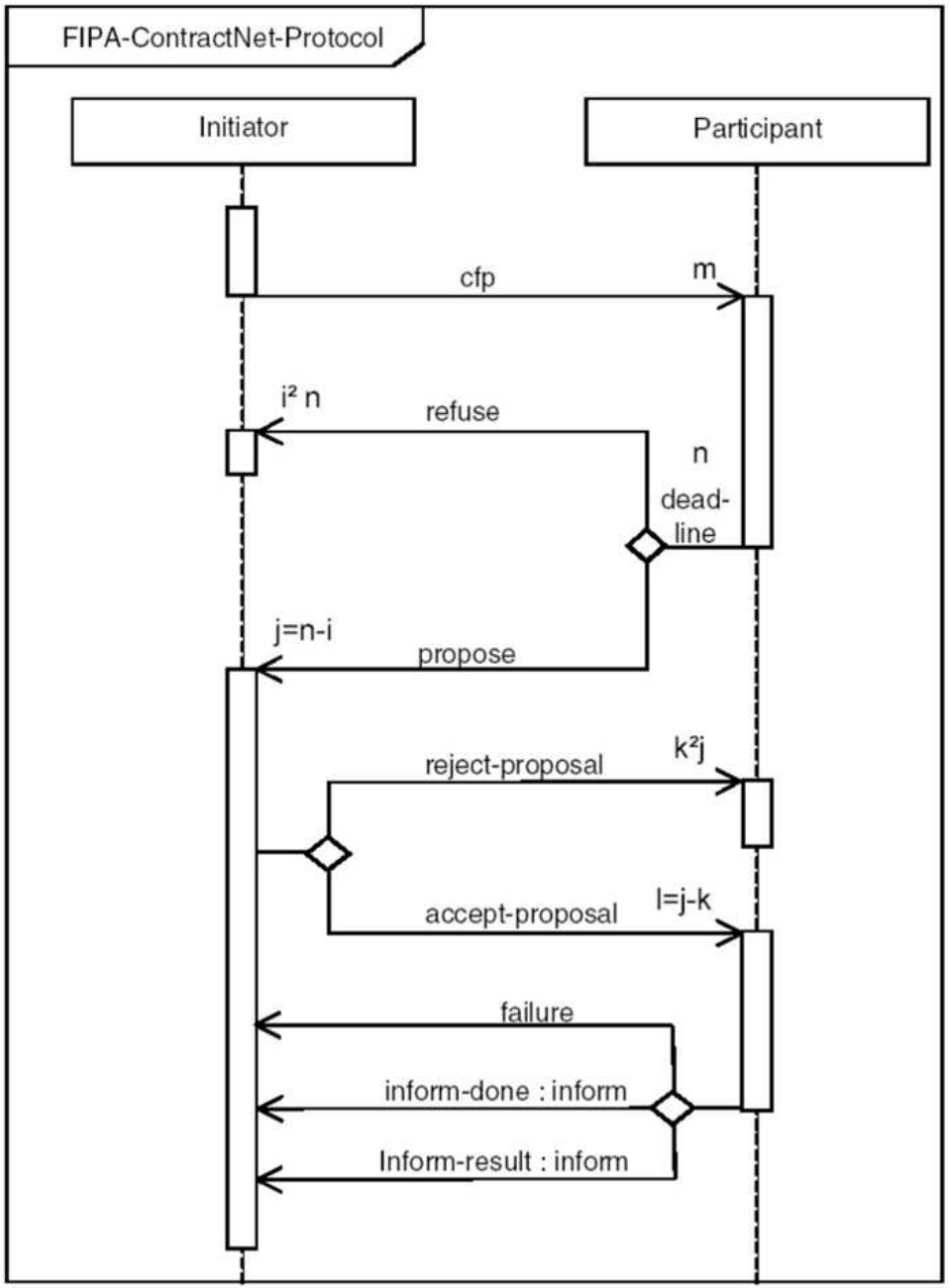
» Implementation - efficient, provide a good fit with existing software, hide the details of lower layers

Agent communication language #2

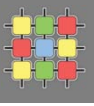
- **Networking** - support all important aspects of modern networking technology, independent of transport mechanism
- **Environment** - it must cope with heterogeneity and dynamism
- **Reliability** - support reliable and secure agent communication



Agent communication language #3



Communication between agents

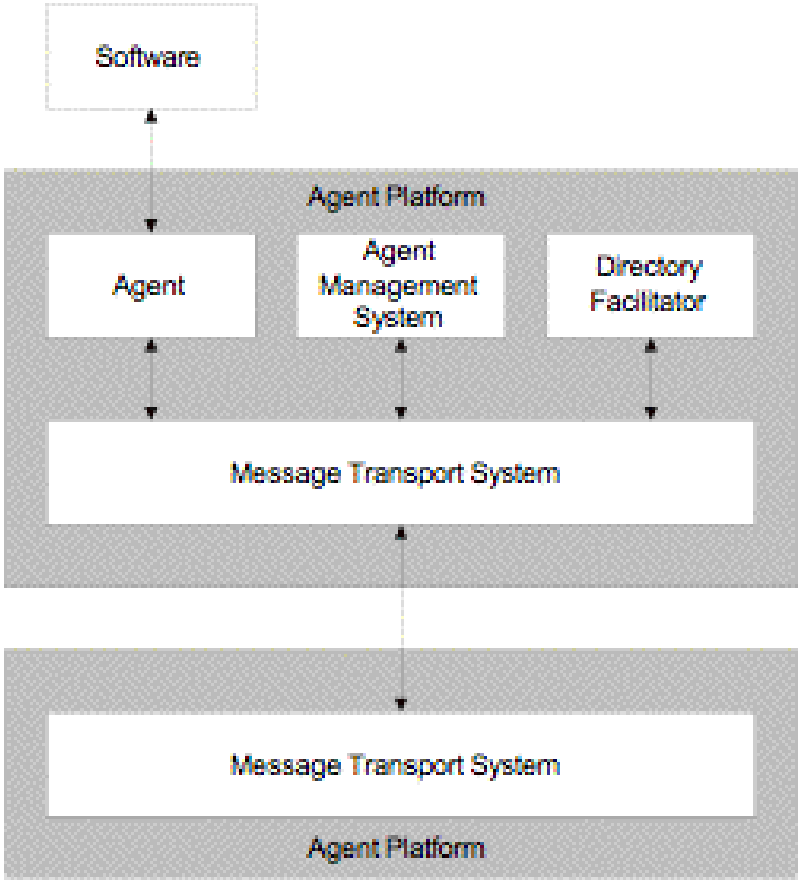


- Agents communicate by exchanging messages which represent speech acts (INFORM, QUERY, etc.)
- Messages are encoded in an agent communication language
- Messages include indication of the type of communicative act, names of sender and receivers, content of the message itself
- Content of the message is specified by ontology
- Unicast, multicast, broadcast messages
- Message is delivered via message transport

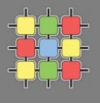
FIPA management reference model #1

Provides normative framework within which FIPA-compliant agents exist and operate.

It establishes the logical reference model for creation, registration, location, communication, migration and retirement of agents.



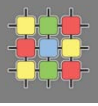
FIPA management reference model #2



Basic components of the reference model are:

- Agent platform provides physical infrastructure in which agents can be deployed, it can be viewed as a kernel responsible for thread, socket and memory management
- Agent is computational process that implements the autonomous, communicating functionality of an application
- Agent must have an identification which is unique within the agent universe
- Agent Management System is mandatory component, it maintains a directory of agent identifiers and transport addresses
- Directory facilitator is optional component, it provides yellow pages services
- Message transport is responsible for physical message delivery, each agent must be registered at message transport

Mobile agents #1



Mobile agents are characterized by code mobility

» Mobile agents travel to places, where they perform tasks on behalf of a user

» Reason why to travel - insufficient computational power, unreliable communication, remote data source

» Security issues

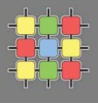
» Mobility vs. cloning, stand-in agents

Types of agent mobility

» **Strong mobility** - mobility of code, data and execution state, transparent for the computational process, requires support from the OS and execution environment

» **Weak mobility** - only the code and data are transferred, intentional mobility

Mobile agents #2



Problems with agent mobility

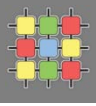
» Various operating systems, programming languages, agent platforms

» Security features - permissions, authorities, access control

» Necessity to transfer all required data, libraries

» Interaction with message transport system - new communication address, delivery of messages received by old message transport

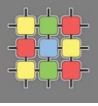
Security



Important aspect especially in the case of open systems

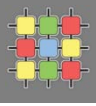
- Thread-safe agent execution model which holds rest of the system harmless against agent failure
- Communication security
  - message encryption/signing
  - security certificates with public/private keys
- Trust and reputation models to create a set of trustful collaborators in open systems
- Protect private knowledge of individual agents

Overview of agent platforms



- In total more that 150 agent platforms and toolkits, most of them obsolete or discontinued
- Provided under an open source license
  - JADE
  - Cougaar
  - AGLOBE
- Commercial platforms
  - Jack
  - LS/TS
  - Cybele

AGLOBE agent platform



Developed at ATG, CVUT Prague

- Strong focus on distributed simulations
- Scalability, high number of fully autonomous agents, lightweight infrastructure
- Agent migration, persistence, communication inaccessibility
- Library management
- Java-based, required JDK 6
- Available on http://agents.felk.cvut.cz/aglobe