Agent-oriented Computing

Franco Zambonelli

April 2010

Goal of Today

- What are "Agents" and why are hey useful?
- How do they impact on software development?

- Three key themes (i.e., three parts of today's course):
  - Autonomous Agents
  - Multiagent Systems
  - Agent-oriented Software Engineering

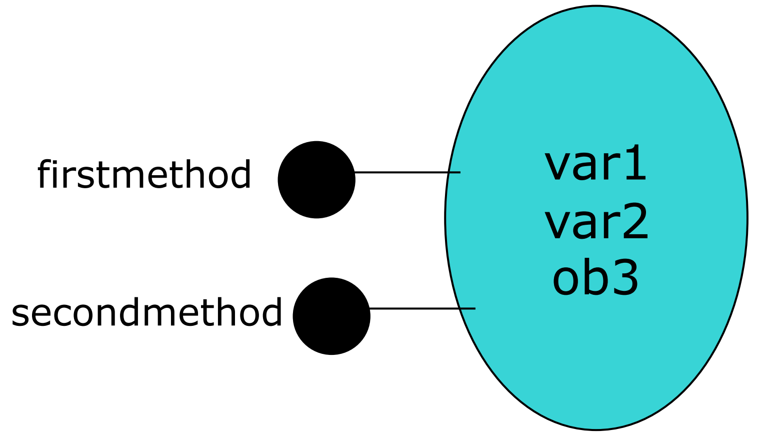
Outline of 1st Part:

- What are Autonomous Agents?
  - Objects vs. Agents
  - Definitions
- Agent Architecture
  - Reactive
  - Goal-oriented
  - Utility-oriented
- Agent Systems
- Agent examples
- Agent Applications

Objects: the “Classical Perspective”

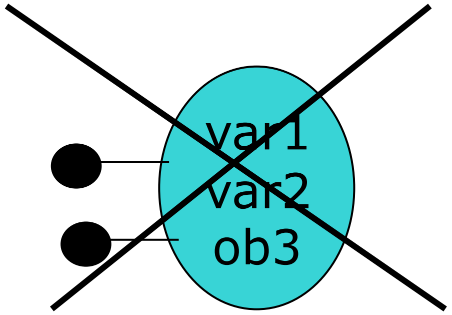
- This is what we learn an object is:
  - State (instance or state variables)
  - Methods (operations)
- Methods are requested by other objects

```
Public class MyClassicalObject {
5nt var1;
char var2;
Object ob3;

public int firstmethod(int x)
{ int results;

result = someop(x);
return x;
}
Public void secondmethod()
{ System.dosomeaction();
ob3.invokesomemethod();
}
}
```



Objects: the Real Scenario

- Actually, other than state and methods
  - Internal threads
  - Event-handling
  - Messaging
  - Access to contextual information



```
public class MyModernObject
Implements Threads, EventListener{
int var1;
char var2;
Object ob3;

public int firstmethod(int x)
{ int results;

result = someop(x);
return x;
}
Public void secondmethod()
{ System.dosomeaction();
ob3.invokesomemethod();
}
}

run()
{
Context lc = Naming.lookup
("LocalContext");

Object cooler = Naming.lookup
("CoolSystem");
```

So What?

- Is this still an object?
  - It does much more than an object
  - It contains much more components and characteristics

- Would you still call a car enriched with a reaction engine, capable of flying, with an automated pilot, still a car. Or would you rather invent another name (e.g., “airplane”) to refer to it?

From Objects to Agents

- The “grown-up” objects of modern adaptive software are
  - Not purely functional (they do not simply answer to request of services but rather try to achieve an objective, a goal)
  - Capable of unsolicited execution (due to internal threads)
  - Adaptive (they can dynamically acquire information and tune their behavior accordingly)
  - Situated (access to contextual and environmental information)
  - Social (they interact with each other either via messaging or via mediated interactions via the environment)
- This is very close to the definition of agents…

The Concept of Agency

- From the Webster Dictionary
  1. how a result is obtained or an end is achieved; "a means of control"; "an example is the best agency of instruction"; "the true way to success"
  2. a business that serves other businesses
  3. an administrative unit of government; "the central Intelligence Agency"; "the Census Bureau"; "Office of Management and Budget"; "Tennessee Valley Authority"
  4. **the state of serving as an official and authorized delegate or agent**
  5. **the state of being in action or exerting power; "the agency of providence"; "she has free agency**"
- From the Latin “agentis”: “those who act”
- So, an “agent” is someone who act on behalf of other, with power to act derived from a delegation

Examples of Real-world agents

- Secret Agents
- Travel Agents
- Real Estate Agents
- Sports/Showbiz Agents
- Purchasing Agents

- What do these jobs have in common?
  - They engage is tasks each with a specific goal (e.g., finding a spy, selling a house, finding a job for soccer player, etc.)
  - They are delegate by someone (the government, a house owner, a soccer player)
  - They know how to do (have the power and the knowledge to do)

Software Agents

- In general, we can talk of “software agents” when
  - Referring to software that has a “goal” to pursue
  - Acting on our behalf to pursue that goal
  - Having the power and knowledge to pursue this goal in autonomy
- “Agent” is one of the more ubiquitous buzzwords in computer science today.
  - It’s getting used for almost any piece of software
  - In several cases, unappropriately
- In any case, we need some more “technical” characterization and definition

Examples of Software Agents

- Filtering agents (antivirus, anti-spam)
  - They have a goal to achieve → monitoring resources and filter viruses and spams
  - They are fully delegated to act on our behalf → we do not even want to know what and how they are acting → we trust them!
  - They know how to do (have the code to analyze streams, and the knowledge – i.e., the filter rule – to act)

- Shopbots/price comparison agents
  - They have a goal to achieve → find a good with a low price
  - They are fully delegated to act on our behalf → We only want to know the final result
  - They know how to do (have the code to access XML Web resources, and the knowledge to interpret XML files describing goods)

Software Agents: Definition

- A software agent is a component that is
  - **Goal-oriented**: designed and deployed to achieve a specific goal (or to perform a specific task)
  - **Autonomous**: capable of acting in autonomy towards the achievement of its specific goals, without being subject to a globally controlled thread of control
  - **Situated**: it execute in the context of a specific environment (computational or physical), and is able act in that environment by sensing and affecting (via sensors and actuators)
- In addition, it can be
  - **Proactive**. It can act opportunistically and in an unsolicited way towards the achievement of its goals (as opposed to Reactive agents, that acts only on reactions to events)
  - **Social**. Interact with other agents in a multiagent systems.

The Concept of Goal-orientedness

- How is a global application goal is achieved?
- Division of labor (as in object-based applications)
  - Functions assigned to different components
  - Coordination is for composing functionalities to lead to global goal
  - As in pipe organizations
- Division of responsibilities (as in agent-based applications)
  - Sub-goals assigned to different components
  - Coordination is for orchestrating the achievement of a global goal
  - As in modern distributed organizations

The Concept of Autonomy

- Related to “decision making”
- Centralized decision making, as in process-based and object-based applications
  - global goal achieved via a *global control scheme* for the application entities
  - design by *delegation of control*
- Distributed decision making, as in agent-based applications
  - sub-goals assigned to autonomous agents (integrating execution capabilities, i.e., threads) which try to achieve in autonomy their own goal
  - design by *delegation of responsibility*
  - *Agents can say “NO!”*

The Concept of Situatedness

- We have already discussed that context-awareness is important for adaptivity
  - And it is even more important when
    - Goal-orientedness
    - Distributed decision making
  - Are involved
- Objects are typically not situated: they interact in a world where everything is an object
- Unfortunately, there are also several agent systems that does not take situatedness into the proper account…
- Clearly, autonomy and situatedness make agent **adaptive entities**, suitable for the dynamics of modern software scenarios!!!

The Concept of Proactivity

- Not only agents have autonomous decision-making capabilities
  - They can also decide to autonomously activate towards the pursuing of the goal
  - They do not need any specific event or solicitation to do that

- Proactivity is a sort of extreme expression of autonomy
- Reactive agents are the less autonomous
- Proactive are the more autonomous

The Concept of Sociality

- Agents are rarely living in an isolated mono-agent world
  - They usually live in a multi-agent world
- Sociality refer to the fact that the typically interactions are more sophisticated than client-server ones
  - Exchange of knowledge
  - Delegation of tasks
  - Open world, competitions in actions, negotiations
- Mediated interactions via common portions of the environment
- Resembling more the interactions occurring in a society of humans…
- Clearly, the capability of acting in a social context is expression of adaptivity, and will make it possible to build, with agents, very **adaptive and complex systems**, able to deal with openness of the system and (together with situatedness) with environmental dynamic!!!

Classical Object vs. Agents

- Function-oriented vs. Goal-oriented
- Centralized decision making vs. decentralized (and responsible) decision making
- “all is an object” vs. “agents and environment”
- Objects are purely reactive while agents can be proactive
- Interactions in objects are merely client-server and devoted to transfer of execution control, interactions in agents can be more sohpisticated and involve communication and negotiation, as in real-world human societies

- Clearly, it is not always black and white…

Modern Object vs. Agents

- Modern objects have features that can make objects resembles agents…
  - They can have autonomous threads of execution
  - They can handle events
  - They can exploit the MW services to sense and effect contextual information

- In effect, several systems for "agent-oriented programming" can be considered simply as advanced tools for object-oriented programming
  - Several Java agents are grown-up objects
  - However, it is also possible to conceive very different internal architectures for agents

Agents vs. Intelligent Agents

- The concept of agency we have given is often considered very weak
  - For may persons, agents do not simply have to be goal-oriented, autonomous, situated
  - They have to be “intelligent”
- Traditionally, this means they have to integrate “artificial intelligence” tools
  - Neural networks
  - Logic-based reasoning
  - Conversational capabilities (interact via a conversation language)
  - Etc.
- But what does intelligence really mean?
  - Can we really define intelligence?
  - Or it is in the eyes of the observer?

The Intentional Stance

- We often speak of programs *as if* they are intelligent, sentient beings:
  - The compiler can’t find the linker.
  - The database wants the schema to be in a different format.
  - My program doesn’t like that input. It expects the last name first.
- Treating a program as if it is intelligent is called the intentional stance.
  - It doesn’t matter whether the program really is intelligent; it’s helpful to us as programmers to think as if it is.
- In agent-based computing
  - Goal-orientation, Autonomy, situatedness
  - Can be conceived as observable expressions of intelligence
  - Event if it is simply a Java program after all…

The Knowledge Level

- The intentional stance leads us to program agents at the *knowledge level* (Newell).
  - Reasoning about programs in terms of:
    - Facts and Beliefs (rather than variables and data)
    - Goals and behaviors (rather than functionalities and methods)
    - Desires/needs/wants/preferences
- This is often referred to as *declarative* programming.
  - It is a different way of thinking and representing things
- We can think of this as an abstraction, just like object-oriented programming.
  - Agent-oriented programming

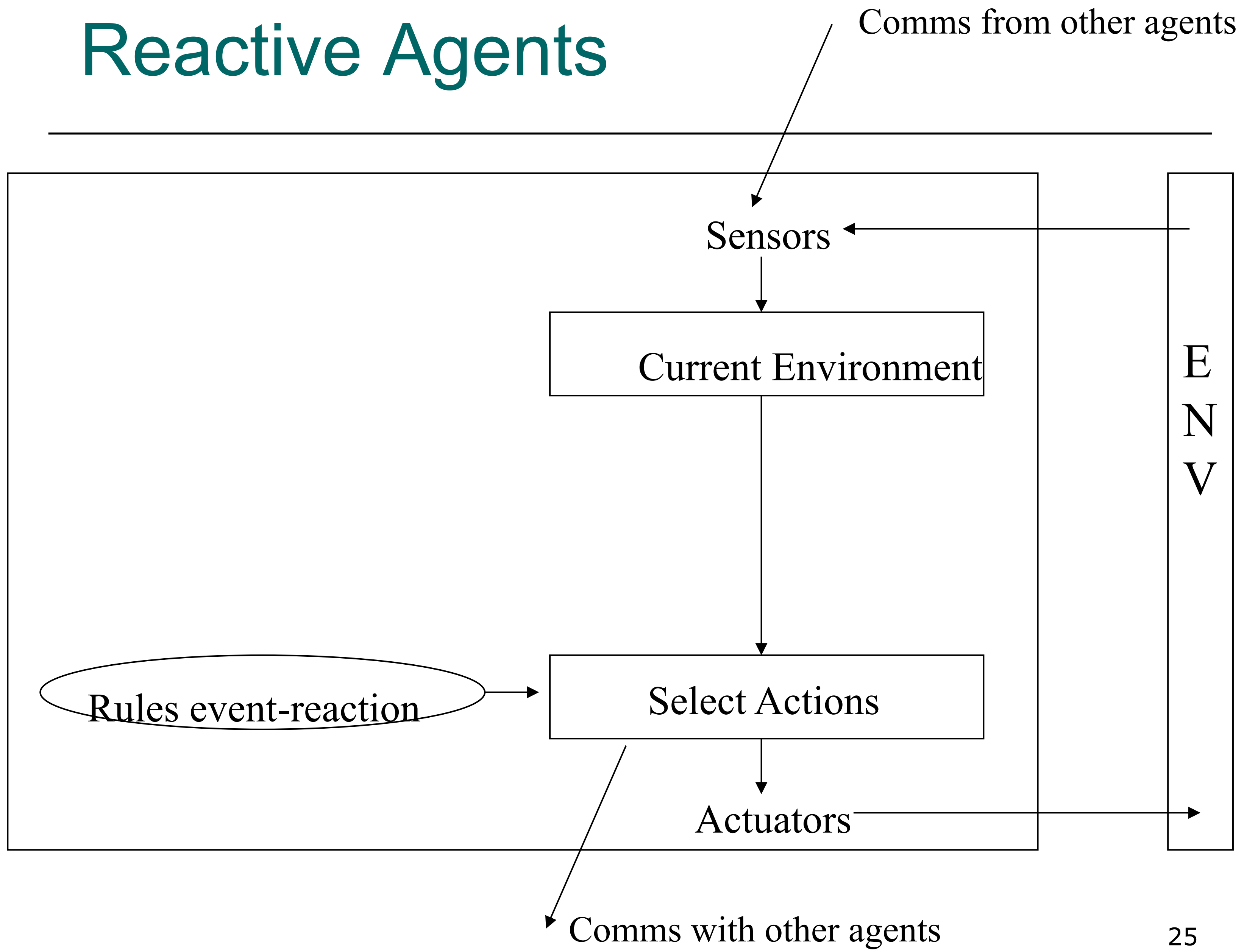
Agent Architectures

- What types of architectures can we conceive for agent?
  - Reactive (or tropistic)
  - Reactive with State (hysteretic)
  - Goal-oriented
  - Utility-oriented

Reactive Agents

- Perceive events
  - Simple set of rules event → action (i.e., activation of a specific behavior)
  - Actions are often known as "**behaviours**"
- Example of a simple "mail agent":
  - **if** send mail **then** check virus
  - **If** new mail **then** check spam
  - **If** spam **then** send message to friends agents
  - **If** new message **then** get new spam information
- Pros:
  - simple and efficient
- Cons:
  - Action depending only on stimuli
  - Not flexible
  - Not really autonomous

Reactive Agents

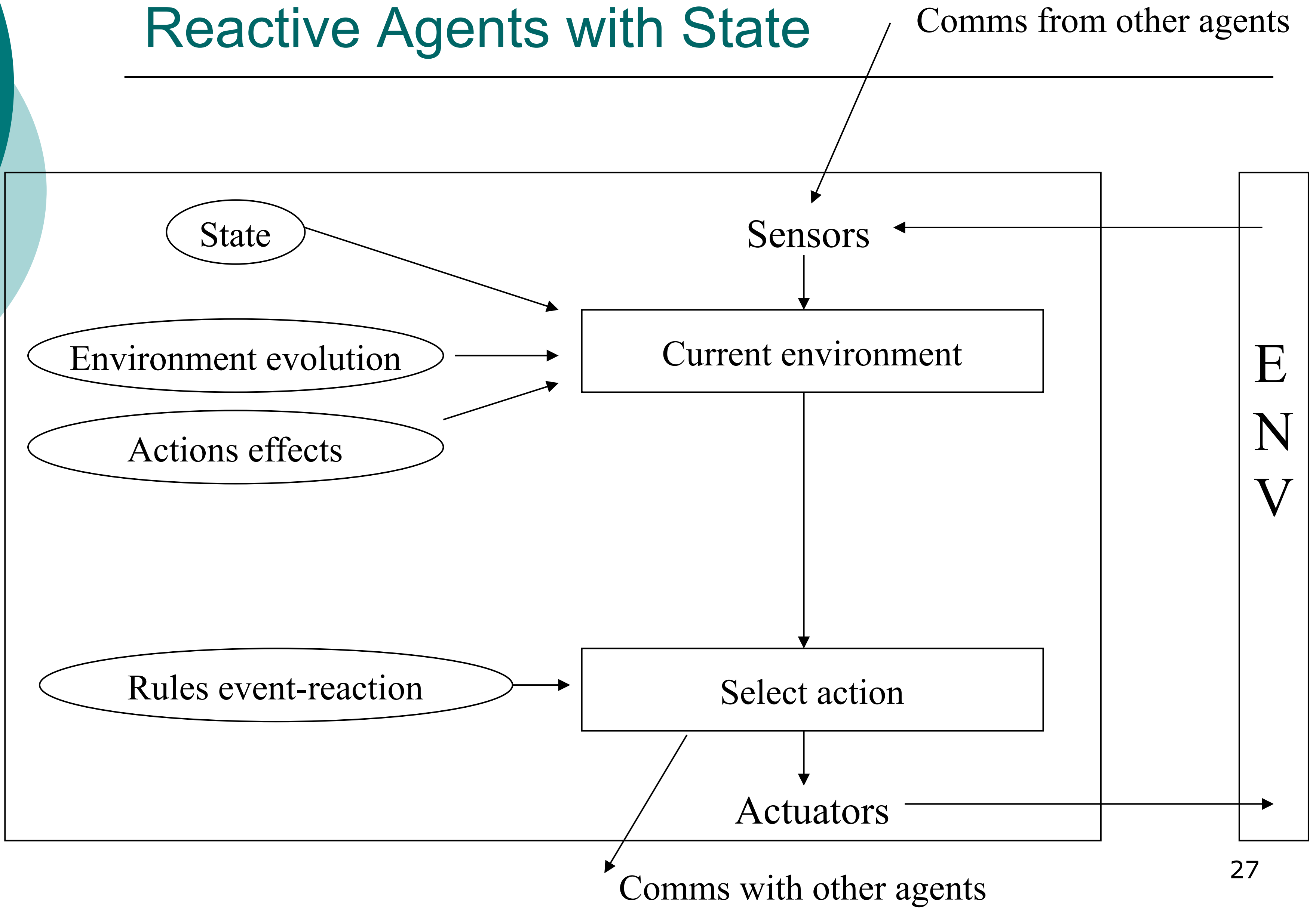


Reactive Agents with State

- Internal state (internal knowledge)
- Update of internal state
  - New state = actual perception + old state
  - The update may require
    - Knowledge on how the world evolves – which can also dynamically acquired by the agent
    - Knowledge on how the agent actions influence the world
  - Select action (i.e., behavior) accordingly
- Example
  - A mail agents that keeps track of the users marking some messages as “spams” and take these into account in future actions

- An object is a sort of reactive agents, but
  - It has no rule for action selection
  - It actions are directly commanded by the external

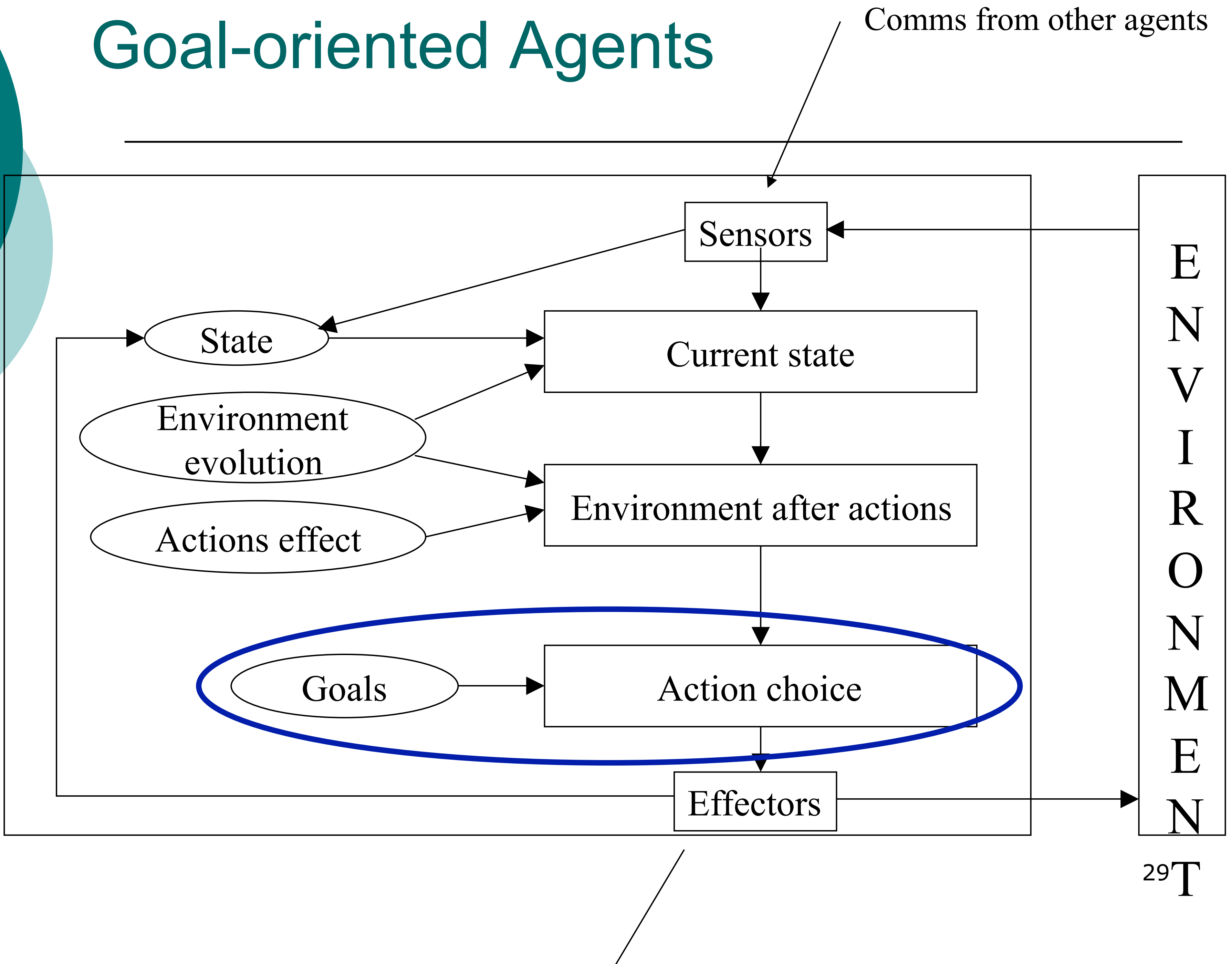
Reactive Agents with State



Goal-oriented agents

- Goal → a desired situation to eventually achieve
- The agent exploits the goal and its knowledge
  - select actions whose effect would be that of approaching the goal
- How can an action be selected?
  - Search in the state space
  - Plannings
  - Heuristics → sub-optimal actions
- Example: an agent to minimize fragmentation in a hard-disk
  - Knapsack problem
  - Do not know the future but know the past
  - Select allocation of new files based on some heuristics
  - An action do not necessarily minimize the current fragmentation
  - Perform de-fragmentation action when the computer is idle

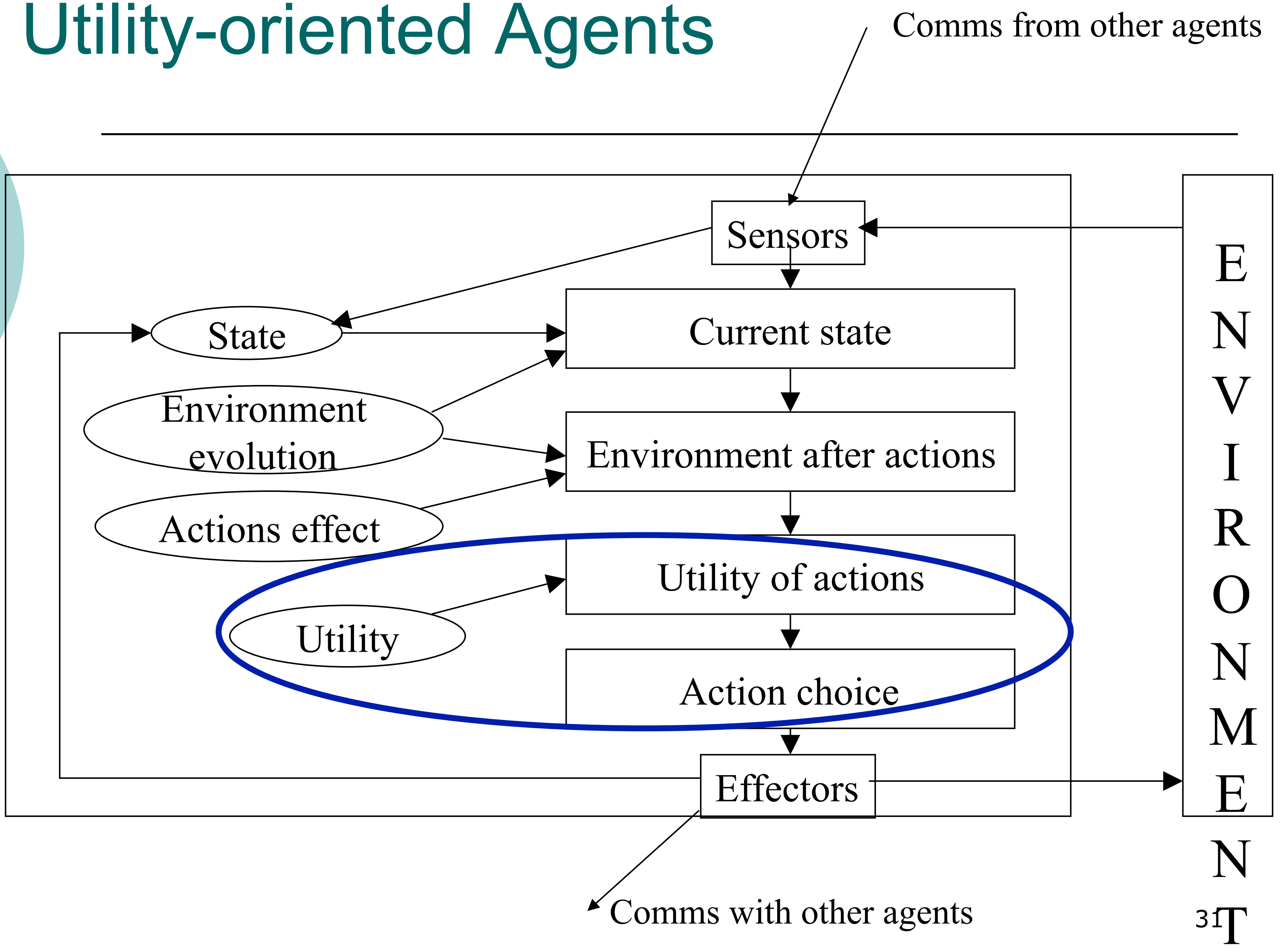
Goal-oriented Agents



Utility-oriented Agents

- The Goal is that of maximizing the current utility
  - opportunistic behavior
- Utility
  - A function of some parameter, measuring the state of "goodness" (with respect to the agent) of a situation
  - Often, it measures a trade-off between contrasting objectives
- Example
  - An agent to maximize CPU utilization
  - Always select the ready process
  - The current choice may be sub-optimal with regard to the global execution time of processes

Utility-oriented Agents



Hybrid Architectures

- Mixing utility and goals
  - An agent that has to achieve a goal and, at the same time, has to maximize a specific utility function
    - Trade-off between the two goals, which may be contrasting
  - Often, the various ways to approach a goal can be quantified by a utility function
    - Do the actions that approach the goal with the maximal utility

- Mixing reactive and goal-oriented behavior
  - A long terms goal that include several short term actions on the environment
  - That could lead to sub-optimal choices

Rational Agents

- How do we determine the right thing for an agent to do?
  - If the agent's internal state can be described at the knowledge level, we can describe the relationship between its knowledge and its goals (or utility function).
- Newell's Principle of Rationality:
  - If an agent has the knowledge that an action will lead to the accomplishment of one of its goals (or to the maximization of its utility), then it will select that action
  - This clearly applies to human agents too
- Game Theory and Decision Theory is indeed of great importance in modern software development!!!!

Preferences and Utility

- Agents will typically have preferences
  - This is declarative knowledge about the relative value of different states of the world.
  - "I prefer filling disk C: before starting using disk D:"
  - "I prefer to buy on eBay rather than on Amazon"
- Often, the *value* of an action outcome can be quantified (utility functions can often be derived for goal-oriented agents too)
  - This allows the agent to compare the *utility* (or expected utility) of different actions.
- A rational agent is one that maximizes expected utility in its actions.

Other Agent Models

- The one presented is the taxonomy of Russell and Norvig
  - Maja Mataric has provided and interesting alternative taxonomy
- Other models may be useful for specific applications
  - Non-rational agents (e.g., probabilistic or "blind" behaviour", useful for searches in large spaces)
  - Ant-like agents (we will analyse them in the following)
  - Specific architectures for specific types of complex robots (e.g., Sony Asimo has a complex multi-layered knowledge-based goal-oriented architecture)
- In any case the most successful and general-purpose model for goal-oriented agent seems to be the BDI one

The BDI Model

- BDI is a very successful and general model to “think” at software agents
- The agent has
  - Beliefs: the fact he knows about the world (its knowledge)
  - Desires: the goals the agents has to pursue, what he desired to eventually occur
  - Intentions: the current plan of action, what he actually intend to do to satisfy its desires
- BDI agents are usually specified using logic-programming approaches
  - approaches that rather than executing “instructions” tries to manipulate – according to specific rules – the base of knowledge and the base of possible actions, and evaluates an utility functioin to select the intentions based on beliefs
- Of course, DBI agents can be programmed in normal programming languages, but this may be more complicated…

Agent Systems and Languages

- As when developing "normal" software, developing agent software requires specific programming systems
- Either defining a specific "agent-oriented programming language"
  - There are several proposals in that directions
  - However, frankly speaking, no one is really convincing so far, and likely to achieve widespread acceptance and usage
- Or supporting with specific package the development of agent in existing programming languages
  - Logic-based programming languages (e.g., Prolog agents)
  - Object-oriented languages (e.g., Java agents)
- In the case of Java agents
  - Specific classes are provided with which to define agents and their interactions, according to some specific architectural model
- The case of Multiagent Systems, will in addition require proper Agent-oriented Middleware

Agent Examples: The Aglet approach

- Originally produced by IBM
  - Then become open source (the manager is Luca Ferrari, researcher at DISMI-UNIMORE!!!),
- Reactive Agents with State
  - Specifically oriented to network management
  - Perceive network and file systems environment
  - React upon specific events (no specific actions selection)
  - Can autonomously move from node to node (agent mobility)
  - Can interact via message-passing or indirectly via modification of the context of a node

Aglets: Code Example

```
import aglet.*;

public class DispatchingExample extends Aglet
{ private boolean _theRemote = false;

public void onMessage(Message msg)   // react when a message arrives from an aglet
System.out.println(who() + "\'onDispatching()\' is starting..."); pause(); }

public void onArrival() // react when arriving on a node
{ _theRemote = true; System.out.println(who() + "\'onArrival()\' is finishing."); }

// main body of the aglet
public void run() {
if (!_theRemote) { System.out.println(who() + "\'run()\' is starting...");

// access the local AgletContext to get the URL of the node
String host = getAgletContext().getHostingURL().toString();
URL destination = new URL((String)getAgletContext().getProperty("location", host));

// ask the local AgletContext (which also act as naming service) for another Aglet
Aglet ag = getAgletContext().getProxy(“myfriend”);
Ag.sendMessage(“hello how are you?”)
…
```

Agent Examples: the JADE Approach

- Goal-oriented type
  - An agent has a set of behaviors ("actions") that code a sub-task of the agent → similar to objects methods
  - And a state, which represents its current knowledge of the world
  - The goal is not "explicit"
  - The agent can be multithreaded
  - Behaviors can be dynamically added on need
- An agent start with a "setup" behavior that may activate other behaviors
  - One behavior, when activated, executes to completion and can
  - Activate other behaviors
  - Depending on the actual knowledge, and in such a way that the goal may be effectively approach by properly composing the behaviors
- Agents can interact with each other
  - In the forms of "Agent Communication Languages", sort of messages exchanges between agents
  - That can influence their knowledge of the world and their behaviors
- No explicit representation of the environment!!!

JADE: Code Example

```
public class SearchAgent extends Agent {
  // starting behaviour
  protected void setup() {
    System.out.println("Hello. I am "+this.getLocalName()+".");
    this.searchAgents();
  }

// another behavior
  private void searchAgents() {
    DFAgentDescription dfd = new DFAgentDescription();
    SearchConstraints c = new SearchConstraints();
    Agent ag = DFServiceCommunicator.Discover("FriendAgent");
    if (ag==null)
      this.searchanother();
   else
        ACLMessage msg = new ACLMessage("Ciao Ciao")
        ag.sendACLMessage(msg);
  }

// another behavior
  private void searchanother() {
…
}
```

Building Agents with Objects

- In theory, we could also exploit a raw object-system to build agents
  - i.e., to build grown-up objects without making use of special JADE or AGLETS classes
  - So as to make them sort of software agents
- This is of course more difficult
  - **BUT OUTLINES AN IMPORTANT POINT**
- The concept of "agency" is a conceptual concept
  - Whenever we have something that is autonomous, situated, goal-oriented
  - We could call it an agent!!!
- Agents are not a language or a system
- They are **a new way of thinking software!!!**

Agent Applications (1)

- Monitoring and autonomous maintenance operations
  - Anti-spammers and anti-viruses
  - Scheduler for resources
  - Personal digital assistance (e.g. Microsoft agents)
- Control of physical processes
  - Control the functioning of specific production machines
  - Access the sensors of the machine
  - Interfaces with the actuators of the machine
  - Action selection as a “rational” – rather than mechanical or electrical – feedback control loop
- Videogaming
  - Aren’t the characters of modern strategy games software agents after all??

Agent Applications (2)

- Autonomous unmanned vehicles
  - Automatic pilots
  - Self-driving cars
  - Robots
  - MARS Robots

- The specific case of self-driving cars
  - Have the goal of reaching a place (the desire)
  - Have the knowledge about streets (the beliefs)
  - Sense the streets and the traffic conditions
  - Act on brakes, fuel, and directions, to approach the goal
  - At the same time, it has to minimize the danger (utility function)
  - The actual actions (intentions) must be dynamically decided based on current environmental conditions

Agent Applications: However…

- The most interesting applications of autonomous agents are those in which multiple agents interact and concur in a system…

- Multiagent Systems
- With specific additional problems to be faced!