Artificial Life and Multiagent Systems
Prof. Dr. Winfried Kurth
SS 2003

Developing Multiagent Systems with AgentTool

George Radev Georgiev
Erasmus student

Introduction

The project AgentTool is developed by Scott A. DeLoach and Mark Wood at the Department of Electrical and Computer Engineering of Air Force Institute of Technology

AgentTool is an IDE based on Multiagent System Engineering (MaSE) methodology that implements the following objectives for heterogeneous agent management:

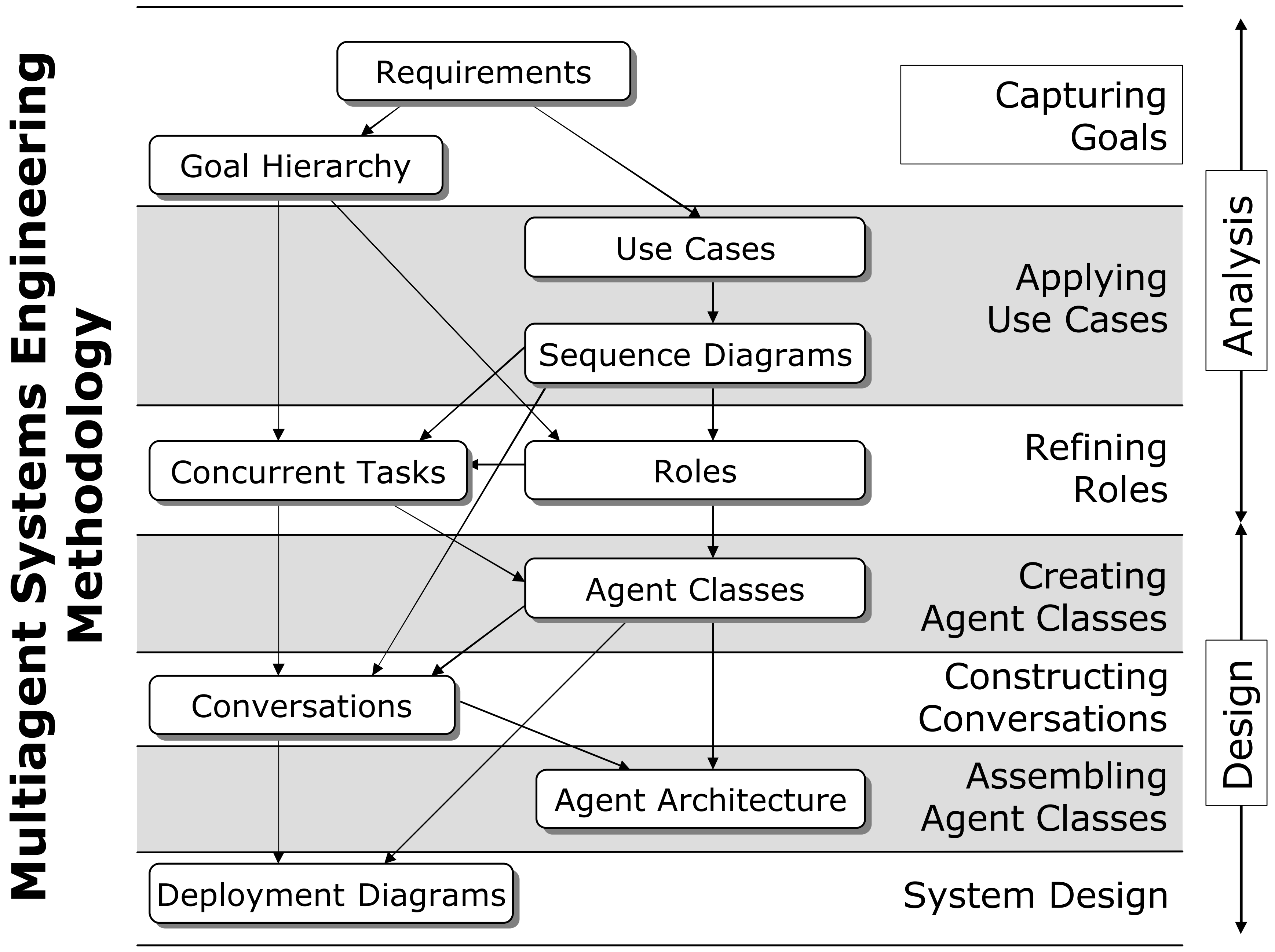
- Analyzing
- Designing
- Developing

Model Abstraction

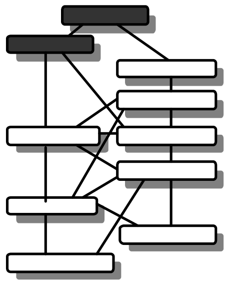
AgentTool is a flexible and powerful agent interaction framework where every agent is a specialization of **Objects**

- **Object** is the fundamental element in the environment;
- All Objects correspond to each other via **conversations**;
- **Role** of an Object is description of the expected functionality;
- **Task** is the decomposition of the role

This abstraction allows us to use object-oriented theory for the specification and design of multiagent systems;
It is also more convenient for analyzing


Multiagent Systems Engineering Methodology
Requirements
Goal Hierarchy
Use Cases
Sequence Diagrams
Concurrent Tasks
Roles
Agent Classes
Conversations
Agent Architecture
Deployment Diagrams
Capturing Goals
Applying Use Cases
Refining Roles
Creating Agent Classes
Constructing Conversations
Assembling Agent Classes
System Design
Analysis
Design

Capturing Goals



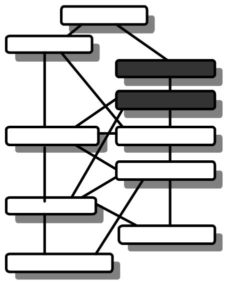
- Two steps of goal capturing:

  identifying goals
  and
  structuring goals

- Creating Goal hierarchy diagram from the system objectives.
- It is necessary that sub-goals satisfy parent goals.

[SAMPLE]

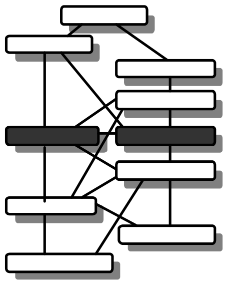
Applying Uses Cases



- This step translates goals into roles and associated tasks.
- Uses Cases depend on system requirements and describe a sequence of events for the system behavior.
- Uses Cases are restructured in sequence diagrams

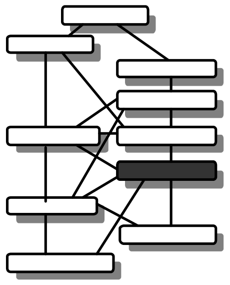
[SAMPLE]

Refining Roles



- The third step in MaSE is making sure that we have identified all the necessary roles and developing the tasks that define role behavior and communication patterns
- Each goal is usually mapped to a single role but it is also useful to combine multiple goals in a single role for convenience and/or efficiency
- Decisions are based on standard software engineering concepts such as functional, temporal, procedural, communicational, sequential cohesion
- Once a role is defined, a task is created to provide a high-level description what role must do

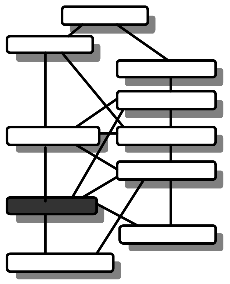
Creating Agent Classes



- Each goal from the hierarchy is associated with a role and with a class that is responsible for it
- Agent classes are roles identified and documented in an Agent Class Diagram

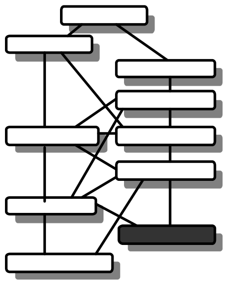
[SAMPLE]

Constructing Conversations



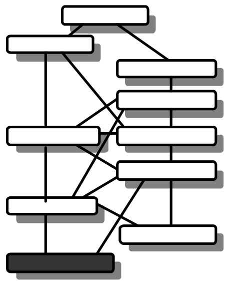
- Agents coordinate their actions via conversations to accomplish individual or common goals
- A conversation defines a coordination protocol between two participating agents in a Communication Class Diagram:
  -> **initiator** and **responder**
- Initiator always sends the first message.
- Similar to concurrent task concept
- This step is often combined with a succeeding one.

Assembling Agent Classes



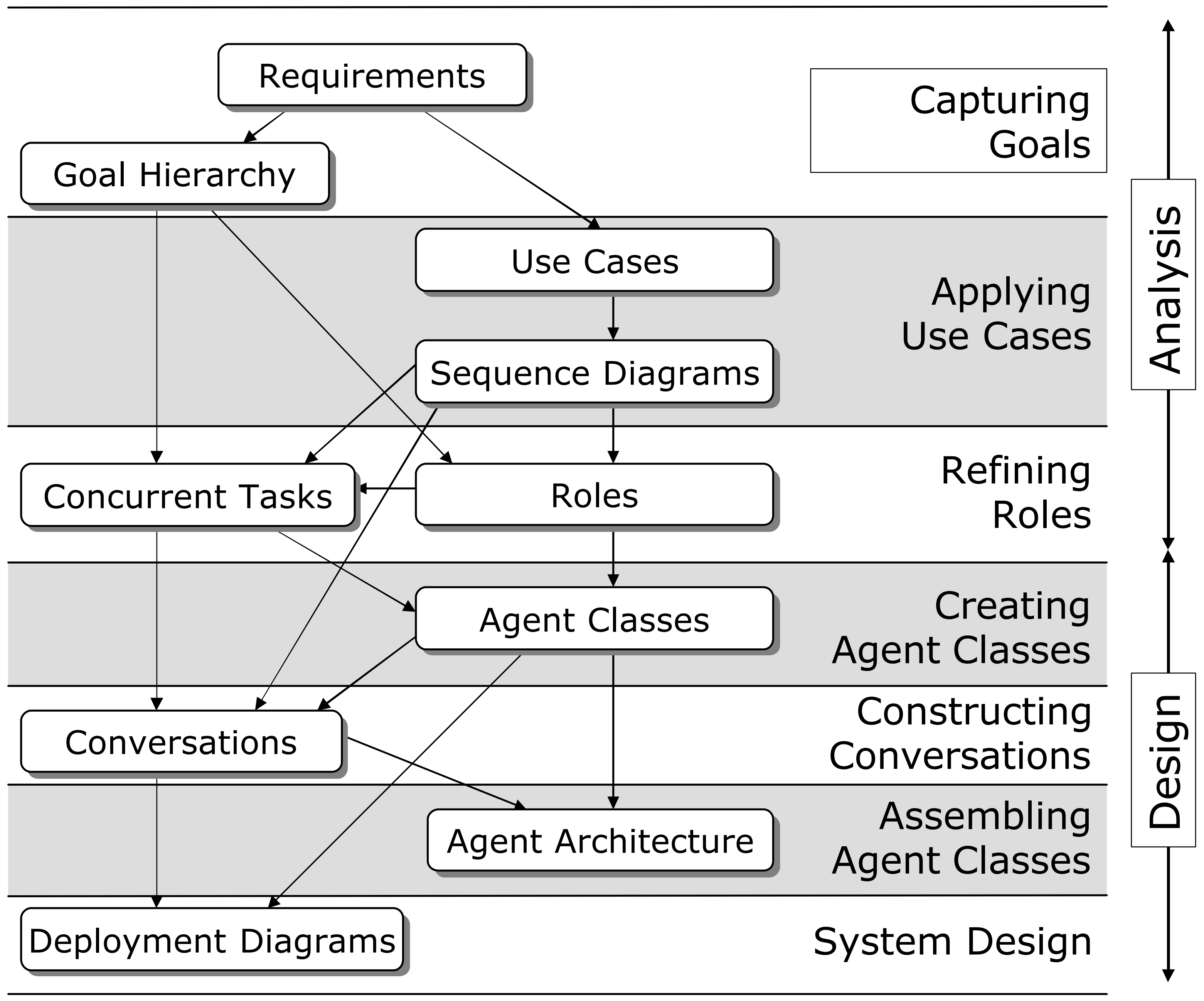
- In this step we implement the internal agent class architecture (in OO terminology – methods) complying with actions specified in the conversations
- All actions specified in the tasks and conversations must be mapped to the functionality of the agent architecture
- Object-oriented theory is very suitable for agent assembling

System Design



- The final step of MaSE defines the configuration of the actual system
- Here the following undefined implementation decisions are made:
  - ✓ programming language
  - ✓ communication framework
  - ✓ system requirements, etc.

Multiagent Systems Engineering Methodology



CROBOTS - Conceptual Scheme

Developed by Tom Poindexter in December, 1985
Illinois State University
http://www.nyx.net/~tpoindex/index.html

CROBOTS is a programming game, where you write a robot control program in C. Your robot's mission is to seek out and destroy other robots, each running different programs.

Variants of CROBOTS:

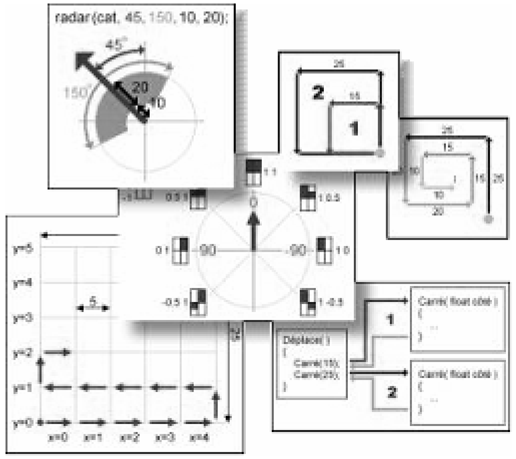
- C++ Robots
- CRobots32
- TCLRobots
- PRobots
- PCRobots
- AT-Robots
- Giavamachia - Java Version
- CeeBot - http://www.epsitec.ch/

CROBOTS - API

All robots are executed in a cooperative non-preemptive multitasking environment in the same battlefield

Common functions:

```
void movement(int speed, int angle);
void get_local_map(char far *buffer);
robotID scan(int angle,int res,int *range);
bool shoot(int angle,int range);
int get_shell_status(void);
void getxy(int *x,int *y);
int damage(void);
int speed(void);
```



Questions