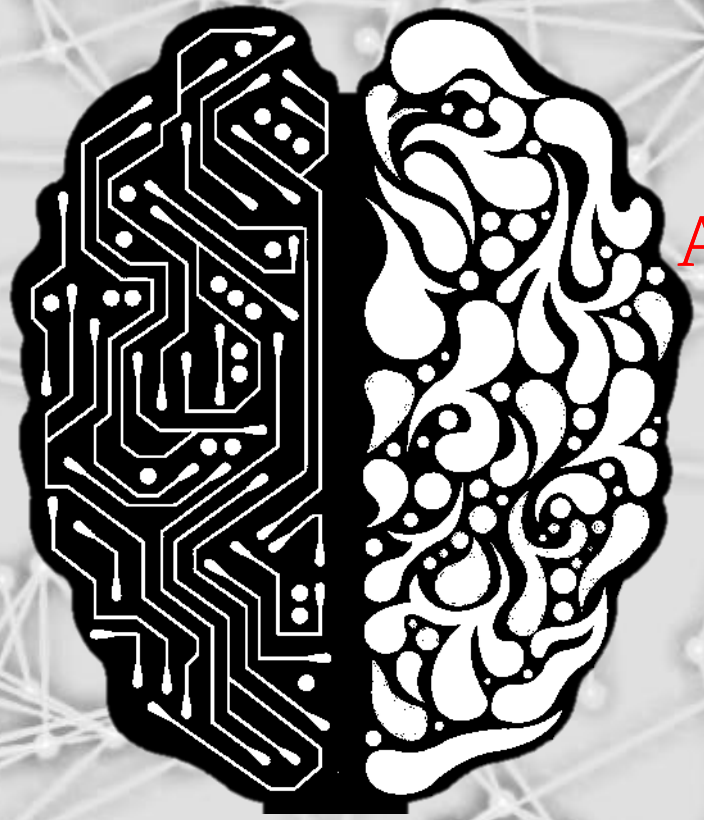


Multi-Agent Systems

Lecture V

- Dr. Nestor Velasco Bermeo,

- Researcher CONSUS (Crop OptimisatioN through Sensing, Understanding & viSualisation),
- School of Computer Science
- University College Dublin (UCD)

Commitment Management

- It is formed from a number of sub-processes which implement a set of strategies that specify how an agent:
  - Adopts new commitments.
  - Maintains its existing commitments.
  - Refines commitments to plans into additional commitments.
  - Realises commitments to action.
  - Handles failed commitments.
- A Commitment Management Strategy is a specific set of strategies that can be employed by an agent.

  e.g. blind commitment, single-minded commitment, social-minded commitment.
- The default strategy in Agent Factory is single-minded commitment.
- An agent maintains a commitment so long as it believes it is still acheivable.

Commitment Maintenance

- Commitments are maintained using a maintenance condition that is associated with each commitment.

BELIEF(**has**(**?food**)) -> COMMIT(**Self**, **Now**, BELIEF(**true**), **eat**(**?food**))

- This condition outlines what must remain true for the agent to keep the commitment (like terms and conditions and a contract).
  - **In the above example, the maintenance condition will always be true. This is sometimes known as blind commitment.**
  - **The maintenance condition is evaluated at each time point.**
  - **If the condition becomes false at any time point, then the commitment is said to have "failed".**

Key AF-APL Agent Concepts

- Agent = Mental State + Commitment Rules + Embodiment Config.
- Mental State:
  - Beliefs. **Subjective knowledge about the current state of the environment.**
  - Commitments. **Mental contract describing which activity, at what time, for whom, and under what conditions.**
    - Activities may be either **primitive actions** or **plans** (SEQ, OR, PAR).
- Commitment Rules:
  - **Map situations (possible environment states) to commitments that should be adopted should the situation arise.**
- Embodiment Configuration
  - Perceptors. **Computational units that convert raw data into beliefs.**
  - Actuators. **Computational units that define how to realise primitive actions.**

Representing Activities

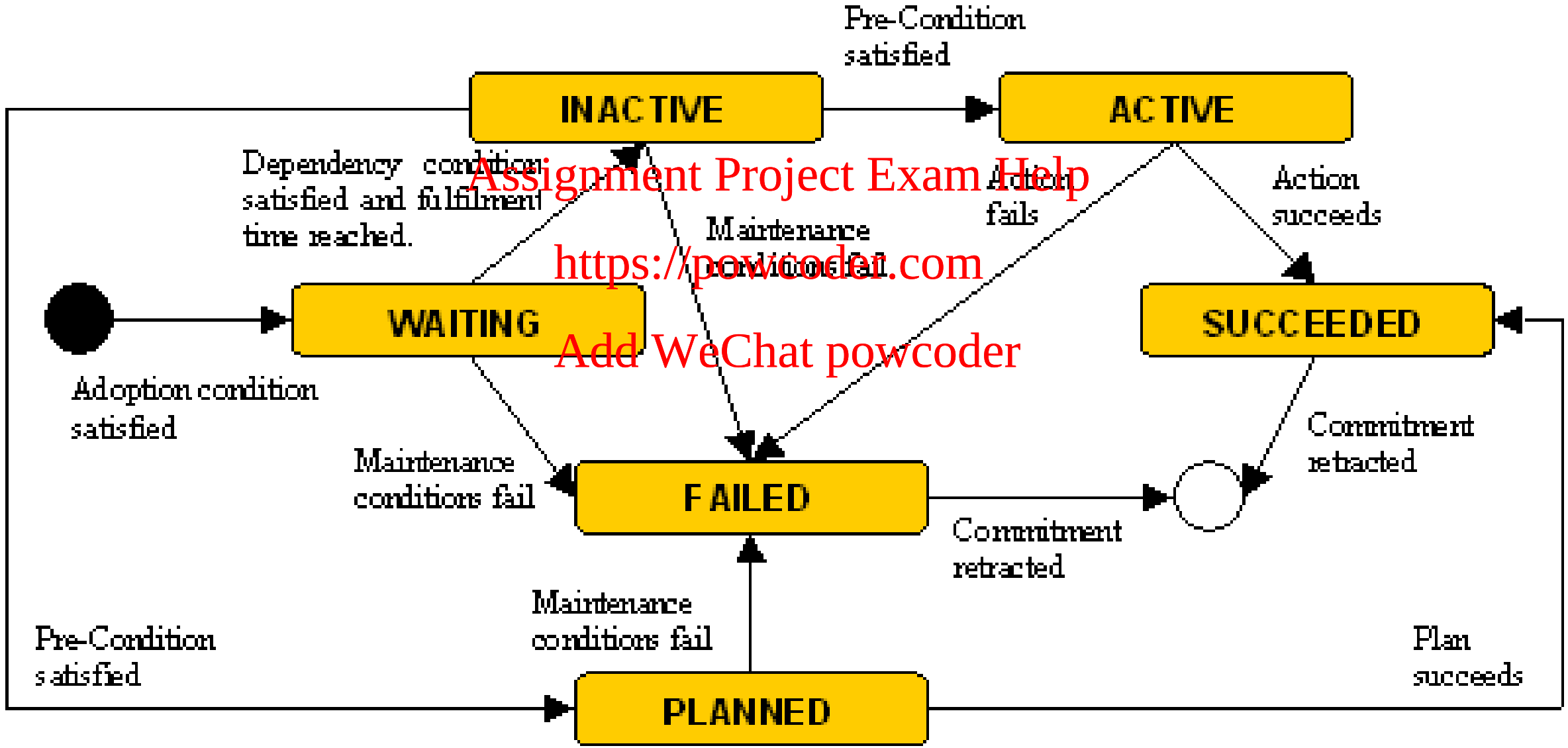
- Activities describe what the agent can do:
  - **Actions.** Primitive abilities that are directly executable by the agent.
  - **Plans.** A recipe that consists of a partially ordered set of activities.
- AF-APL supports the definition of actions and explicit plans.
  - Actions are defined in the constructor of the associated actuator unit. **The definition consists of:**

    Unique identifier **eat(?food),**
    Pre-condition **BELIEF(has(?food))**
    Post-condition **(not used).**

  - Explicit plans are defined within the activity field of a commitment. **They take the form of a plan operator (SEQ or PAR for AF-APL) together with a list of activities that may be either additional plan operators or actions.**

    SEQ(**PAR**(boilWater, addCoffee), pourWater, **PAR**(addSugar, addMilk))

Commitment States



Commitment Realisation & Refinement

- At some point in time, the agent will try to fulfill its commitments.
  - **Commitments to action are fulfilled through** actuator activation.
    - The agent finds the corresponding actuator and activates it.
    - If not corresponding actuator exists, then the commitment fails.
  - **Commitments to plans are fulfilled through** commitment refinement.
    - The agent adopts a set of secondary commitments that correspond to the activities specified in the plan.
    - Plan operators may be used to place an order on the achievement of these commitments.
- The set of commitments adopted when fulfilling a primary commitment to be a commitment structure.

Commitment Structure Example

Gregory,
2005/01/20-8:00:00,
BELIEF(true),
SEQ(dcA, dcB),
INACTIVE

Commitment Adoption

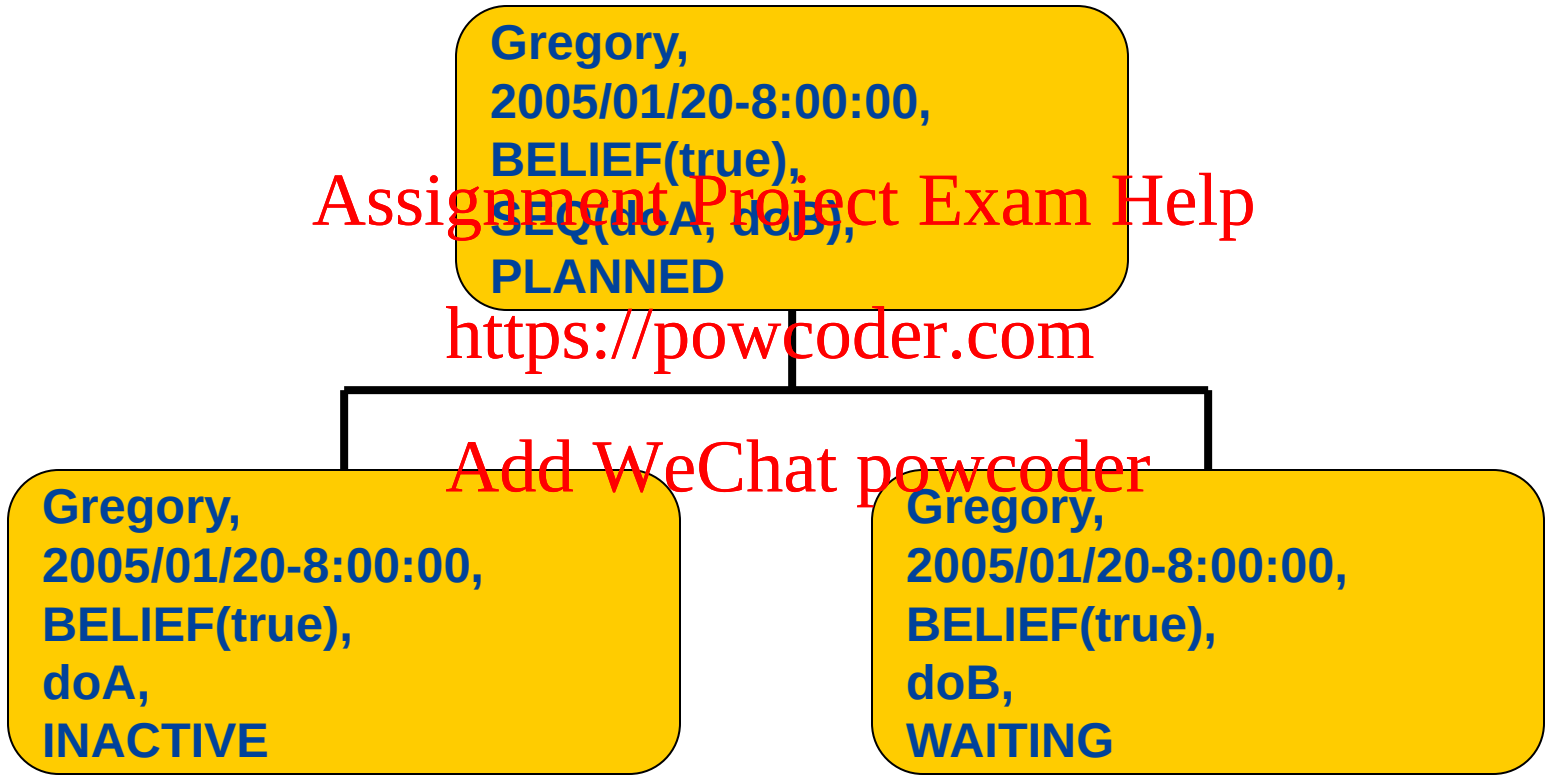
- Commitments are adopted as a result of the triggering of Commitment Rules.
- A commitment rule defines a situation in which the agent should adopt a commitment.

BELIEF**(has(?food))** => COMMIT**(Self, Now, BELIEF(true), eat(?food))**

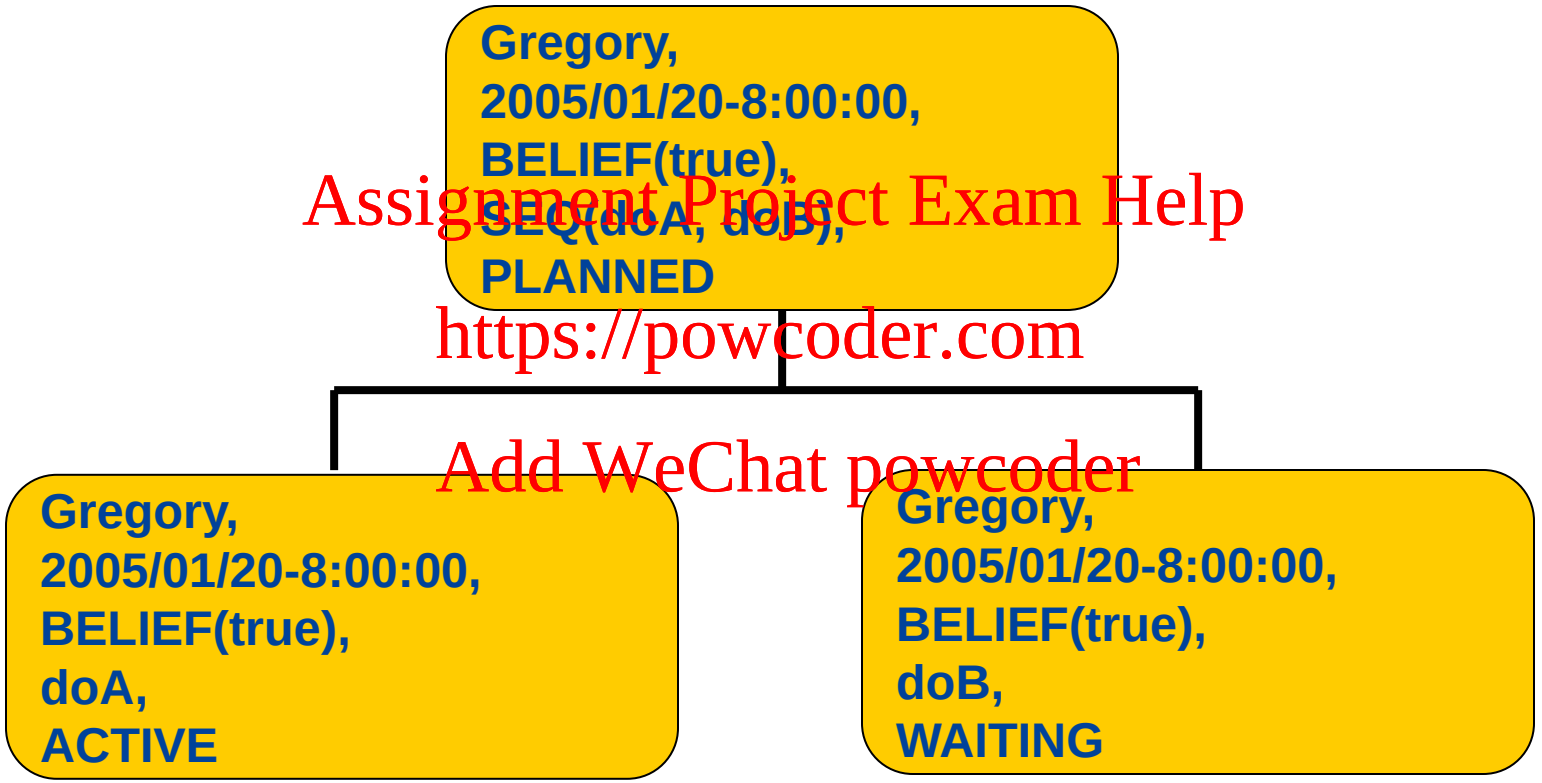
- Each of the commitment rules is checked during each iteration of the AF-APL interpreter.

  - **If the situation (left-hand side) of any rule is evaluated to true, then the rule is said to have been** triggered**.**
  - **Whenever a rule is triggered, there exists (at least one) set of variable bindings.**
  - **Each set of bindings is applied to the commitment construct on the right-hand side of the commitment rule, and the corresponding** primary commitment **is adopted by the agent.**

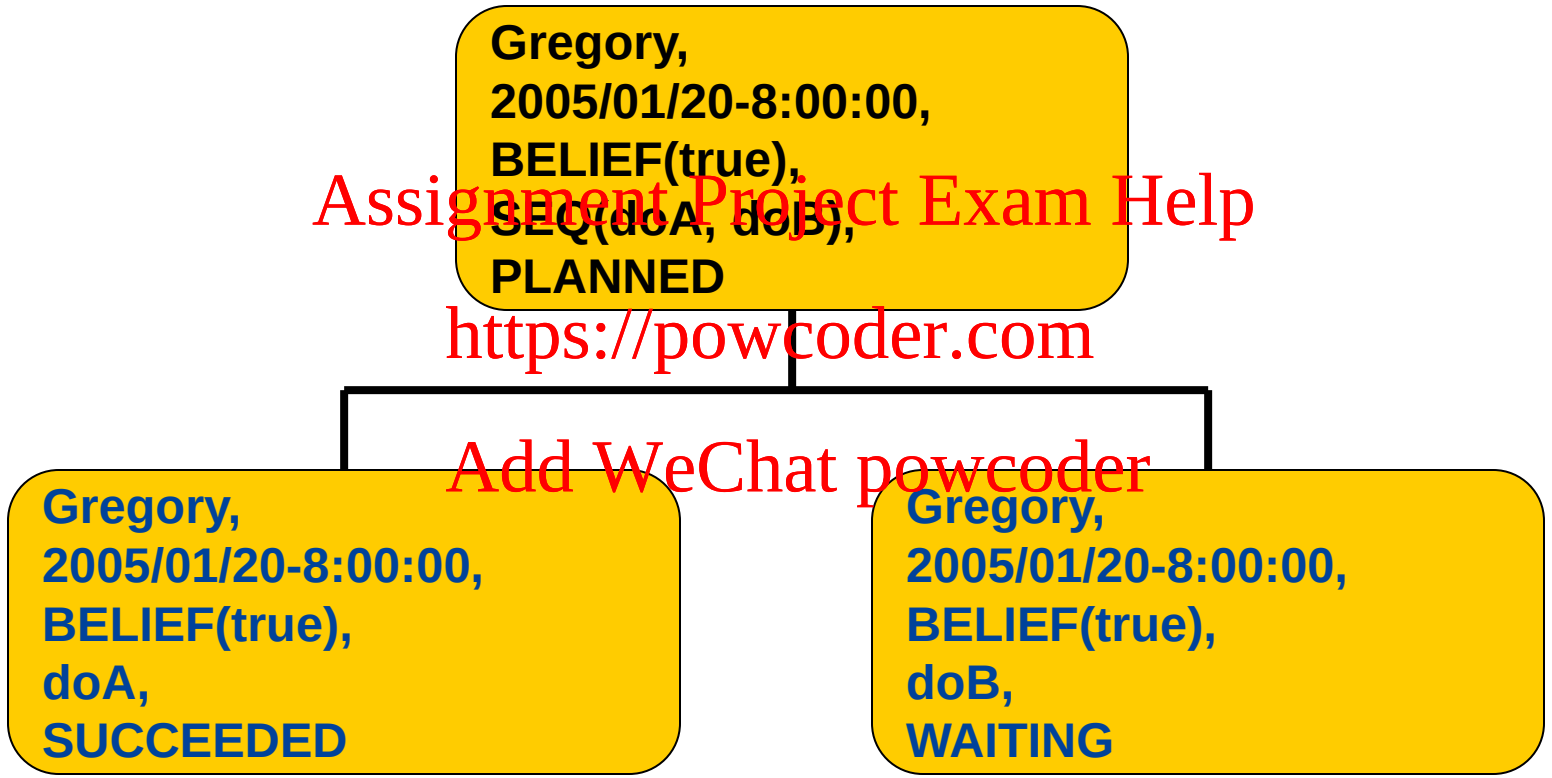
Commitment Structure Example



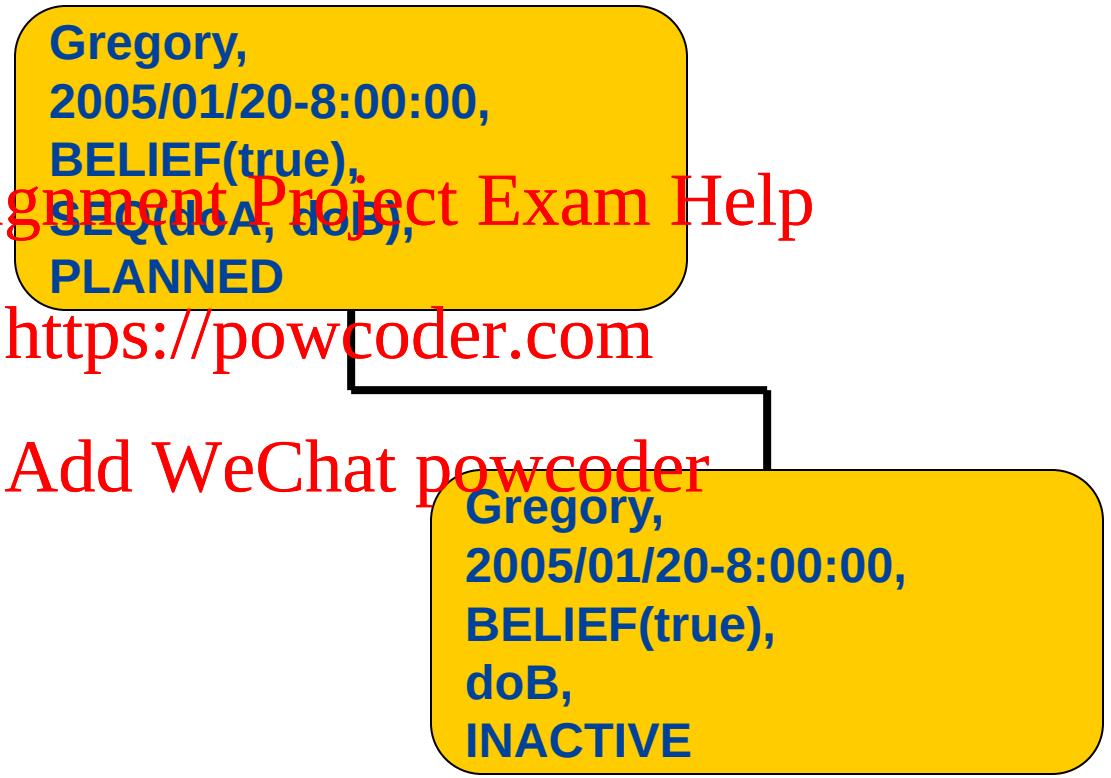
Commitment Structure Example



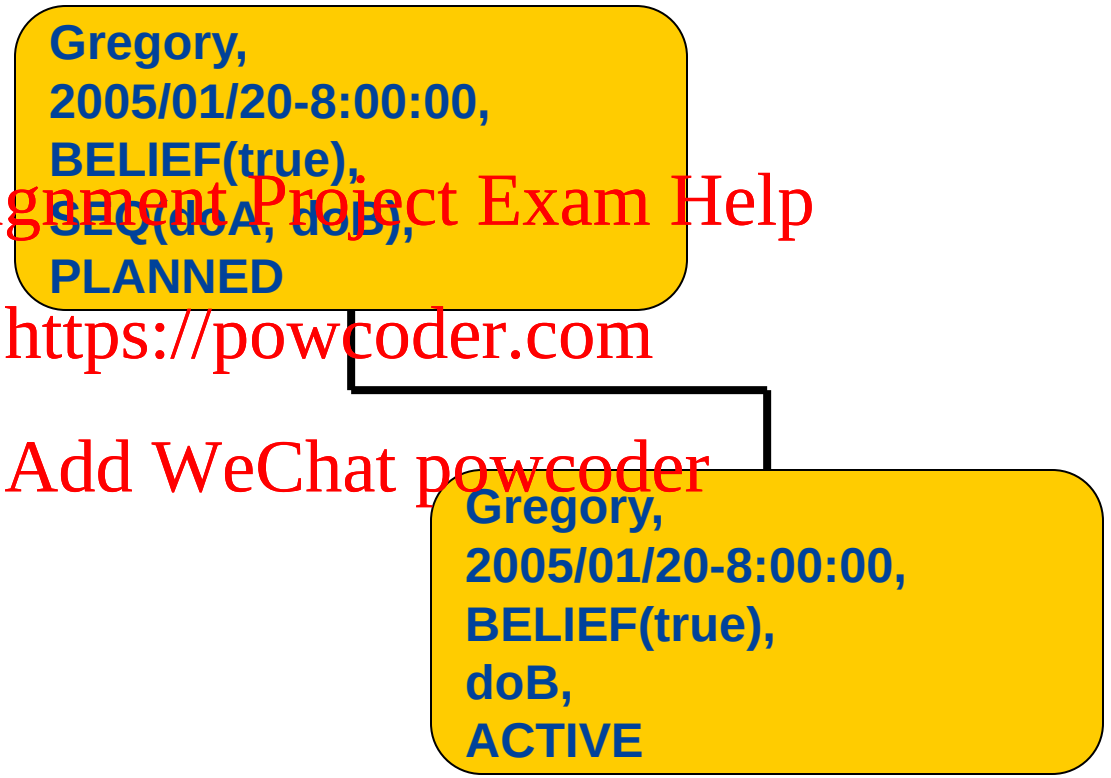
Commitment Structure Example



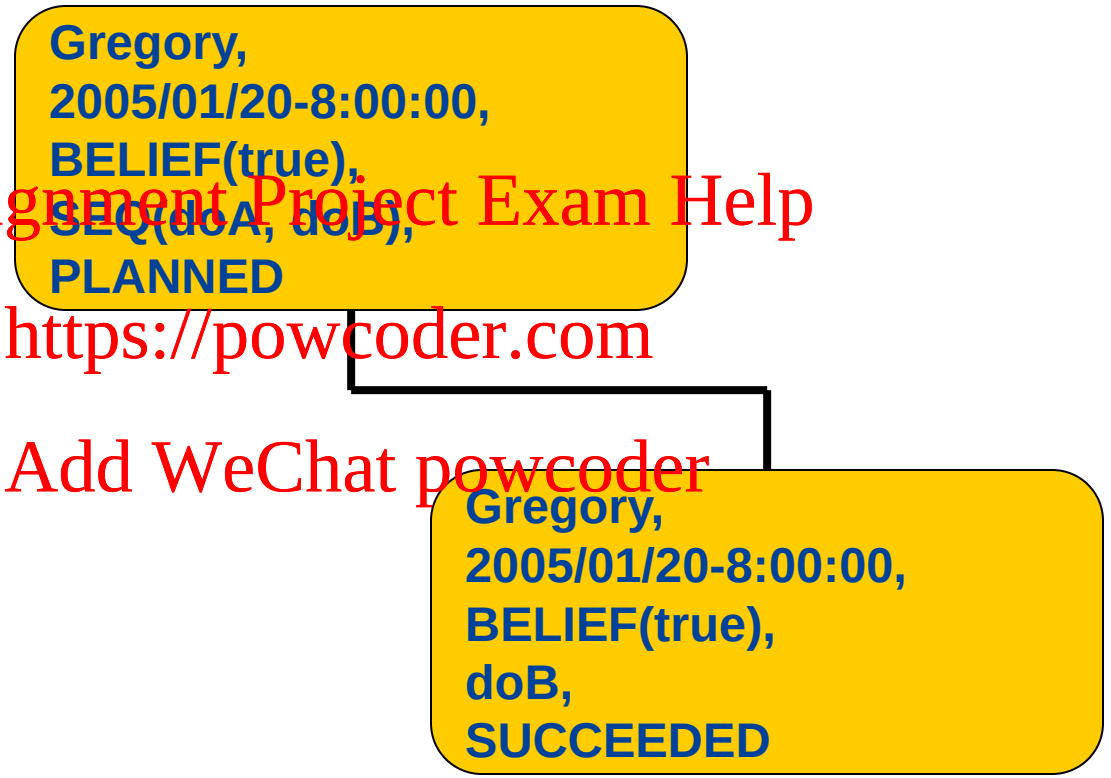
Commitment Structure Example



Commitment Structure Example



Commitment Structure Example



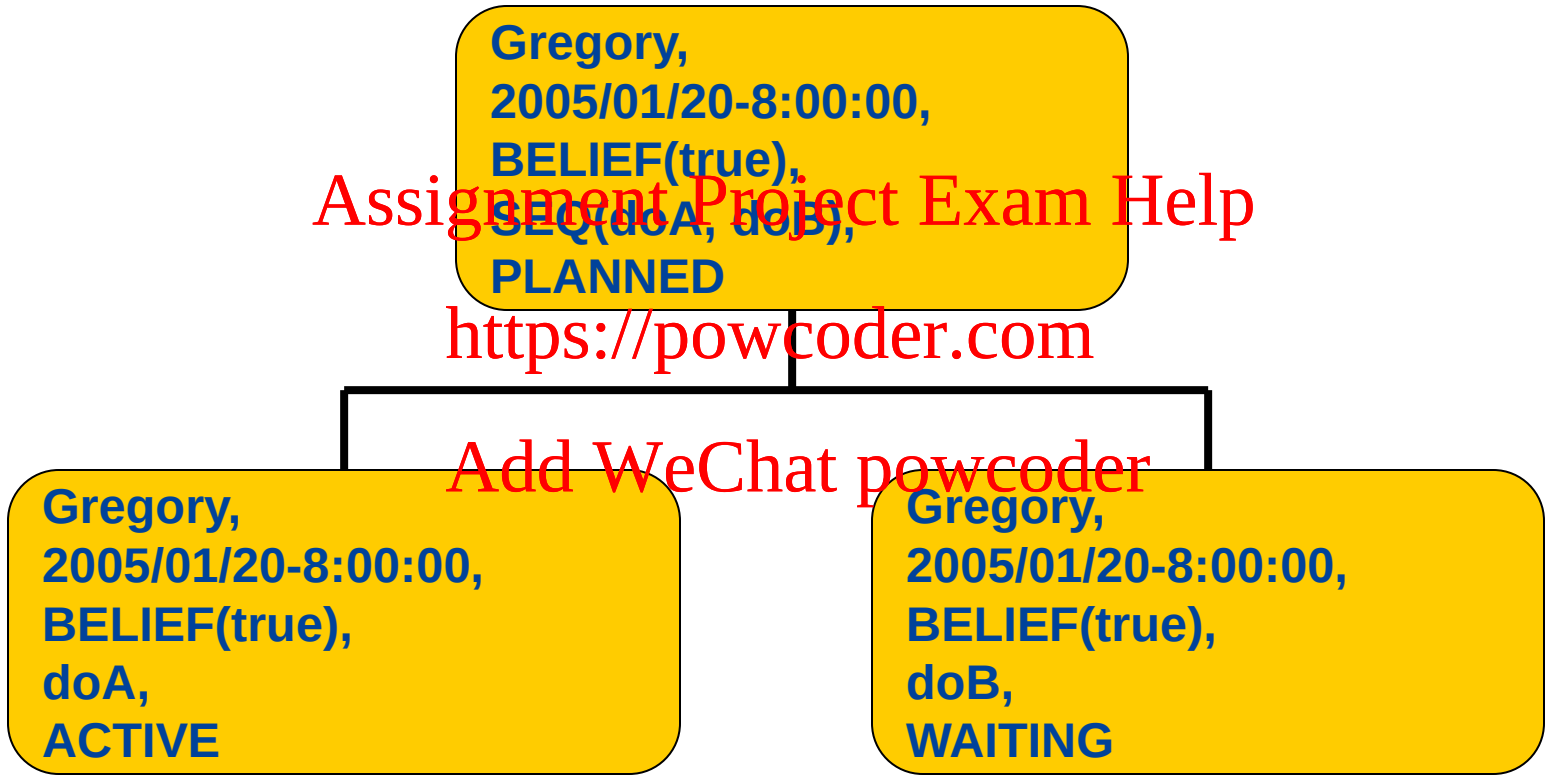
Commitment Structure Example

Gregory,
2005/01/20-8:00:00,
BELIEF(true),
SEQ(dOA, dOB),
SUCCEEDED

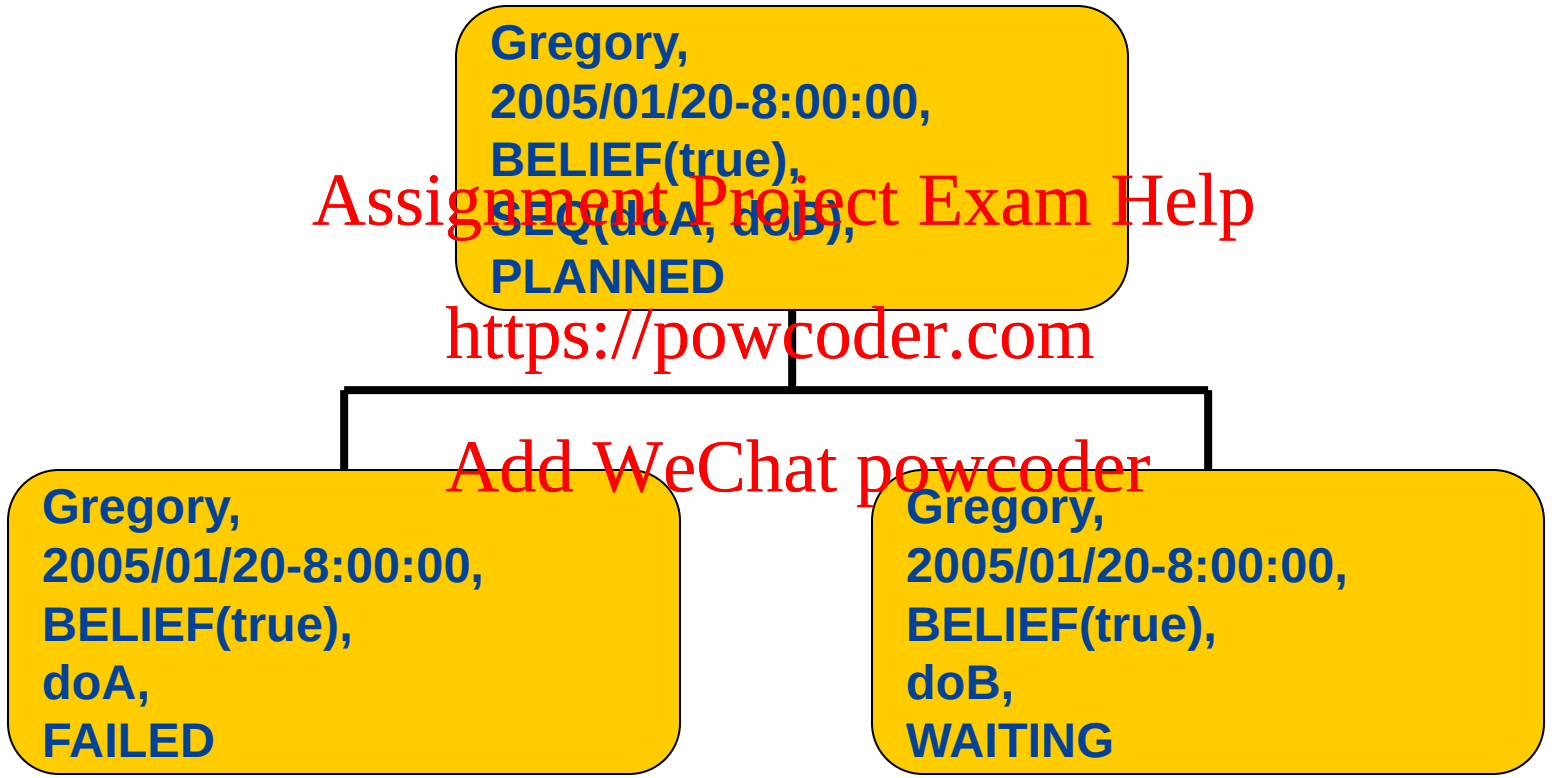
Commitment Failure Handling

- If any commitment fails, the failure handling strategy defines how the agent should respond to the failure.
  - In AF-APL, the strategy is simple:
    - The failure of a secondary commitment is passed to the parent commitment. The impact of this failure is assessed with respect to the parent commitment.
    - The failure of a commitment that has children causes the children to fail. There is no assessment here!
  - During the failure handling process, this strategy is applied recursively through the commitment structure.
    - This recursive process, while potentially computationally expensive, is essential to ensure the agent does not continue to try and fulfil commitments that are now redundant.

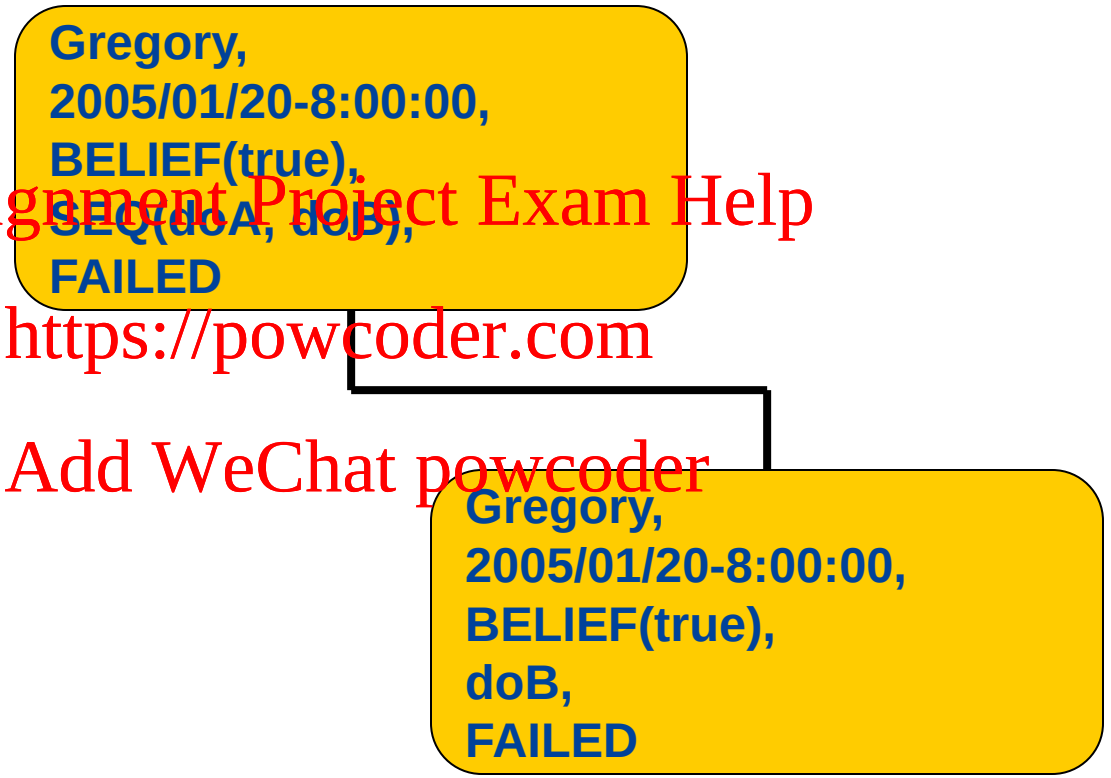
Failure Example



Failure Example



Failure Example



Agent Factory in Context I

- A number of other Agent Development Tools exist:
  - **LEAP (LEAP Consortium)**. Integration of JADE and ZEUS that is compliant with J2ME.
  - **JADE (TILAB)**. FIPA-compliant Java API that supports the fabrication of reactive agents.
  - **ZEUS (BT Labs)**. A graphical toolkit for creating deliberative agent designs, which when completed, are compiled into Java code, customised and finally, executed.
  - **JACK (Agent-Oriented Software)**. Extends Java with agent-based concepts. JACK code is compiled into Java code and executed.
  - **FIPA-OS (Emorphia)**. The first FIPA-compliant agent platform. Similar to JADE.

Agent Factory in Context II

	AF	LEAP	JACK	ZEUS	JADE	FIPA-OS
BDI	√	√	√	√		
Mobility	√	√			√	√
White Pages	√	√		√	√	√
Yellow Pages	√	√		√	√	√
FIPA Compliance	√	√		√	√	√
Fabrication Mode	*Design*	*Instance*	*Design*	*Instance*	*Design*	*Design*
Inheritance	√		√		√	√
Construction	*Graphical*	*Graphical*	*Graphical*	*Graphical*	*None*	*None*
Visualization	*Graphical*	*Graphical*	*None*	*Graphical*	*None*	*None*
Integrated Methodology	√	√		√		

Lecture V Learning Objectives

- Review the characteristics and elements of Agent Oriented Programming and Object Oriented Programming
- Review the differences between An Agent and an Object
- Understand the elements and characteristics of an Agent Programming Language
- Understand how Belief Management occurs on a MAS and the temporality of Beliefs
- Understand and identify the different Commitment States

Things to Do!

Agent Oriented Programming

- de Moraes Batista, A. F., dos Passos Alves, B., Kobayashi, G., Marietto, M. D. G. B., de Castro, S., Ruas, T. L., & Botelho, W. T. (2011). *Principles of agent-oriented programming*. INTECH Open Access Publisher.

AgentFactory:

- Russell, S., Jordan, H., O'Hare, G. M., & Collier, R. W. (2011, October). Agent factory: a framework for prototyping logic-based AOP languages. In *German Conference on Multiagent System Technologies* (pp. 125-136). Springer, Berlin, Heidelberg.

- Collier, R., & O'Hare, G. M. (2009). Modeling and Programming by Commitment Rules in Agent Factory. In *Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches* (pp. 393-421). IGI Global.
- Ross, R., Collier, R. W., & O'Hare, G. (2004). Af-apl: Bridging principles & practices in agent oriented languages. Programming Multi-Agent Systems. *Lecture Notes in Computer Science (LNAI)*, *3346*.

Things to Do!

- **AgentFactory** https://sourceforge.net/projects/agentfactory/files/
- **JAVA Agent DEvelopment Framework (JADE)** https://jade.tilab.com/
- **ZEUS**

  Nwana, H. S., Ndumu, D. T., Lee, L. C., & Collis, J. C. (1999). ZEUS: a toolkit for building distributed multiagent systems. *Applied Artificial Intelligence*, *13*(1-2), 129-185.
- **JACK Intelligent Agents** http://aosgrp.com/products/jack/
- **FIPA-OS** http://fipa-os.sourceforge.net/index.htm