

# Using JACK Intelligent Agents™ to enable code reuse in Agent Systems

Michael Coburn

michael.coburn@agent-software.com

**Agent Oriented Software Pty Ltd.,**  
221 Bouverie Street, Carlton, VIC 3053.

AUSTRALIA

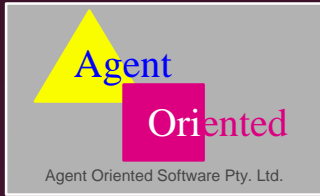
<http://www.agent-software.com>

# Contents

- Introduction.
- Background: About agent oriented programming.
- The problem: Scalability, maintainability and extensibility.
- The solution: Capabilities!
- The BDI model and how it is implemented in JACK.
- Capabilities explained.
- Future directions - what do capabilities buy us?

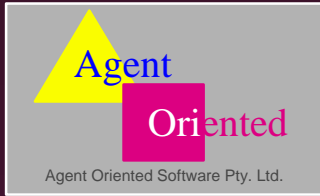
# Introduction

- Agent systems based on the Belief Desire and Intentions (BDI) model are becoming increasingly common.
- As the number of agents in such systems increases the number of components required can grow quickly.
- Many of these components are similar yet are built over and over again for specific agent instances.



## Introduction (cont.)

- The resulting code can be hard to understand, difficult to maintain and can suffer in terms of robustness and reliability.
- Capabilities, a new feature in JACK Intelligent Agents, can help solve these problems and allow extensions to agent based systems that are easier to implement and easier to understand.



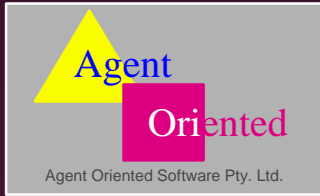
# Software engineering with intelligent agents

## Intelligent Agents:

- autonomous, trusted and cooperative entities showing flexible behaviour

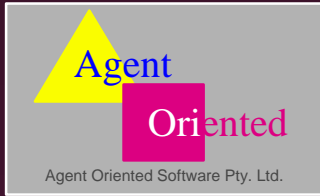
## Advantages:

- agents offer a unifying view of software;
- new encapsulation frameworks to assist developers;
- builds on work in artificial intelligence, databases, networks and software engineering.



# Using an Agent Framework

- Understanding the computation principle that the framework implements
- Understanding procedural v.s. functional v.s. object-oriented programming
- And considering agents v.s. “non-agents”
  - ❖ interfaces
  - ❖ databases
  - ❖ legacy systems



# Where are Agent Oriented systems useful?

- Performing complex processes in complex environments, for example:
  - ❖ coordination of logistic processes in companies with hundreds of transport vehicles;
  - ❖ management of information through a hospital - sharing knowledge & control among interest groups - doctors, nurses, pathologists, administrators; or
  - ❖ air traffic management.
- Agent-oriented solutions bring:
  - ❖ **autonomy**, with a balance between reactivity (event-driven) and proactivity (goal-driven).
  - ❖ **robustness**, perhaps with a performance penalty.
  - ❖ **interoperability**, enabling heterogeneous systems to work together & managing the legacy software problem.

# The Challenge

## Scalability:

- Most solutions to real world problems are complex and involve many agents;
- As Agent Oriented Design is still in its infancy (and human nature hasn't changed) these systems “evolve” in a very “organic” fashion;
- Resulting solutions often contain a lot of redundant code and have poorly defined conceptual divides between different aspects of an agents' functionality.

# The Challenge (cont.)

## Maintainability:

- Avoiding convoluted code that is difficult to understand.
- Identifying where a “capability” that an agent has begins and ends.
- Avoid making changes to code being an onerous task.

# The Challenge (cont.)

## Extensibility:

- Having a too “fine-grained” view of an agents’ functionality makes it difficult to incorporate new concepts into the agent paradigm.
- This applies especially to formalisms for teams and roles within teams.

# Capabilities

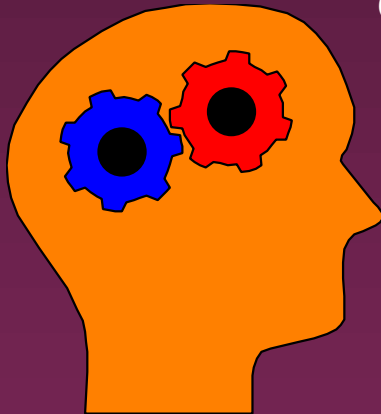
**Capabilities** are a new concept that:

- ❖ allow agent elements to be **structured**;
- ❖ provide **encapsulation** of functionality;
- ❖ **declare interactions** between encapsulated functionalities; and
- ❖ introduce **sound software engineering practice** to agent programming

# The BDI Agent Model

Human  Belief, Desire, Intentions Agent

*Beliefs - perceived understanding of the world*

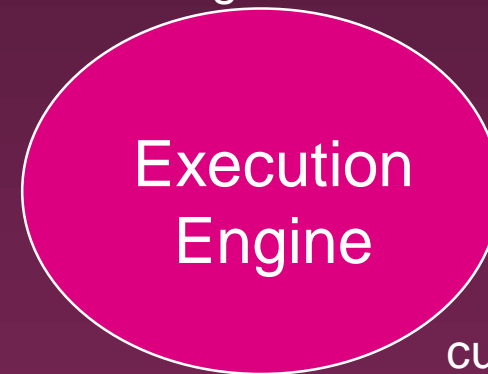


*Accumulated experience and behaviours*

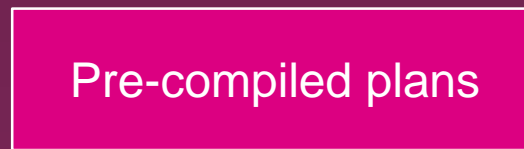
*Goals or desires*

**Beliefs** - database of perceived world knowledge

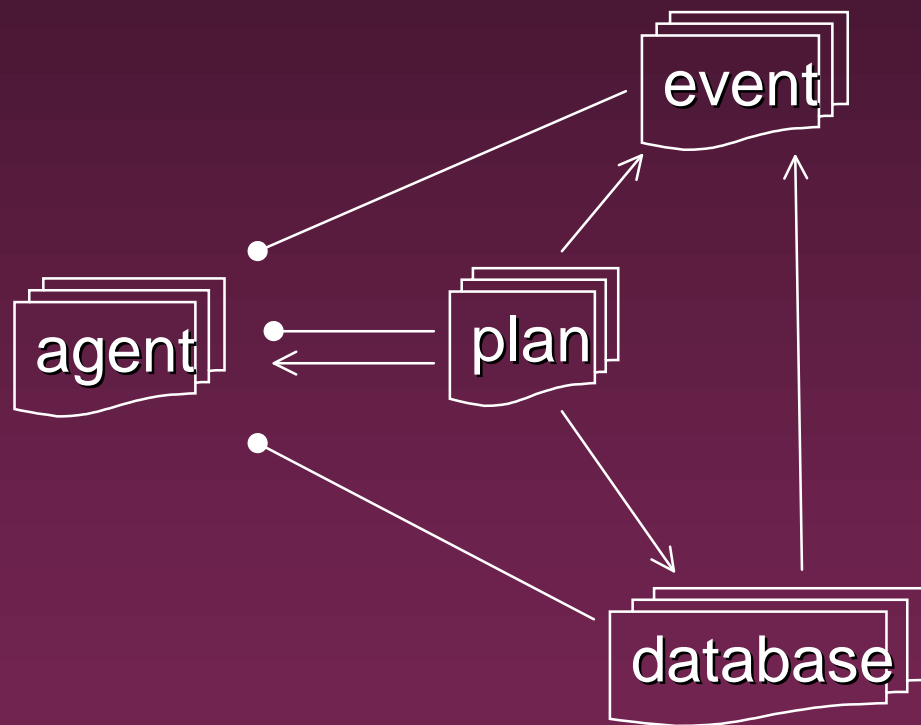
Goals or **desires**



**Intentions** - currently executing plans



# BDI Programming Concepts



● — “has”  
—> “uses”

# JACK BDI Execution

databases

*assert*  
*retract*

**beliefs**

events

*post*

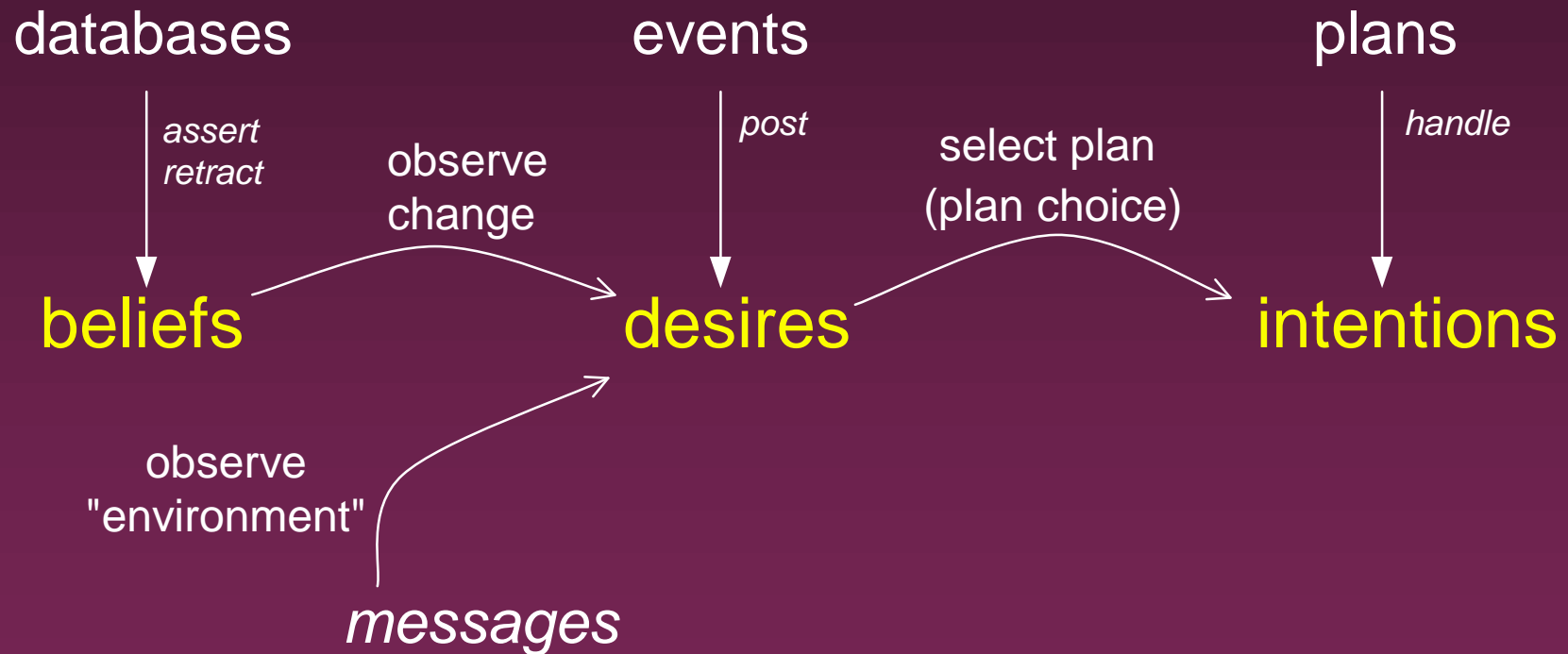
**desires**

plans

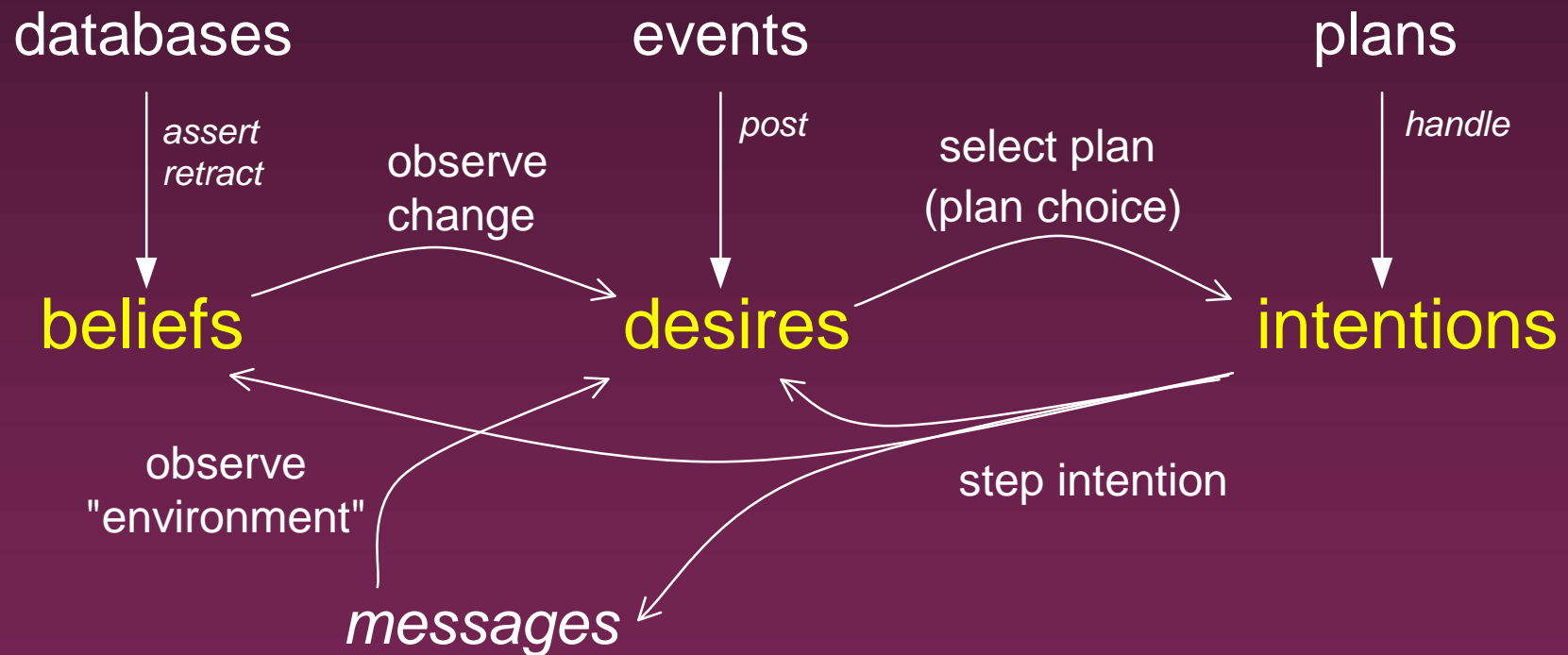
*handle*

**intentions**

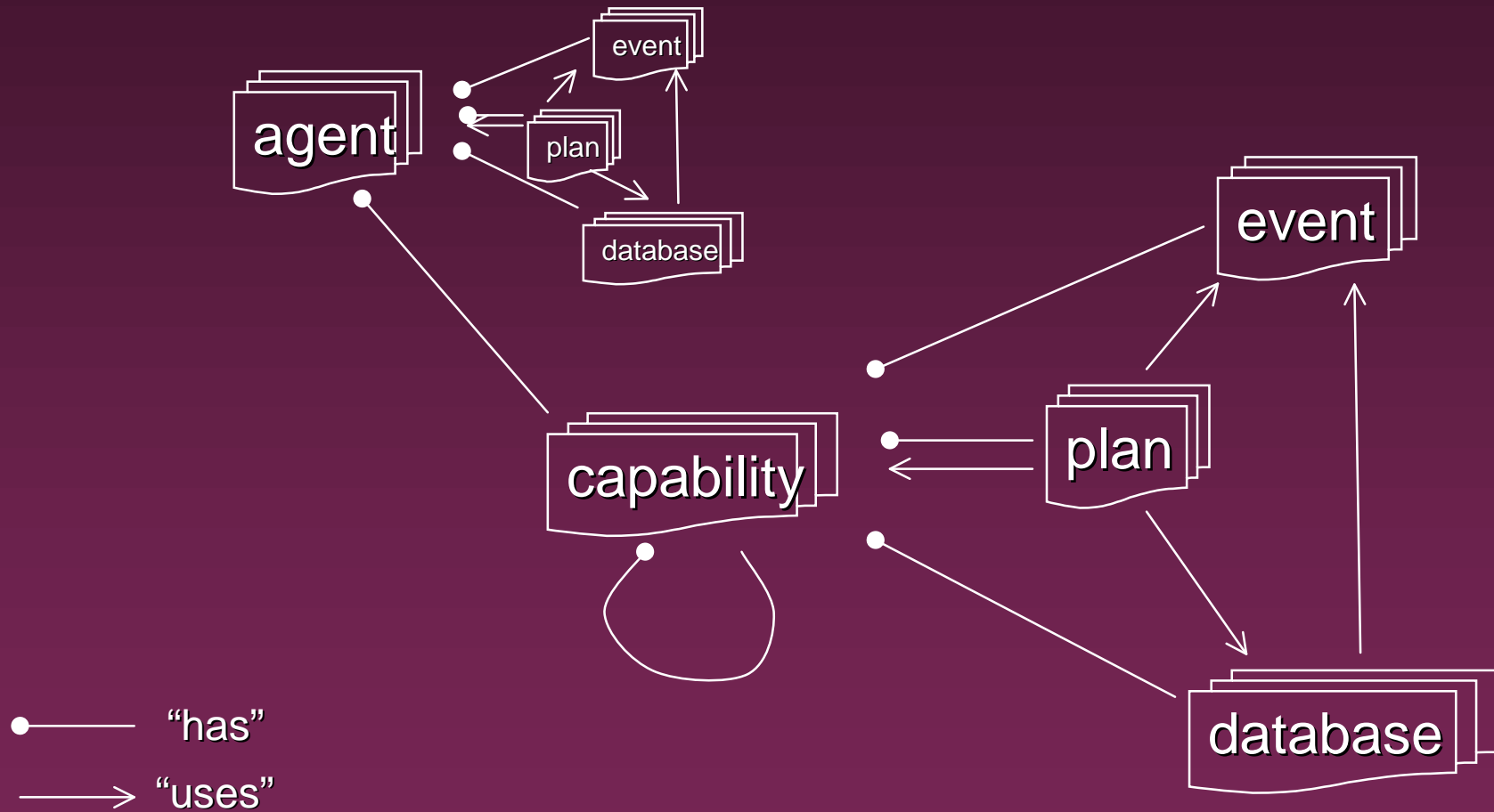
# JACK BDI Execution

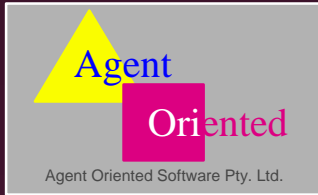


# JACK BDI Execution



# JACK Capability Concept



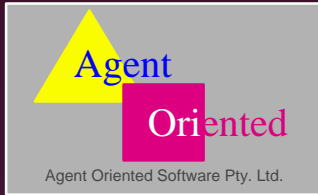


# An Agent in JACK

```
agent ClientBidder extends Agent {
    #handles event NegotiationMessage;
    #handles event Acknowledge;
    #handles event NegotiationGoal;
    #handles event SubmitBid;

    #posts event SubmitBid;
    #posts event NegotiationMessage;
    #posts event Acknowledge;

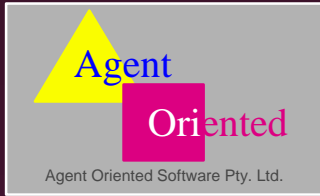
    #private database BiddingDB bidding();
    #private database NegotiationDB negotiation();
    #private database WinnerDB winner();
    #private database BidderDB bidder();
    #private database BidsDB bids();
}
```



# An Agent in JACK (2)

(Agent ClientBidder continued...)

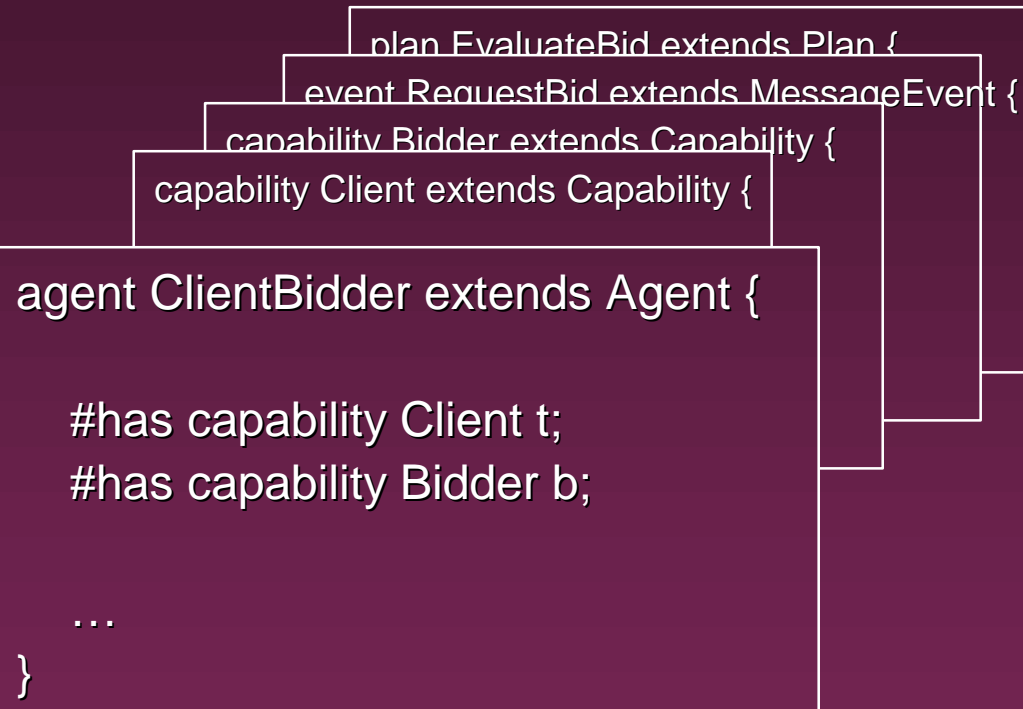
```
#uses plan NegotiationMessPlan;  
#uses plan BiddingPlan;  
#uses plan AcknowledgePlan;  
#uses plan NegotiationGoalPlan;  
#uses plan SubmitBidPlan;  
#uses plan AnalyseBids;  
}
```



# Contrasting with Capabilities

```
agent ClientBidder extends Agent {  
  
    #has capability Client client;  
    #has capability Bidder bidder;  
  
    ...  
}
```

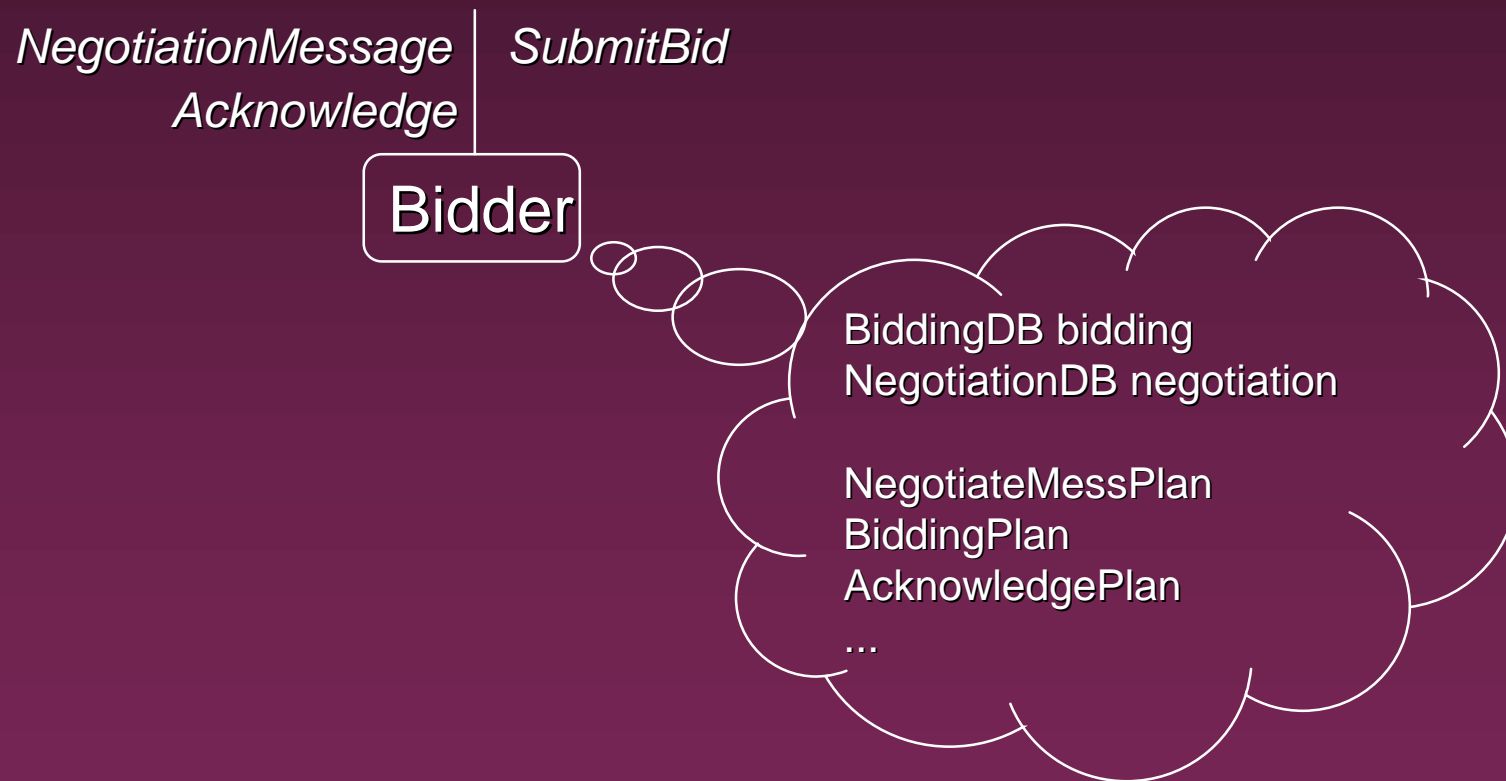
# Programming Elements



# Bidding Example: Client



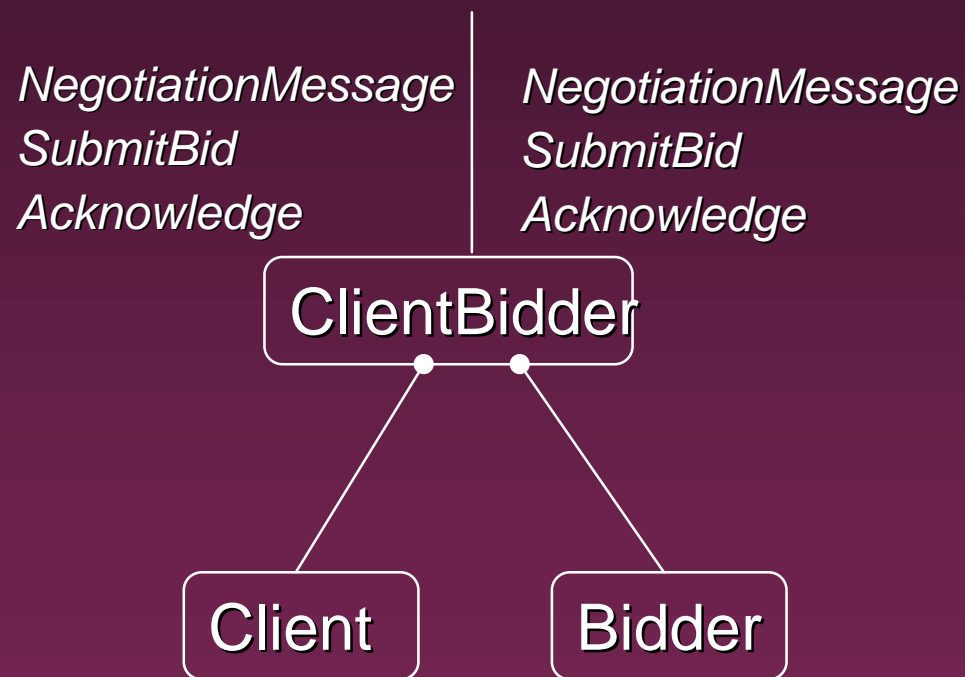
# Bidding Example: Bidder



# Bidder Capability

```
capability Bidder extends Capability {  
    #handles event NegotiationMessage;  
    #sends event SubmitBid;  
    #handles event Acknowledge;  
  
    #private database BiddingDB bidding();  
    #private database NegotiationDB negotiation();  
  
    #uses plan NegotiationMessPlan;  
    #uses plan BiddingPlan;  
    #uses plan AcknowledgePlan;  
}
```

# ClientBidder Capabilities



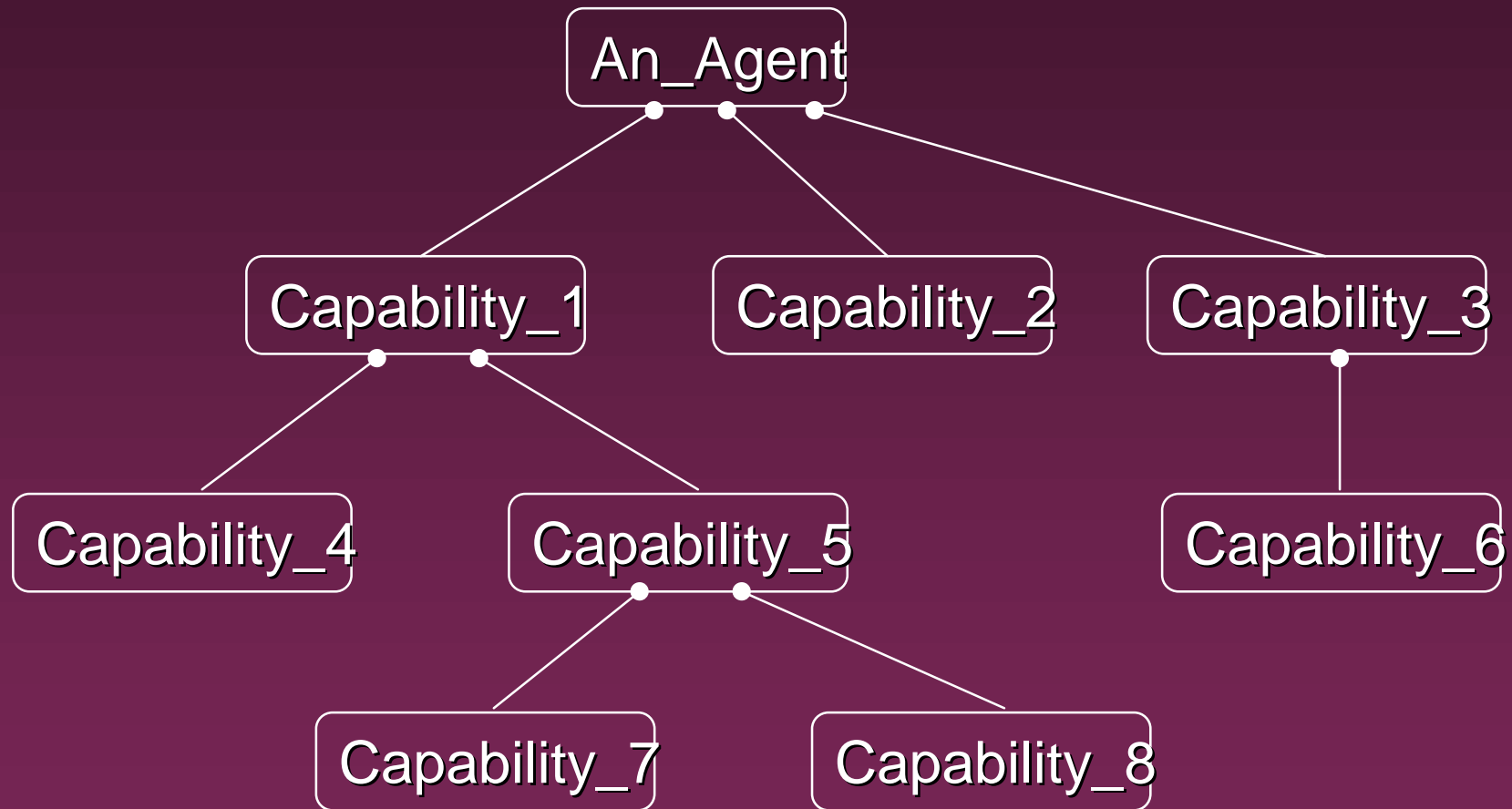
# Capability Type Definition

```
capability <CapType> extends Capability {  
    external events for this capability;  
    exported belief structures;  
    imported belief structures;  
    internal events for this capability;  
    private belief structures;  
    plans of this capability;  
}
```

# Client Capability

```
capability Client extends Capability {  
    #sends event NegotiationMessage;  
    #handles event NegotiationGoal;  
    #handles event SubmitBid;  
    #sends event Acknowledge;  
  
    #private database WinnerDB winner();  
    #private database BidderDB bidder();  
    #private database BidsDB bids();  
  
    #uses plan NegotiationGoalPlan;  
    #uses plan SubmitBidPlan;  
    #uses plan AnalyseBids;  
}
```

# Capability nesting



# Capability Concept Summary

## So, to reiterate, Capabilities:

- allow agent elements to be structured;
- provide encapsulation of functionality;
- declare interactions between encapsulated functionalities;
- facilitate re-use; and
- introduce sound software engineering practice to agent programming.

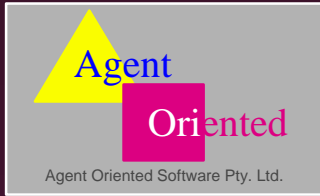
# Future work...

- Capabilities set the scene for more interesting work in the agent field. Especially interesting is the study of the relationship between capabilities and *roles*.
- An agent that is aware of its capabilities can bid for a role in a team.
- Work is currently in progress to formalise the Capability concept within the BDI framework (Lin Padgham@RMIT).
  - ❖ In addition, this work describes how “self-aware” agents can dynamically change their goals and intentions as their capabilities change.

# JACK Intelligent Agentsä

- A product by Agent Oriented Software
  - ❖ Current version (1.3) released May 1999. Includes the capability concept.
- Based on company's R&D
  - ❖ "third generation" agent system
- Component-based approach:
  - ❖ core architecture and capability for developing & running distributed software agents
  - ❖ allows for variety of types of software agent to be layered on top of base kernel, from simple agents to teams of intelligent agents





# JACK Intelligent Agents

- A significant advance in the maturity of agent oriented software engineering
- Brings agent technology further in line with mainstream software engineering techniques

<http://www.agent-software.com>