Agent-Based Computing
Faculty Research & Creative Activity Posters
Faculty Day 2004

Chang-Hyun Jo
Department of Computer Science
California State University Fullerton
jo@ecs.fullerton.edu
Recent Research

- BDI Agent Software Development Process
- Pure Instincts
- A Flexible Agent Coordination Language
- BDI Agent CASE Tool
- A Compiler Design for the Agent-based Programming Language
- A Modeling Language for the BDI Agent Software Development
- Collaborative CASE Tool for Agent Based Software Development Process
BDI Agent Software Development Process

- MS Student: Guobin Chen, Fall 2003
- Advisor: Chang-Hyun Jo
- Reviewer: James Choi
BDI Agent Software Development Process

- Intelligent agent has been regarded as a new concept to analyze and design complex software systems.

- This research proposes an agent-based software development process based on Belief-Desire-Intention (BDI) agent model as a new software development process.
  - Here we show a seamless software development modeling technique consistently based on the BDI model.
  - Each agent is made flesh by assigning its own belief, desire and intention.
BDI Agent Software Development Process

Applying External Use Case to extract goals from the requirement
Desires (Goals)

Applying Internal Use Case to capture the plans
Intensions (Plans)

Applying DFD to obtain the data moving in the environment
Beliefs

Requirement Analysis Design

Extracting Goals Capturing Plans Obtaining Beliefs

Enterprise Software Assessment Brief External Use Case
Detail External Use Case

Initial Problem Statement Goal Hierarchy Diagram

Internal Use Case Sequence Diagram Activity Diagram

Agent Belief List Agent Card Agent Class
BDI Agent Software Development Process

- To prove the usability our software development process, we also provide a case study to clarify the description of our BDI agent software development process.

- Case Study:
  - The California Super Lotto Finding System (CSLFS)
Pure Instincts

- MS Student: Kelvin Yong, Spring 2004
- Advisor: Chang-Hyun Jo
- Reviewer: Donna Kastner
Pure Instincts

- The problem of path finding occurs in many applications.
  - Robot movement, military applications, games

- The Pure Instincts algorithm provides an efficient method for the exploration and navigation of unknown environments, which can ultimately be used in time critical environments.
  - A combination of a modified Q-Learning algorithm, a modified A* Path Finding algorithm, and multiple agents
Pure Instincts

- The results and findings of the Pure Instincts program seek to bridge the gap between the sophisticated machine learning algorithms currently used in research projects and modern gaming environments, which currently employ very simple artificial intelligence algorithms.

- The Pure Instincts algorithm was designed to create a baseline for future work in the area of machine learning in gaming environments.
Pure Instincts

- The Pure Instincts Demo is intended to demonstrate the Pure Instincts algorithm.
  - Upon loading the program, the user will be presented with 4 agents inside a 3-D virtual environment.
A Flexible Agent Coordination Language

- MS Student: Andrew Hess, Spring 2004
- Advisor: Chang-Hyun Jo
- Reviewer: Donna Kastner
A Flexible Agent Coordination Language

- The purpose of this project is to present a new way for agents to coordinate their conversations.
- Many languages already offer this functionality, however, these solutions do not resolve all the known problems associated with agent coordination.
A Flexible Agent Coordination Language

- By introducing the practice of object-oriented design it is possible to solve the problems plaguing the coordination level of agent communication.
  - Inheritance can be used to allow conversations to reuse as much of the conversation as possible.
  - Polymorphism can be used to allow conversations to be processed in a generic way.
A Flexible Agent Coordination Language

- In order to test the strength of our new coordination language, a video game was created where characters in the game are agents and use the object-oriented coordination features described.

- Implementing the game showed true advantages in agent coordination including code reusability and extendibility.
BDI Agent CASE Tool

- MS Student: Willy Tanimihardja, Spring 2004
- Advisor: Chang-Hyun Jo
- Reviewer: Tae W. Ryu
BDI Agent CASE Tool

- The BDI Agent CASE Tool is one of the necessary instruments in supporting modeling and design process of the BDI Agent-based Software Development.

- This tool assists the agent modeling process by encompassing all of the Agent Modeling Technique (AMT) artifacts:
BDI Agent CASE Tool

- The application supports eleven different artifacts:
  - Initial Problem Statement
  - Enterprise Software Assessment
  - Brief External Uses Cases
  - Detail External Uses Cases
  - Goal Hierarchy
  - Internal Uses Cases
  - Sequence Diagram
  - Activity Diagram
  - Data Flow Diagram
  - Agent Cards
  - Agent Class Diagram
A Compiler Design for the Agent-based Programming Language

- MS Student: Dongshi Zhang, Spring 2004
- Advisor: Chang-Hyun Jo
- Reviewer: Xiong Wang
A Compiler Design for the Agent-based Programming Language

- Agent-based programming has been emerged as a new paradigm for programming autonomous systems.
- This project is to build a prototype compiler for the BDI Agent-based Programming Language (APL) [Jo and Arnold 2002].
  - This compiler translates the APL program into the Java bytecode which is enable to run on different hardware.
A Compiler Design for the Agent-based Programming Language

APL Program

APL Compiler

Java Bytecode

Java Virtual Machine

Multi-Platforms

agent CSLFS {
  belief <int> Hot_Numbers[] = new <int>[6];
  intention <int>[] RequestHotNumber()
    belief <int> i;
    belief <int> Numbers[];
    agent <HotNumberManager> c = new <HotNumberManager>();
    Numbers = c.RequestHotNumber();
    . . .
  return Hot_Numbers;
}

SOURCE

01 .source cslfs.apl
02 .class public synchronized cslfs
03 .super java/lang/Object
04 .field private Hot_Numbers [I
05 .method public <init>()V
06 .limit stack 2
07 .limit locals 1
08 aload_0
09 invokevirtual java/lang/Object/<init>()V
10 aload_0
11 bipush 6
12 newarray int
13 astore_3
14 aload_3
15 invokevirtual HotNumberManager/RequestHotNumber()[I
16 astore_2
17 aload_0
18 invokevirtual HotNumberManager/<init>()V
19 .end method
20 .method public RequestHotNumber()[I
21 .limit stack 4
22 .limit locals 4
23 new HotNumberManager
24 .limit stack 2
25 .limit locals 4
26 dup
27 invokevirtual HotNumberManager/<init>()V
28 astore_3
29 astore_2
30 aload_3
31 invokevirtual HotNumberManager/RequestHotNumber()[I
32 astore_2
33 . . .
34 astore_3
35 invokevirtual HotNumberManager/<init>()V
36 astore_2
37 astore_3
38 astore_2
39 aload_0
40 getfield cslfs/Hot_Numbers [I
41 areturn
42 .end method

Agent-Based Computing 2004 Chang-Hyun Jo
A Modeling Language for the BDI Agent Software Development

- MS Student: Tianzhi Zheng, Spring 2004
- Advisor: Chang-Hyun Jo
- Reviewer: James Choi
A Modeling Language for the BDI Agent Software Development

- The BDI Agent Software Development process was proposed as the next evolution in the software development process.

- This project is to describe precisely a modeling language for the BDI (Belief-Desire-Intention) Agent Software Development Process.

- This modeling language for the BDI Agent Software Development Process helps software designers to have a disciplined design approach in BDI agent software development methodology.
A Modeling Language for the BDI Agent Software Development

- There are twelve models in the BDI Agent-based Modeling Language.
- These twelve models cover the BDI Agent Software Development Process from gathering software project requirements to delivering design documents to the implementation team.
- Here we also provide a case study to introduce how each model works in our BDI Agent Software Development Process.
A Modeling Language for the BDI Agent Software Development

Case Study: Agent BDI Card for the Grading System

**Agent: Professor**

BDI List:

1) Desire: SubmitGrade
   Pre-condition: Professor has logged in the system, and the professor has added grades.
   Belief: NoticeDB, ConnectDB
   Post-condition: Grades are sent to ProfessorManager.
   Intentions:
   1) Professor asks ProfessorManager to submit students’ grades to the database.
   2) ProfessorManager calls DBAccessControl in order to gain access to the database.
   3) ProfessorManager submit students’ grades to the database.

**Agent: ProfessorManager**

BDI List:

1) Desire: RecordGrade
   Pre-condition: Grades are received from professor.
   Belief: Record
   Post-condition: Grades have been recorded to the database.
   Intentions:
   1) DBAccessControl connects the database.
Collaborative CASE Tool for Agent Based Software Development Process

- MS Student: Sujatha H. Thippeswamy
- Advisor: Chang-Hyun Jo
- Reviewer: Tae W. Ryu
Collaborative CASE Tool for Agent Based Software Development Process

- The Software engineering is inherently a team-based activity; most projects today are complex and typically require several iterations of design, development, and testing by numerous engineers.
  - Unfortunately, very few tools are available to support real-time, synchronous and collaborative software engineering.
  - There are many tools well suited to synchronous unstructured collaboration such as chat applications and shared digital whiteboard.
Collaborative CASE Tool for Agent Based Software Development Process

- This project is to build a collaborative CASE tool for Agent Based Software Development Process.

- This application supports collaboration within pairs of software engineers online.
  - The application supports sharing of SVG files (XML files) online by all the team members, making changes and saving them back to the server.
  - The modified files can be accessed by other team members.
Collaborative CASE Tool for Agent Based Software Development Process
Future Research Work

- On-Demand Streaming Service for both Multimedia Contents and Software Applications Using BDI Agents
- Security Issues in Mobile Agent Computing Environments
- Game Creation Techniques using Mobile Agents
- CMMI Using Agent-based Software Process
Previous Research

- Available at http://jo.ecs.fullerton.edu/research/
- The BDI Agent-Based Programming Language: BDI-APL (2002-2003)
- Development of a Web-based System for Interactive Browsing of Citation Flight Data (2001-2002)
- The Agent-Based Modeling Technique (2001 and 2003)
- Parallel-C++ for the Java Virtual Machine (2000)
- The Distributed Programming Environment (1999 and 2003)
- Audit Guidelines for Object-Oriented Software Development (1998)
- Design and Implementation of Parallel-C++ (1991)
- Abstraction and Specification of Local Area Networks (1988)