OPUS One
An Artificial Intelligence - Multi Agent based Intelligent Tutoring and Adaptive Learning Environment

September 2009

Dr. Attilio Pedrazzoli - Dr. Luisa dall’Acqua
Gsi Edu Research Group
Vision and Research finality – Global Overview of Concept Objectives

**Teaching**
- Pedagogical “freedom”
- Teaching habits
- ID Model transparency
- Collaborative functions
- Authoring tools
- Context based hint facility
- “One on one” facility
- etc.

**Tutoring**
- AI-, Profile-, Subject Matter based
- Human Tutor support
- Real time “AI-Behavior Recorder”
- Rule Engine, Rules based
- Subject Matter Knowledge base
- Allows Human Tutor interventions
- Selectable “Surveillance Modes”
- CTAT based Hint facility
- etc.

**Adaptation**
- Tightly coupled with Tutoring
- Human Tutor interface
- Own Presentation-, Adaptation Engine
- Rule Engine, Rules-, Profile based
- Dynamic Course-, Navigation structure modification
- Selectable “Adaptation Modes”
- etc.

**Student or Group (called “Learning Entity”) centered Focusing on Learning Success and Student Satisfaction**
Concept and Implementation Philosophy

Vision and Research finality - Architecture

- LMS Extension
  Primarily for easier access and personalized features

- Student Socialization Area
  Based on Apache Pluto

- Dedicated User Area
  for Socialization

- Cougaar
  Artificial Intelligence Framework

- OPUS One Extensions
  Integrated into the LMS System

- Generic LMS System

- AI – LMS Interface

- Artificial Intelligence Platform
  - Behaviour Recorder Engine
  - Knowledgebase Engine
  - Adaptation Engine
  - Presentation Engine
  - Rule Engine
  - Surveillance Engine

A. Pedrazzoli  September 2009  GSi Edu Research Group
Student Environment and Characteristics

- Profile driven with synchronized Course curriculum

- Mode selection
  . Tutoring Mode (if enabled in profile)
  . Surveillance Mode only
    e.g. Logging, Tracking and Behavior recording in silent mode
  . Walk trough Mode (standard mode if tutoring disabled in learner profile)
    Walk trough Mode can be considered as standard LMS (OLAT) exercise mode
Student Environment and Characteristics extended “native” LMS functionalities

- Personalization

. Personalized “Home Page” with the following features:
  - My Courses
  - My Groups
  - My Tutors per Course / I am Tutor in Courses
  - My Achievements, my Grade book
  - My e-Portfolio
  - My Tutoring Section (autonomous Window)
  - My News, Messages and Dashboard

. Personal Folder / Personal Resources / Free Open Courses Catalog by Faculty
OPUS One additional functions

- **Global Student Area**
  - Student Café / Global Forum / Global Wiki etc.
OPUS One additional specialized functions

- **Bidirectional asynchronous Wrapper**
  Tight integration of external systems for bidirectional Data- / Data structure transfer

- **Chat and Scribe collaborative Function**
  The Scribe Activity is used in 'dual-screen' roles, such as with a Chat activity it allows a group of Learners to collaborate and create a written collaborative document describing the outcomes of a chat session including features like attachment uploads, scheduling of subsequent activities etc.

- **Group based Video conferencing facility using DimDim**
  Tight integration of DimDim Videoconferencing facility as collaborative Group function as part of a Course.

- **Text analyzing Agent**
  Allows to analyze “any type of free form text” from collaborative activities based on predefined rules and associated parameters like:
  - presence and sequence of terms
  - amount of words, number of modifications, number of interventions
  - search activities (internal, external)
  etc.
**OPUS One future functions (alpha-/beta state)**

- **Graphical Course Editor**
  Development of a graphical interface for the OLAT LMS course editor allowing to build courses using “drag and drop”

- **Subject Matter (CTAT) Hints Authoring Tool**
  Development of an easy to use authoring tool to create “subject matter” specific tutoring- / adaptation rules

- **Subject Matter course simulator**
  Development of a script based course simulator, able to react on predefined “misconception” conditions to test the tutoring-/adaptation reaction of the AI system.
Concept and Implementation Philosophy

Vision and Research finality – How to make an Agent dedicated

**Cougaar plugins** are software components that provide a specific piece of application business logic to the agent.

The behavior of an agent depends primarily on its set of loaded plugins.

Agent profiles are parameter structures intended to characterize the agent in question.

**OPUS One - eTutor profiles** are developed and loaded as Cougaar plugins.
### Concept and Implementation Philosophy

#### Our “Learning Path Adaptation” Approach

| Why adapt | To ensure personal learning success  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Target driven, Student centered, considering actual knowledge, learning style</td>
</tr>
</tbody>
</table>
| When adapt | On recognized situations of “learning difficulties”  
|           | On recognized “under performance” compared to “subject matter” specific performance indicators defined in the profile.  
|           | On the learners personal request. |
| What adapt | The content presented according to a predefined profile, driven by “subject matter” adaptation rules  
|           | The actual “learning path” of the learner according to his profile and “subject matter” adaptation rules  
|           | Additional “subject matter” content upon the learners personal request |
| How to adapt | Based on profiles and rules by the AI – adaptation agent community  
|           | On “Human Tutor” intervention request via AI - API  
|           | On the learners personal request defining his personal learning approach via AI - API |

A. Pedrazzoli  
September 2009  

GSi Edu Research Group  

10
To be “Tutoring and Adaptation” enabled Learner (LE) must be:
- Personal profile
- Subscribed for a “tutored” Course
- Have a “curriculum” (e-Portfolio)

LDAP – Authentication profile:
- Learner tutoring parameter extension

Tutoring and Adaptation setup actions:
- Learner Agent will be selected and activated
- Learner profile associated to the LE Agent

LMS action:
- Learner personal (personalized) Homepage presented
- If Tutor enabled additional tutoring functions enabled in the Learners Homepage
A Tutor enabled Course consists of:
- Standard LMS Course
- AI Course Wrapper
- Main “Subject Matter” Profile and associated Sub profiles (per Section)
- Linked “Decision rules”
- Linked “Path adaptation rules”
- Misconception Surveillance
Concept and Implementation Philosophy

OLAT - LMS Course Environment

- START
- Syllabus
- Section 1
  - Subsection 1A
  - Subsection N
- Section N
- End Course

OPUS One Tutoring and Adaptation Environment

- Rule Engine
  - Generate SM Agent Community
  - Activate SM Expert Engine
  - Activate Misconception Analyzer
- Knbase DB
- AD Rules DB
- Execute Adaptation via Agents
- Adaptation Engine
- Activate Behavior Recorder
- Associate SM - Agent

A. Pedrazzoli   September 2009

GSi Edu Research Group
“personalizing / adapting” Concept

Tutor enabled, adaptive “Subject Matter”

Course section example

- Section 08
  - Subsection 08.1
    - Go to HTML page
    - Read (page)
    - Download PDF doc.
    - Discuss File

“Tutor enabled, adaptive” content:

- Profile controlled
- DB based Logging and Tracking
- Behavior Recorder enabled
- Behavior-, Knowledge driven “Learning Path” adaptation
- Section- / Subsection- and Element based deficiency management
  (Misconception detection and associated actions)
- Event based Rule Authoring and Execution

Subject Matter Profile

Behavior Recorder

Logger / Tracker

Misconception Management

Knowledgebase

DB, Rules, Procedures etc.

Rule Engine
Cognitive Tutor Concept

**LMS (OLAT)**

Selected e-Course

- Syllabus
  - Section
    - Subsection
      - Content
      - Collab. Tools
      - Assessments
      - etc.

**AI – LoTr Controller**

**AI – TuBe Controller**

**AI – Feedback Controller**

**Rule Engine**

**LTB DB**

**3HTutor**

**AI - Society**

**Behavior recorder**

**AI – Society including Misconception Management**

Rule Execution (Tutoring or Surveillance Mode)

1) – Logging/Tracking Agent pass relevant L/T parameters to the 3HTutor Evaluation Agent

2) – Behavior Recorder will pass “Subject Matter” defined parameter block to be elaborated by the Rule Engine using a sequence of algorithms defined as RuleSeqID parameter

LoTr Controller will pass
- CourseID, SecID
- SecTimeID
- Par1 to Par5 (optional)
To 3HTutor Eval. Agent

Behavior Recorder will pass
- CourseID, SecID
- RuleSeqID, ReturnPar
- ParX1 to ParX(n) to evaluate the action to 3HTutor Evaluation Agent
The control structure for the “learning path adaptation” is based upon the OLAT course structure and its access and visibility mechanism. We use here the “Score” approach. 3HTutor and its OLAT interface Agents are able to dynamically adapt / modify this parameters based on defined rules, driven by “Student Course activities”, transformed in parameters by the “Behavior Recorder” and “Misconception Manager”.

**Example**

<table>
<thead>
<tr>
<th>Section 08</th>
<th>Content Page</th>
<th>Test S08</th>
<th>Subsection 08.1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Min. Entry Score</strong></td>
<td><strong>Finish Score</strong></td>
<td><strong>Add. Score</strong></td>
<td></td>
</tr>
<tr>
<td>800</td>
<td>N.A</td>
<td>N.A</td>
<td></td>
</tr>
<tr>
<td>800</td>
<td>10</td>
<td>N.A</td>
<td></td>
</tr>
<tr>
<td>810</td>
<td>N.A</td>
<td>Test dependent</td>
<td></td>
</tr>
<tr>
<td>850</td>
<td>N.A</td>
<td>N.A</td>
<td></td>
</tr>
</tbody>
</table>

The Student will be allowed to follow the original “Learning Path” if he achieve the minimum scoring predefined per Section, Subsections and Course Elements, the score of the single course elements will be considered for possible adaptive actions.
"Adapting Learning Path” Concept (2)

Actual Student course curriculum-score is - 820

- Section 08
  - Subsection 08.1
    . Go to HTML page
    . Read (page)
    . Download PDF doc.
    . Discuss File
    . as collab. Assignment
    . Test
  ** end Subsection

Requested Entry score set to 810

Content Element "Read Page" score credit set to 5
- Control param are :
  - Entry in element
  - min read time 120 sec.

Content Element "Download elem." score credit set to 5
- Control param are :
  - Entry in element
  - Download successful.

Content Element "Discuss File" score credit set to Var.
- Control param are :
  - Entry in element
  - Text evaluation (AI–text agent)
  - Human Tutor final evaluation.

Content Element "Test" score credit parameters set to
  - Min. score to pass / Max. score (example: score 50).
- Control parameters are :
  - Test attempts
  - Test responses / sequence
  - Time for completion
  - Test score.

Min. score to pass Subsection : 60
Min. score for next Section: 870
What happens if:
- Student globally fails minimum element score requirement

Example – Read page time less then specified

- Section 08
  - Subsection 08.1
    . Go to HTML page
    . Read (page)
    . Download PDF doc.
    . Discuss File
    . Test
  ** end Subsection

Base parameter structure

Content Element “Read Page” score credit set to 5
- Control param are:
  - Entry in element
  - min read time 120 sec.

Actual Tracking parameters:
- Entry_element = true
- Total_time_element = 32

Rule Engine

Sec08_readp_01 rule:
IF Entry_el = true and
Tot_time >= 120
then ret_cre = 5;
else ret_cre = 0

AI Subject Matter Agent
- Update curriculum Score
- if return Code false then
  activate tutoring activity according to
  “Subject Matter” profile

Do scheduled Tutoring action according to Subject Matter profile linked exception rules
Cougaar Agent Reference Model

- **Agents**
  - Local
  - Behavior (plugin)
  - State (BB)
  - Pub/Sub (BB) API
  - Black Board

- **Agent/Env. API**
  - sensor
  - effectors'
  - coordinator
  - active API

- **Environment**
  - Distributed
  - Services
  - Components
  - Imported libraries
  - Service oriented API

---

A. Pedrazzoli  September 2009

BBN Copyright 2007
Separation of Application from Environment

Agents handle **Application Behavior**
Environment handles **Systemic Adaptation**

Agents and Environment can be independently developed, tested, and configured, but **run together**
Integration with Legacy Systems

Main Cougaar architectural feature: imported libraries and component wrappers
Application

1. Functional modules (oval shaped)
2. Underlying distributed environment
3. Sensor to control loop coordination
4. Evolving degree of human involvement

Cougaar

1. Agent societies
2. Cougaar environment
3. Agent coordinations
4. Transitioning of control loops human to automation
Agent Community Adaptation

- **Adaptation** picks the best implementation which meets the application QoS requirements within the resource constraints.
- To make this **tradeoff**: adaptive systems must have:
  - Multiple implementations
  - Characterization of each implementation based requirement and constraint conditions
  - A mechanism for **detecting** the system’s conditions
  - A policy for **choosing** which implementation given the conditions.
  - A mechanism for **enabling** the implementation
Support for Coordination Artifacts

Method for the Separation of Functions: Coordination logic vs. Domain logic

- Coordination Artifacts: CAs
  - Are first-class entities in MAS. They are designed to separate communication mechanisms from the domain-specific work of any given Agent
  - Define explicit roles for role-players
  - Offer shared state between the role-player & the CA
  - Coordinate behavior among role-players
  - Have distributed implementation
QoS-Adaptive Translation
Changes the Translation Mechanics to Match the Situation

QoS-Adaptive Translation
Deltas

Change Detection

Transfer Constraints

Change Reconstruction

• Translation should take into account
  – Structure of starting and ending data structures
  – Probability and frequency that structures will change
  – The constraints of the transfer path

Latency = Load / Capacity
Frame Set Knowledge Representation

- Java Objects are code generated
  - Frames and relationships defined using XML
  - Support multiple Java interfaces
    - Cougaar Blackboard,
    - JESS Shadow Facts,
    - Java Beans
    - Web Server
- Slot inference (Real-time)
  - Type (is-a)
  - Containment (has-a)
  - Visitor Pattern (composed-of)
  - Aggregation (summary-of)
- Relationships are also Frames
  - Benefits from Frame inheritance
- Meta-data tags
  - Defined at compile-time
    - Slots, frames, framesets
  - Example Slot meta-data
    - Type, default-value, units, path, doc, member, warn, immutable, notify-blackboard, notify-listeners, transient
Multiple Knowledge Processing Frameworks

Agent Domain Processing

Rule code

<table>
<thead>
<tr>
<th>LHS</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patterns</td>
<td>Trigger Assert Retract</td>
</tr>
</tbody>
</table>

Procedural code

Domain Processing

- Domain Routines
- Domain Objects
- Code Libraries

Agents Concentrate on domain processing

Facts from multiple Partitions

Partitioned Blackboard

Coordination with External Systems

Coordination with Physical Environment

Coordination with Peer Agents

Real-Time Knowledge Feeds
Conclusion - Where do we stand today

Preproduction Test System
Demo System scheduled availability end October 2009

Common Shared LDAP Authentication

1) Tight Integration OLAT Cougaar AI - MAS via dedicated Agent - Portlets
Conclusion - Where do we stand today

eTutor enabled “Course page” OLI Layout on OPUS One / OLAT 6.1.x

**eTutor controlled Section Subsection Status.**
- **RED** - ▲ Not executed or not passed
- **Yellow** - ▶ Work in progress
- **Green** - ◼ Section passed

-Clicking on the symbol will show the detailed STATUS
Conclusion - Where do we stand today

**3HTutor Status**
- Enabled / Disabled

**eTutor Function**
Available eTutor functions are User profile depended.
Using the eTutor facility, a "enabled" User is able to switch between 3 eTutor – Modes:

1) eTutor active – request to adapt “tutoring” level (if allowed in the profile).
2) Request eTutor help – (Tutoring on demand function)
3) Switch to “Walk trough” mode – this means “Exercising mode”, an optimized “Exams” preparation mode.
Conclusion - Where do we stand today

- eTutor – Request “Tutoring on Demand” Example – Active AI Agent (3HTutor)
- Hint- / tutoring on demand is performed using the CTAT approach proposed by a dedicated presentation engine.

**eTutor request will open Exercise according to “Context specific” rules.**

The exercise has rule controlled, responses. The “Behavior Recorder” will log and supervise all actions done by the Student.

**Context sensitive AI Hint Agent, will display a cascading series of problem context sensitive Hints based on subject matter specific rules.**
Conclusion – Future developments

- Research on standardization of the “Adaptation engine approach”
- Course construction using “dynamic reusable learning objects” loaded on demand from the learning object repository (DSpace).
- Learning path creation using “Student personalized” schema approach
- Integration of GLO Maker interactive learning resources
- Make the 3HTutor environment LMS independent
- Generic independent bidirectional asynchronous Wrapper based on Cougaar AI-MAS Agent Community
- Integration interface for LAMS, Sakai 3 and Moodle
THANK YOU FOR YOUR ATTENTION

Attilio Pedrazzoli
avp@gsiserver.com