A Generative Decision Support Architecture (GDSA)

Doug Lange
Mike Cowen
Mark St. John

15 January 2002
C⁴ISR Model

- **Data Collection (Sensors)**
- **Processing** (Correlation, Association, Fusion)
- **Situation Map (Perceived Truth)**
- **Intelligence Preparation of the Battle Space (User Input)**
- **Collection Plan (User Input)**
- **Database of Forces, Assets, etc. (Ground Truth)**
- **Decision (Command and Control)**
- **Action (Movement, Combat)**

Lt Col Greg McIntyre, C2 Modeling in JWARS
Decision Support Model

- **Data Collection (Sensors)**
  - Processing (Correlation, Association, Fusion)
  - Information Collection and Analysis (Agent)
  - Collection Plan (User Input)
  - Commands and Monitoring Plan
  - Situation Map (Perceived Truth)
  - Decision (Command and Control)
  - Action (Movement, Combat)

Communication flows between the components.
Agents

Russell and Norvig, Artificial Intelligence: A Modern A

Objective

• Develop an agent generation architecture for decision support applications.
  – Improve the tie between cognitive task analysis and software development.
  – Lessen the time necessary for developing decision support software.
  – Improve the quality of decision support software
  – Provide the flexibility necessary to support NCW

• Provide a method to evaluate an agents contribution to decision support.
Decision support requirements change rapidly in the operational war-fighting environment.

Our current process for developing decision support software cannot meet the needs of the move towards NCW. Even current demands are stressing our capabilities.
Technical Approach

• Identification of cognitive task domain.
• Evaluate models of cognitive decision-making.
• Define a cognitive model that describes the environment.
• Translate the cognitive task model into a formal software model within a generative software architecture.
• Create a domain specific language (DSL).
• Domain design.
• Domain implementation.
Template Based Techniques

Quava

Repository Adapters
Model Editor
Schema Server

Metadata

Templates
Generated Source Files
Remote Files

Client
Business Application Server
Developer’s Source Files

Information Repositories
Oracle, Sybase, etc

LEGEND
Quava Component
COTS Component = Hand-Coded Component

Generative Software Development

Domain Engineering

- Domain Knowledge
- Domain Model
- System Family Architecture
- New Requirements

- Domain Analysis
- Domain Design
- Domain Implementation

Application Engineering

- Requirements Analysis
- Custom Design
- Integration and Test

- Product Configuration
- Custom Development

- • Domain-specific languages
  • Components
  • Generators

SEI, Model-Based Software Engineering
Partial Evaluation

\[ [p] \{ \text{in1, in2} \} = [p_{\text{in1}}] \text{ in2} \]

Jones, “An Introduction to Partial Evaluation”
GOMS Models

Method for goal: edit the document
Step 1. Get next unit task information from marked-up manuscript.
Step 2. Decide: If no more unit tasks, then return with goal accomplished.
Step 3. Accomplish goal: move to the unit task location.
Step 4. Accomplish goal: perform the unit task.
Step 5. Goto 1.

Selection rule set for goal: perform the unit task
If the task is moving text, then
accomplish goal: move text.
If the task is deletion, then
accomplish goal: delete text.
If the task is copying, then
accomplish goal: copy text.
... etc. ...
Return with goal accomplished.

Method for goal: move to the unit task location
Step 1. Get location of unit task from manuscript.
Step 2. Decide: If unit task location on screen, return with goal accomplished.
Step 3. Use scroll bar to advance text.

Method for goal: move text
Step 1. Cut text
Step 2. Paste text
Step 3. Verify correct text moved.
Step 4. Return with goal accomplished.

Min, et. al., “Distributed GOMS: An extension of GOMS to Group Task”
A Colored Petri Net Model of Tactical Decision Making

Lu
Petri Nets for Code Generation

F. Kordon, I. Mounier, E. Paviot-Adet, D. Regep, “Formal verification of embedded distributed systems in a prototyping approach”
Domain Specific Language

• Based on Deterministic Timed Hierarchical Colored Petri Nets. Adds semantic content to the places, transitions, and edges relative to decision support agents.
  - Interactions with infrastructure
  - Use of sensors and actuators
  - Information item and list management strategies
  - Analysis steps
  - Result display

• Initial level is based on the level of reusable sensor and actuator modules and the level of abstraction of associated information objects.
Research Areas

- Cognitive model adaptation for decision strategy description for agent use and generation
  - Critical Information Requirements List
  - CIRL management criteria
  - Information analysis method
  - Result display
- Domain specific language for decision strategies
- Agent generation engine
- Sensor and Actuator reuse bases and semantic descriptions for selection
- User interface language for decision strategies
- Case-based reasoning support for decision strategy selection