agentToolIII (Static)

MSE Project Presentation 1

Presented By:
Deepti Gupta

Agenda

- Project Overview
- Project Requirements
- Project Plan
- Cost Estimation
- Architecture Elaboration Plan
- SQA Plan
- Demo
Project Overview

agentTool

AgentTool is a Java-based graphical development environment to help users analyze, design, and implement multiagent systems. It is designed to support the Multiagent Systems Engineering (MaSE) methodology. The system designer defines high-level system behavior graphically using the Multiagent Systems Engineering methodology. The system design defines the types of agents in the system as well as the possible communications that may take place between agents and the environment.

agentToolIII (Static)

My project called agentToolIII (Static) will help the user to draw static agent diagrams like Goal model, Organizational model, Role model and Agent model. The user will be able to perform static structural modeling for efficient design of multiagent systems. The tool provides the user with the capability of saving the diagrams.

Project Overview Diagram

Figure 1. Project Overview
Use Case 1 – View System

This use case describes the capability of the user to view the entire system and browse through each system diagram.

- The user will be able to click on any diagram in the hierarchy to view or edit it.
- The user can delete diagrams
- The user can create new diagrams as part of the loaded system
Use Case 2 – Draw Static Diagrams

- Static Diagrams
  - Draw Goal Model
  - Draw Role Model
  - Organizational Model
  - Agent Model

Critical Requirements
- Each model entity will have attributes name and description which the user can edit.
- Each diagram will provide a drawing capability in which the user can drag/drop the required icons onto the drawing space.
- The user will also be provided with the facility to remove/delete model entities, move entities around in the drawing space and modify their properties.
- Corresponding to each user action for every diagram, the system object model will be maintained by the tool in the background.

Goal Model

- User can define goals and sub-goals
Role Model

- Each organization will have a role model, which defines the various roles that must be played within the organization. These roles may provide services and require capabilities.

Organizational Model

- The organization model would be used at the top level to describe how the top-level organization fits into its environment.
Agent Model

Critical Requirement
- The agent model describes various roles that an agent can play and what capabilities it possesses.

Use Case - Change View
- Some of the boxes in the role, agent and organizational model diagrams can be hidden and annotated within other boxes. Instead of showing a <<possesses>> relationship between agents and capabilities, the capability name could be embedded within the agent box. This forms an alternate view of the diagram. The relationships that can be annotated within other boxes are: <<plays>>, <<possesses>>, <<provides>>, <<requires>>

Eg:
Use Case 3 – Import XML

Critical Requirement
- This use case describes loading XML representations of the environment model into the tool so as to render the diagrams.

Use Case 4 – Export Model

Critical Requirement
- This use case describes how a user can save his diagrams by saving XML model files.
- The user can select a filename for model saving via a provided dialog interface.
- The system model will be transformed using the XML schema. The model will be saved as an XML file.
Use Case 5 – Printing Diagrams

- The user should be able to print out a hard copy of any diagram he wants.

- Select the printer
  The user should be able to select the type of the printer he wants to use for printing. A dialog box will appear for this purpose.

- Scale the diagrams to fit the paper
  The diagrams will be made to fit on the paper. The user will have the option to select this in the dialog box.
**Cost Estimation**

- **COCOMO Organic Model**

  \[
  \text{Effort} = 3.2 \times \text{EAF} \times (\text{Size})^{1.05} \\
  \text{Time (in months)} = 2.5 \times (\text{Effort})^{0.38}
  \]

  where
  
  \(\text{EAF} = \text{Effort Adjustment Factor}\)
  
  \(\text{EAF evaluated to 0.94 based on 15 adjustment factors}\)

**Calculations**

- **Size Estimation**: The previous agentTool has a hand-generated codebase of 3KLOC. Since the new tool has additional functionality for modeling capabilities, I estimate the tool size to be around 3.5KLOC.

  Hence, the effort calculates to:

  \[
  \text{Effort} = 3.2 \times 0.94 \times 3.5^{1.05} = 11.2 \text{ staff months} \\
  \text{Time} = 2.5 \times 11.2^{0.38} = 6.2 \text{ months}
  \]
Architecture Elaboration Plan

- **Revision of Vision Document**
  The Vision Document will be revised to provide a complete representation of all requirements.

- **Revision of Project Plan**
  Revisions will be made to the project plan to update any schedule changes for the project.

- **Architecture Design**
  Complete project design will be documented with the help of UML diagrams. Each component in the architecture will be documented at the interface level. The underlying data model will be refined, updated and documented. Class diagrams, sequence diagrams and state chart diagrams will be used as needed.

- **Development of Prototype**
  The second prototype should be able to demonstrate more functionality and product features. Critical requirements as identified in the vision document will be demonstrated.

- **Test Plan**
  A plan will be developed for the project to address the required tests to show that the product satisfies the requirements as mentioned in the vision document. The plan will include evaluation criteria for all critical use cases and a set of test data deemed adequate for acceptance testing.

- **Formal Technical Inspections**
  The architecture design will be inspected by Binti Sepaha and Dominic Gelinas. A formal checklist to be used by the inspectors will be prepared.

- **Formal Requirements Specification**
  It is proposed that a part of the object model will be formalized.

SQA Plan

- **SQA Plan**
  SQA Plan was developed during this phase intending to outline a plan describing what measures will be taken throughout the project lifecycle to achieve an acceptable level of quality in the final product.

- **Based on**
  IEEE Guide for Software Quality Assurance Planning

  IEEE Standard for Software Quality Assurance Planning
SQA Plan (contd)

- **Documentation**
  The software portfolio for the developer will be found at [http://www.cis.ksu.edu/~deepti/MSEProject/](http://www.cis.ksu.edu/~deepti/MSEProject/)

- **Standards, practices conventions and metrics**
  **Documentation standards**: Wherever applicable IEEE standards will be used as a guideline to prepare all documentation. Object oriented analysis and design methods will be used.
  **Coding standards**: Java naming conventions will be used throughout the project. Javadocs will be used to generate API documentation.
  **Metrics**: COCOMO will be used for effort estimation

SQA Plan (contd)

- **Reviews and audits**
- **Test and problem reporting**
- **Tools, techniques and methodologies**
  - Eclipse IDE – for development
  - SWT (Standard Widget Toolkit) – for implementation
  - USE – for formal specification
  - MS Word – for documentation
  - Microsoft Project – for project schedule
  - Eclipse Jigloo Plugin – for GUI building
- **Media Control**
Deliverables

- **Phase I:**
  - Vision Document
  - Project Plan
  - Demonstration (Initial Prototype)
  - Software Quality Assurance Plan

- **Phase II:**
  - Vision Document
  - Project Plan
  - Formal Requirements Specification
  - Architecture Design
  - Test Plan
  - Technical Inspection Checklist
  - Executable Architecture Prototype

- **Phase III:**
  - User Manual
  - Source Code
  - Project Evaluation
  - Formal Technical Inspection Letters
  - Executable project

Initial Prototype Demonstration