Designing Autonomous Agents for Process Control.

Virginio Chiodini
vlc@gensym.com

Anshu Mehra
am@gensym.com

Gensym Corporation
125 CambridgePark Drive
Cambridge MA, 02140.

During the past two years we have developed the Agent Development Environment (ADE) and have used ADE in multiple planning and control applications. In this paper we provide a brief overview of ADE and present two Adaptive Control applications developed with ADE.

The Agent Development Environment is an integrated platform for building distributed multi-agent applications. ADE enables a graphical definition of Agents and Agents' Activities. ADE provides a simulation environment in which Agents can be tested before being deployed. ADE is built on G2, an object-oriented graphical environment that offers a robust platform for the development of real-time systems.

Distinctive components of ADE are:

- A predefined class hierarchy of agents and agent components.
- An agent communications "middleware".
- A graphical programming language to design and develop Agents' behavior based on the Grafcet standard.
- A complete debugging environment.
- A distributed simulation environment to test Multi-Agent applications built with ADE.
- A deployment center to deploy Agents as G2 objects or JavaBeans in a Java virtual machine.
- A Learning Context to enable Agents to develop their behavior through a learning process.

Multi-Agent applications built with ADE may run on a single machine or on a distributed network. Agents communicate through ***Messages***. ADE provides a basic direct addressing message service, with some optional functionality, for example, guaranteed delivery and subject-based addressing. ADE uses a delegation based event mechanism similar to JDK 1.1 model. Agents use messages to generate and listen for events. Each Agent has a network-wide unique ***Name*** and can be endowed with specific ***Properties***. Using an SQL language, Agents can query the Multi-Agent environment in search of Agents with specific properties. ADE also enables Agents to send messages to other Agents qualified by their properties. An Agent can concurrently perform multiple ***Activities*** while multiple Agents can perform a specific Activity. Agent Activities can be ***Transient*** or ***Permanent.*** Transient Activities are created and started by messages sent by Agents and are deleted when the tasks assigned to the activity have been completed. Permanent activities are created and started at Agent initialization and are deleted at Agent termination. Permanent activities concurrently perform tasks requested by multiple messages.

Agent Activities are defined as ***Grafcet,*** using the ***ADE Grafcet Development Environment.*** The Grafcet Development Environment facilitates the design and development of single- and multi-thread Agent Activities as standard IEC-848 Function Charts with a sophisticated graphical environment. Agent activities defined as Grafcet Charts can be executed in interpreted or compiled mode. In Interpreted Mode, Agents activities can be monitored and suspended with a variety of graphical tools and can be modified in real time by modifying the corresponding Grafcet Chart.

Applications.

ADE is currently being used in two prototype control applications:

- A Model Predictive Control application of an Induction Heating Process.
- A Model Free Control application of an Electrolytic Metal Plating Process.

Closed-Loop Control of an Induction Heating Process.

Induction Heating involves placing an electrical conducting work piece in a varying magnetic field. The magnetic field induces eddy currents at a power level and frequency in the work piece. The work piece heats up because it has resistivity to the eddy currents induced by the magnetic field. The goal of induction heating for forging is to uniformly heat a work piece to the proper forging temperature. Work pieces are moved through several induction coils whose voltage can be controlled. Multiple work pieces may be concurrently processed within one induction coil. The goal of the closed-loop control system is to adjust the voltage of the induction coils to stabilize the temperature of the outgoing work pieces, managing variations in temperature of incoming pieces and transitions from "run" to "hold" and back to "run" of the work pieces along the heating line.

An adequate mathematical model is available for the induction heating process. Using the mathematical model, we developed and validated an Agent Based process simulator. The simulator is then used to enable Agents to learn an appropriate control policy.

Closed-Loop Control of a Plating Process.

In a Plating Process work pieces are moved through a set of baths that deposit various metals via electrolysis. At the end of the process a control device checks the thickness of each metallic layer and evaluates discrepancies between set points and actual values. The process has two control variables: the rectifier current of each electrolytic bath and the line speed. Increasing the speed of the line increases the production throughput, but demands higher current in the electrolytic baths. High current levels may cause instabilities in the plating process. When the thickness of a specific metal is insufficient, plates must be discarded, causing a loss in productivity. On the other hand, raising thickness set points would cause a waste of precious metals, increasing production costs. Other state variables like metal concentration, temperature, and PH affect the plating process. The goal of the closed-loop control system is to maintain the best possible equilibrium among these competing requirements and to manage unpredictable changes in the conditions of the environment that affect the plating process. The functional relationship between current, process time and metal thickness can be determined only via complex mathematical processes. Lengthy and costly system identification procedures must be performed for each different type and shape of plate to derive transfer functions linking each control variable to the thickness of metallic layers. No accurate model is available to evaluate the interaction among the different control variables.

Agent-Based Design.

In both applications process entities are represented by Agents. We distinguish two types of Agents:

- ***Static Agents.***

The behavior of Static Agents is defined when Agents are created through Grafcets or standard methods. Work pieces heated by an Induction Heating Process are examples of Static Agents. The main task of a "Work Piece Agent" is to evaluate its final temperature profile, given the values of the control variables (for example, voltage and frequency) utilizing the mathematical model of induction heating.

- ***Dynamic Agents.***

The behavior (policy) of Dynamic Agents is developed and refined during the learning phase. All the Agents that operate control variables are modeled as Dynamic Agents. Dynamic Agents adapt their behavior by interacting among them and with the environment through the Learning Context. The Learning Context is a special activity of Dynamic Agents that continuously modifies their control by optimizing environmental feedback, through a mapping between perceptions and actions. Learning Contexts operate until stable and consistent sequences of good control decisions are made. The Learning Context is activated whenever the current control policies fail to produce good results in unexplored regions of the state space. Several learning contexts with different learning strategies have been developed. All the Dynamic Agents of an application must use the same Learning Context.

When a model of the process is not available (for example in the Plating Process) two options are available:

1. Learn an approximate model using Supervised Learning techniques like Feed-forward Neural Networks. In this case the learning process can take place in a simulated environment. Work-pieces can be represented as Static Agents whose behavior is defined by a Neural Network.

2. Learn control policies without learning a model of the environment. In this case a simulation environment is not available. The Learning phase must be performed in real-time concurrently with the execution of the actual production process. Some techniques developed in Reinforcement Learning enable Agents to learn directly state-action value. A model-free approach is feasible only if the speed of convergence of the Learning Process is fast.

Both approaches are currently being evaluated.