A New Programming Paradigm

Agent-Based Programming Technology

Echo Messaging Systems, Inc.

Agent-based programming technology created by Echo Messaging Systems is at the heart of our ADIN AI framework. ADIN, which means Anomaly Detection and Intelligent Notification, is a set of software technologies that enable rapid development of highly complex and extremely stable software systems.

Programming paradigms such as structural, procedural, object-oriented, and functional paradigms describe ways to organize specific programs solving problems in human readable formats that, through compilation, transform into computer executable formats. Agent-based programming is a new paradigm where agents are highly structured and configurable, are combined to run independently of each other, as parallel and/or autonomous processes to solve larger complex problems. As the problems get bigger and more complex, agent-based programming does not equally become more complex and unwieldy. Agents easily scale in number, in configurable options and adaptability without becoming more difficult to manage.

Why Agent-based programming technology?

ADIN Agents run in containerized environments, which means the hardware requirements can be tailored and managed via container orchestration systems, such as Docker and Kubernetes. ADIN has a UI browser where agents can be monitored at high levels or drilled down to view all the specific actions taken.

ADIN Agents take on different types of tasks such as application, health, integrity, adaptive and more. A complex software system we created called Safe Zones is built upon the ADIN AI framework. All system requirements that can be performed autonomously are done via ADIN Agents, such as synchronizing data from CMS's, keeping Safe Zone contact status up-to-date based on a complex set of expiring test results, and notifying via email, text and phone calls as needed.

ADIN UI Interface – Summary of ADIN Cells

The structure of an agent is two basic components: the triggering criteria and the action response. The agent-based programming paradigm is to address complex requirements in terms of these two components. Additional agents can interact and update agents and are known as adaptive agents. For example, an adaptive agent that monitors the how many records are processed by an application agent, can have an action response to clone the application agent, and update parameters into the original and cloned agents to divide up the workload. Health agents monitor agents to make sure they are operating within statistical norms that are tracked, such as how often they fire and how much data

they process. These qualities define norms and the health agents triggering criteria fires when quantities go outside the norms.

Triggering criteria is based on any set of data, time ranges, and/or GPS/location-based ranges. Data triggers can be when something new, or changed is detected, which can come from any digital data source, such as a database, file system files, and/or API's. API's can be connected to software systems or IoT systems.

Action responses are all the actions that happen in response to something - a new status will result in a new record, a change in field value will cause a notification to be issued, or a device's GPS location crosses into a geo-fenced region will result in a status device being turned off.

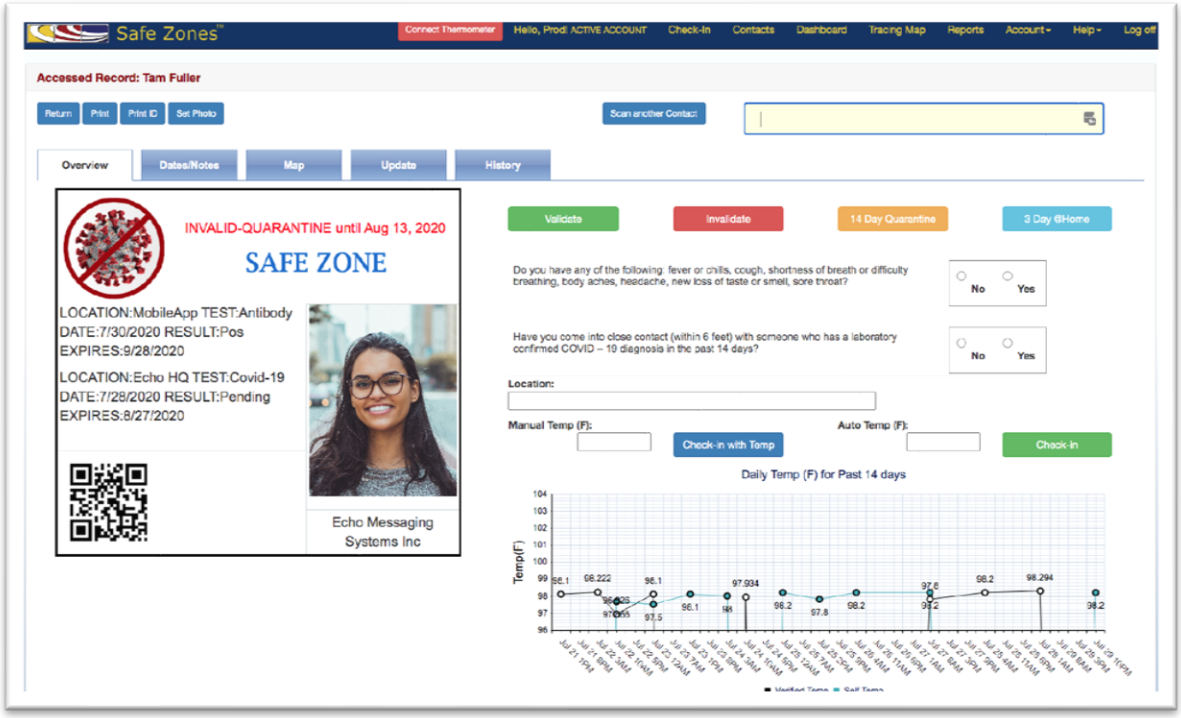
Agent-based programming means the complex computation of 'what has happened, when and where', along with requirements of 'what shall we do in response' has been distilled down to triggering criteria and action responses. Agents become building blocks that run independently and autonomously from each other.

Proven Approach

Time is about 1/6th the time spent developing software using Agent-based programming. This is because coding time centers on triggering criteria and action responses without the overhead of issues that make software more complicated and slower to developing when using object-oriented and procedural approaches. Constraints such as 'one time only', 'one per update', 'daily', etc. are configuration settings. Agents are parameterized to provide access to Read-only or Read-write parameter settings during configuration. Read-write parameters become available to other agents that provide adaptation as their action.

Faster development time and better use of programming human resources result in lower costs, faster development times and the ability to create more complex and adaptive software. This in turn means that costs are lower for clients and projects delivered on faster time frames. In our experience with agent-based programming, we were able to develop agent-based processes in one week that using traditional object-oriented programming paradigms would have taken up to 6 weeks.

Safe Zones is an example, where ADIN agents perform all business logic tasks, such as data syncing CMS contacts, test results and contact trace alerts. Safe Zone contact status changes are updated as new tests are added, and as time



Safe Zones Interface – Contact Page

passes to check for expiration dates. Also fever warning, O2 saturation warning, contact importing overages, GPS-based contact tracing, etc. are also processed via ADIN agent-based programming. As more contacts and accounts increase, complexity is handled by adding more agents that target specific accounts.

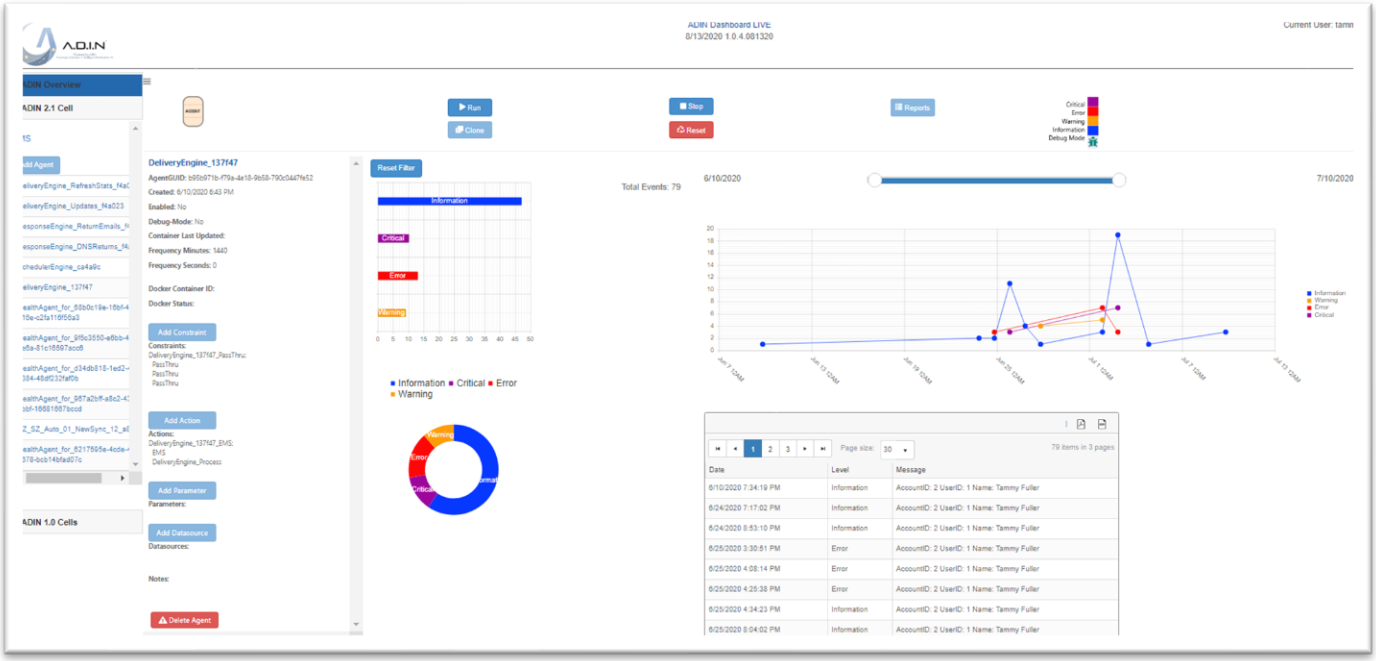
Tests and Vaccines can be added at any time, and expiration dates updated at any time. This continues to be a fluid area, and agents monitor all contacts and test/vaccine expiration dates, and any that expire and cause the contact status to change, such as 'Expired' or 'Quarantine' are notified by email, text and phone call. Additional agents can be quickly added as new triggering criteria and action responses are identified.

Echo Messaging has 15 years of seeing agent-based programming work because all application programming systems developed were based on agent-based processing and led to the creation of ADIN.

Beyond a Microservices Architecture

Microservices architectures have been increasing in number and dominance as an approach to creating complex software. Agent-based programming is 15 years beyond microservices architecture. Echo skipped over a microservices architecture and developed a full agent-based framework that provides structure and usability. Where microservices wrap usable applications and small programs into separate units, ADIN's agent-based programming has structure and usability to combine, build, recombine at a much larger scale. ADIN can spin up very large numbers of agents and is flexible in how agent technology is utilized to solve problems.

Agents can be independent and autonomous, where each agent is configured to solve their own piece of a large problems through a divide-and-conquer approach. Or agents can be mass configured to attack problems where hundreds, thousands, or more agents can be configured similarly with slight variations. Agents that fire more often can be adapted to have more processing time and those not processing can be deactivated. This allows for processing to adapt to the environment to make best use of processing resources. These are just two examples of how agent-based programming go beyond current microservice architectures to bring about a shift in processing paradigms.



ADIN UI – Agent Details