Accepted to 12th Intelligent Systems Conference 2026,
3-4 September 2026 Amsterdam, The Netherlands

Agentic N-Way DataSyncs for IT, Manufacturing and Quantum Optimization

Tammy R. Fuller[1] and Gerald E. Deane[2]

[1] Chief Architect/VP, Echo Messaging Systems, Inc., 198 Chapel Street, Lincoln, Rhode Island 02865, USA
tammy@echomessaging.com
[2] CEO/President, Echo Messaging Systems, Inc., 198 Chapel Street, Lincoln, Rhode Island 02865, USA
gerald@echomessaging.com

Abstract. Data synchronization is a challenge that has been around almost as long as digital processing. Sharing data using OS file systems or data sources like RDBM's is insufficient with limited options. API's are a standard offering, but the work of sharing data efficiently, accurately, and dependably over systems' lifecycles lacks support beyond basic API access. With expansions in IT, Automation and AI, there are many best-in-class applications to leverage, but problems arise in orchestrating larger systems built from these components. By applying agentic processing techniques, we show how to utilize data beyond their original confines, by sharing among disparate systems by forming data bridges, while also tackling the massive problem of migrating ancient IT systems and outdated manufacturing operations into modern infrastructures. Agentic processing promotes a "divide-and-conquer" approach that reduces complexity by breaking problems down into smaller tasks, defined by rules for activation and their associated action responses. With myriad agents, each devoted to solving their own part of the overall problem of sharing and migrating data across disparate systems, APIs, and hardware/firmware boundaries, we show working in an agentic framework is an effective and efficient processing solution.

Keywords: data, datasyncs, data transfer, agents, multi-agent systems, data bridges, quantum optimization

1 Introduction

Without data to fuel information technology systems, AI, automation, gaming, entertainment, manufacturing, healthcare, and more, then the entirety of digital realms become useless. Data created in applications, morphs and evolves as it lives out its digital life, sometimes forever in one application or data source, sometimes moving between systems as it serves its important purposes of defining, describing and maintaining the status of things both virtual and analog. However, more often than not, data is restrained and held back from reaching its full potential due to limitations that constrain it from reaching beyond their original environments. Synchronizing data between digital systems, where replication occurs to the extent needed in other IT systems, or forwarded via notification systems to trigger downstream actions is deceptively simple. Protecting data, preventing it from venturing beyond where owners intend or for cases where data holds personal information or protected data delineated by digital rights management systems [1] or protected by laws such as GDPR and HIPAA is counterbalanced with the usefulness of the data distinct and unto itself.

Taking into account that data must be both protected at all times and leveraged, when possible [2], the practicality of bridging data between two or more systems is challenging due to the asynchronous nature of multiple IT systems, while maintaining data integrity within the boundaries of increasingly complex sets of IT systems. Systems run on servers with a physical location or set of

locations, such as server farms, underwater data centers, near-term orbital data centers, and even the computer in front of you. Time and space matter in data connections between physical locations even when applications share data on the same computer. Databases and operating systems have long incorporated semaphore, multi-tasking, and shared resource processing to managing shared resources. When applications share a data source such as a relational database system or files in a file system, built-in coordinated mechanisms protect data integrity and manage the process of sharing digital resources according to predetermined policies. When IT systems and applications share data not connected by shared data sources, mechanisms to coordinate shared resources have to be done, nonetheless.

As an extension of shared resource management inherent in databases and operating systems, we approached this problem with an agent-based processing framework and developed a multi-way data synchronization system between not only disconnected, but also disparate IT systems. Many issues are handled by myriad agents in this multi-agent system, including detecting new and updated data from potentially more than one data source, determining which systems get access to which data, interacting with source and destination IT systems at an API level, logging a full audit trail, and notifying any anomalies, warnings and/or errors to appropriate parties.

A set of independent, asynchronous agents, each tasked with solving one small piece of the larger puzzle, maintains data integrity across IT systems, employ best-of-class applications and is flexible enough to solve real-world problems and evolve with the overall lifecycle of the data solution. Agents are not only configured to solve their part of the problem but are also configurable through parameterization. Agents can respond to their environment and affect other agents, such as when data throughput needs spike, an agent can be cloned to increase processing capacity as needed. When no longer needed, agents can likewise go dormant or disappear until needed again.

Agents in our proprietary framework called ADIN™, which stands for Anomaly Detection and Intelligent Notification, as shown in Figure 1, have targeted functions such as adaptation tasks or communication tasks, are used in many applications where asynchronous processing, such as datasyncs and notification systems, benefit. When interacting with data from various data sources, agents simplify processes that require data alignment, interfaces with hardware/IoT devices and benefit from productivity optimization opportunities, such as those offered by quantum computers.

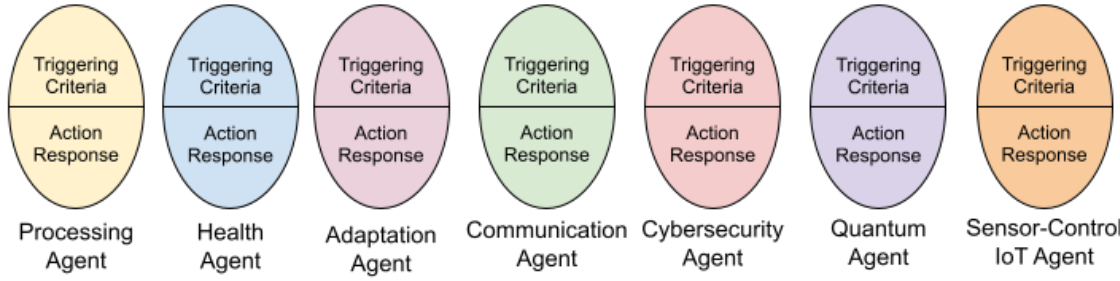


Fig. 1. Agents have various purposes, all based on the same two components – triggering criteria and action responses.

Bringing agents together in one place to address IT, datasyncs, manufacturing and quantum optimization with common connections to data sources, API, and user interfaces, as shown in Figure 2, defines an ADIN cell. Our experience with agent-based processing across many application domains have shown through repeated practical application of a multi-agent system reduces overall complexity to not only implement, but also maintain, with an added benefit to being responsive to evolving requirements over time, making ROI extend longer than non-agent-based approaches.

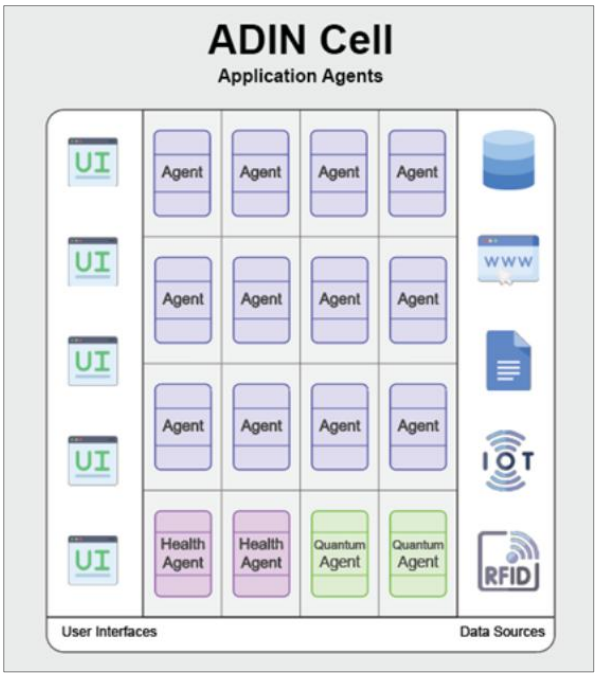


Fig. 2. ADIN Cell – a collection of related ADIN agents connected by API's, data sources and user interfaces.

Agents performing their various tasks are not limited in number, but by the computing capacity of the host and run containerized using Docker or other container systems with Kubernetes or other container orchestration systems [3].

To briefly review the ADIN AI Automation Platform, agents are programs that are configured with a triggering criteria and action response. The set of triggers and actions are preset, as well as expandable within the ADIN Automation Platform. Triggers are based on timing, location, and/or data. Agents' triggers are checked on a timer, on a schedule or on-demand. For Agents with access to GPS location either from a database or device, they are triggered on specific locations, upon distance travelled or crossing into or out of geo-fenced regions. Data triggers occur when a data field becomes a particular value, such as a status field becomes 'Ready,' changes value, or a new record is detected. Furthermore, triggers can be a combination of time, location, and/or data triggers. To configure an agent to fire when a new record is created, but only within a particular time window, and only when the status of another agent is set to 'Ready' is an example of a combination trigger.

2 Agent-based Processing Framework

Agent-based processing has proven be a viable approach to problem solving that reduces overall complexity, while increasing overall stability of complex IT systems. [4] [5] . Agents are small programs having two primary components – the triggering criteria, which activates the agent, and the action responses, the actions performed once activated. Rules that must all pass for an agent to activate are constraints based on any combination of data, time and/or space. For example, an agent activates when a new data customer record is recreated from an API-based interface. The associated action for that same agent is to create a new agent in a second system with identical contact information. This is essentially a simple one-way datasync where one agent is syncing all new customers from the source IT system to their destination. Another agent monitors when any customer record is updated and replicates the updated data.

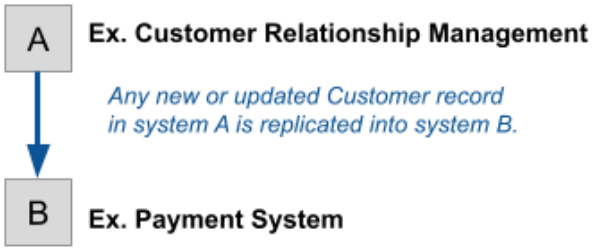


Fig. 3. One-Way DataSync

Data, being the most important aspect of any IT system, can also be limited in its use outside the confines of the IT system. Data in IT systems have limited ability to migrate, affect and otherwise interface with data from other IT systems. Architectures that allow for common data sources, such as data lakes or data warehouses, set up IT systems where the data is not confined within one system, but as its own entity. However, like data that reside within IT systems, data access is limited and highly controlled. API's provide access but are often not fully documented and can have incomplete features.

Widespread problems in aging architectures running critical systems for commerce and other industries must migrate to newer supported hardware before components, such as old mainframe systems, break that cannot be repaired. The challenge is even greater because migration must happen without interruption of day-to-day operations. Our Agentic architecture, where agents run in containerized environments, allows duplicate systems to run side-by-side, sharing data while new hardware, software and human interfaces are tested and proven.

Figure 3 above shows a basic one-way datasync where a Customer Relationship Management system (A) that manages all Customer Account data needs to copy all new and updated account information to a separate Payment System (B). If there are no built-in features to transfer data between these two systems, or if one of the systems is based on older technology, then options to implement datasyncs are limited.

ADIN Agents each have one or more action responses, such as creating a new record in an IT system, sending an email notification to an admin, activating an IOT device, or cloning an ADIN agent to run over high volumes of data, or any number of other actions. ADIN Agents are flexible and re-configured, which means their overall lifecycle is less taxing on developers and system architects, allowing for more time for key function development instead.

2.1 DataSync Agents

Because each ADIN Agent is composed of two sections: the "Triggering Criteria" and "Action Response", we assign an ADIN Agent to each ruleset and flow of data, as shown by the

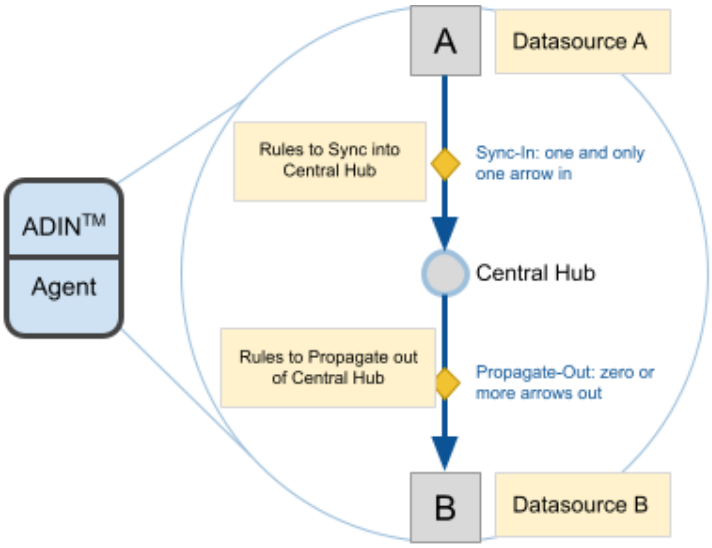


Fig. 4. Parts of an Agentic AI Automation Station

blue arrows below from one source data source synced into the Center Hub, along with the set of arrows propagated out to the target data source. These data sources, rules into and out of the central hub is called a Station. Many stations can run simultaneously operating over different classes and types of data.

For example, one station syncs Customer Account data between data sources, another station syncs invoices, and a third station syncs payments. Stations define the flow of one object type from one data source into any number of target or destination data sources according to a set of configurable rules as shown in Figure 4. We call the graphic representation a 'Hub and Spoke' architecture. Yellow diamonds indicate rules that include digital rights management and other legal IT systems that may be invoked for the action of the activated agents to proceed.

Station rules define the conditions and configurations that may vary either over time or from user to user. For example, a sync-in rule can determine if only new records are processed or new and updated records are processes.

3 Hub and Spoke

The Central Hub is key to our Agentic N-Way datasyncs working by providing a central place where data, keys, status, configurations, and more are stored and maintained. Web applications and client applications provide interfaces into the Central Hub that give insight into results and status, as well as a place to configure rule options. Interfaces are shown in a later section called "Interfaces - Web and Client Apps".

Even one-way datasyncs between two disparate systems can be a challenge beyond the most basic requirements. Mechanisms to detect something new or if something important changed, how often to check, matching records between systems that represent that same account, running asynchronously, data gaps from webhook event processing and many other challenges that are common hurdles are built into foundational processing levels of every agent, so with new API's the focus is on connecting and working the the data rather. Using a "Hub and Spoke" approach with ADIN agents assigned to each station, we can create sophisticated solutions were moving data, replicating records, syncing key items between systems can be built into large IT systems.

A one-way data sync allows companies to grow and scale their existing IT systems using various proven solutions and data sources without being forced into enterprise-wide solutions that provide umbrella-coverage to the enterprise but may not necessarily provide the depth or key features needed. Companies that grow are often forced to migrate to enterprise ERP systems in order to scale but then are dependent on costly custom buildouts to cover missing features. Options in the ERP system are limited and for smaller businesses cost prohibitive.

3.1 Migrating Old to New IT Systems

The challenge with updating new technology is how do you get the old data into the new system. If the new technology doesn't provide tech support or import features, this can be a challenge. Often, it is best practice to run old and new systems concurrently to ease transition challenges, such as detecting missing requirements in the new system, or comparing results from the new technology against the old technology to verify requirements are met.

By employing a two-way datasync as shown in Figure 5, data collected in the old system A will replicate into the new system B. As features are tested in B, data is replicated back into A allowing for an overlapping usage. Initial testing will require that A is backed up and restored into a sandbox environment. System B is setup as initially blank.

Running just the one-way datasync initially and setting rules out of A based on a new data from the Central Hub's point-of-view going back to a launch date will seed system B with replicated data from A's sandbox system. Next, turning on the second datasync so that new data from system B will replicate into A. Testing data integrity according to a predetermined test plan can be run over both data sources. Discrepancies and errors will allow for a correction to System B, and a full restart of the sandbox environment. Once the sandbox passes the test plan, this process can be done one more time, but the sandbox becomes the new production environment, where both the current and system new systems will run concurrently if desired.

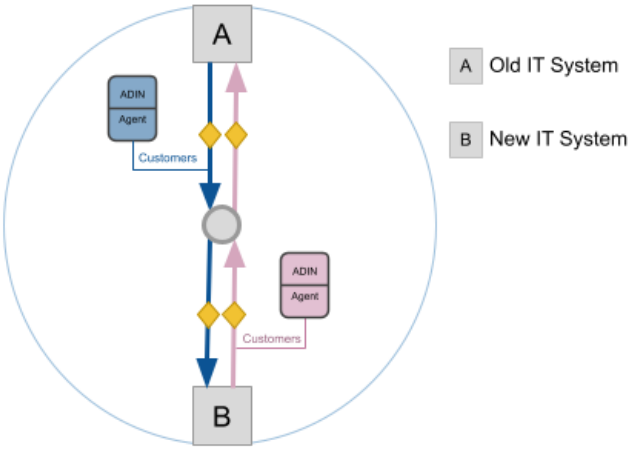


Fig. 5. Two-Way DataSync, two ADIN Agents

Older architectures on mainframes that used older languages like COBOL and Fortran are at risk of causing harm to companies through the disruption of IT services as these older technologies become harder and harder to maintain We help our customers migrate to new systems migrating the old data by using containerized Agents.

Containers such as Docker allow each ADIN agent to run independently and asynchronously from all other agents. Agents can connect to the same data sources. For migration tasks, ADIN agents first extract and relocate the old legacy data out of the confines of the old IT system. Our agents work with all API's but for older systems, connect directly to databases and work with the lowest common access points such as flat files, directories of data, and with information on proprietary systems extra directly from binary data sources. Once mapping rules are established, any number of agents running in their own containers can chip away at the job of migrating legacy data into a more modern data environment, such as relation databases, data warehouse and data lakes. API into the target data source will be used to complete the old to new data mapping tasks.

Once the process of extracting and migrating data is established, with containerized ADIN agents, the process can be replicated over and over, as new systems and architectures are developed and tested. New systems that work on migrated data can be testing as many times as needed, replicated to as many data locations as needed. Once the new system requirements are met and final migration of data from the old system to new will give the new system a full history of the legacy data. The new system can run concurrently to the old for a period of that that architects feel is necessary for meeting all testing milestones.

3.2 Multi-Way DataSync

Our "Hub and Spoke" architecture for agent-based datasyncs means that more complex data streams can be configured between any number of systems.

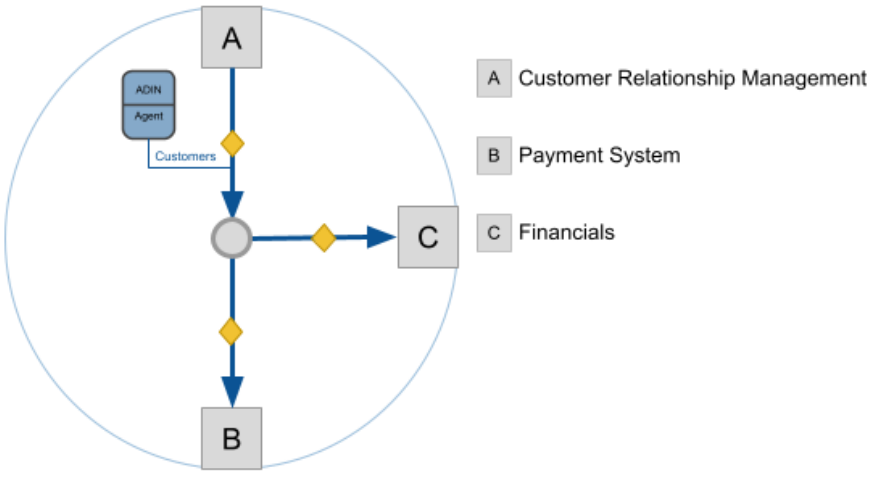


Fig. 6. Multi-Way DataSync

In the example shown above, Figure 6 shows an ADIN agent is configured to sync new and updated Customers from a Customer Relationship Management system into two other systems, one a Payment System and the other the company's financial system. All customer information is updated in one place in the CRM and further updates to the other two systems are done automatically. Triggering ADIN agents into activation can be done a number of ways. Using the DataSyncHub web application (shown in Section 4), Stations (and their associated ADIN agents) can be run on a timer, on a preset schedule or on demand. Timer-based processing means that every "X" minutes on an interval, where "X" can be set in the user interface, the source IT system is checked for new and updated data. Frequent checking means that data quickly syncs between systems.

Webhooks, if available by the source IT system, can be employed to handle events as they occur, caching them into the central hub for processing on the timer, schedule or on-demand. ADIN Agents are not limited to only these three processing types. The 'Triggering Criteria' section of all agents can be activated by time (timer, schedule) or by data (a button press) or any combination of triggers involving data, time and space.

3.3 Complex Application using Agentic Multi-way DataSync Stations

Building on the idea that multiple agents can run simultaneously, with syncing different types of data among two or more IT systems, we can architect complex solutions such as the following example based on an actual client solution, as shown in Figure 7. The main system is (A) Customer Relationship Management system. This is where employees use and maintain customer information, including orders and if account payments are up to date.

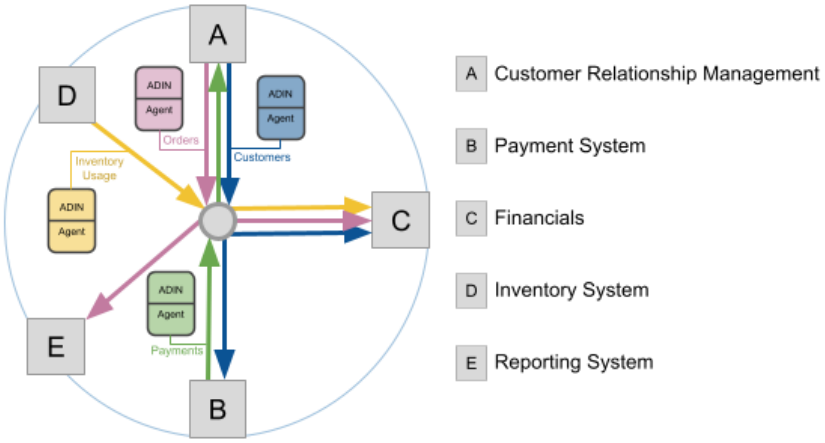


Fig. 7. Stations are color-coded and data of each type is processed by one ADIN Agent

Access to the (B) Payment and (C) Financial systems are restricted within the enterprise for best practices, so customer data in sync the CRM into the Payment system for automated payment processing and into the financial system for accountants to perform all bookkeeping tasks. An inventory system (D) is used by the warehouse to track items received from vendors to make orders. And finally, a (E) Reporting system gathers information for forecasting and advanced AI analytics, which aren't offered within the financial system, but serves as a key tool by the corporate team for deep insight into the health of their business.

Without asynchronous, autonomous agent-based processing of N-Way datasyncs, our client's administrative team would spend hundreds of person-hours replicating data or making use of inadequate tools in the CRM or going without key insights into their business. Once Agentic N-Way datasyncs are set up, audit logs are produced for all syncing activity, and exception reporting shows where data may need human intervention. Additionally, Notification ADIN agents can be added to datasync systems for automated alert-based notifications in real-time.

4 IT Applications and Interfaces - Web and Client Apps

The Central Hub is where coordination of the various ADIN Agents happens and where users can, at any time, see results. The Results Audit Log interface includes color coding, which is tied to user interactions, shown in Figure 8.

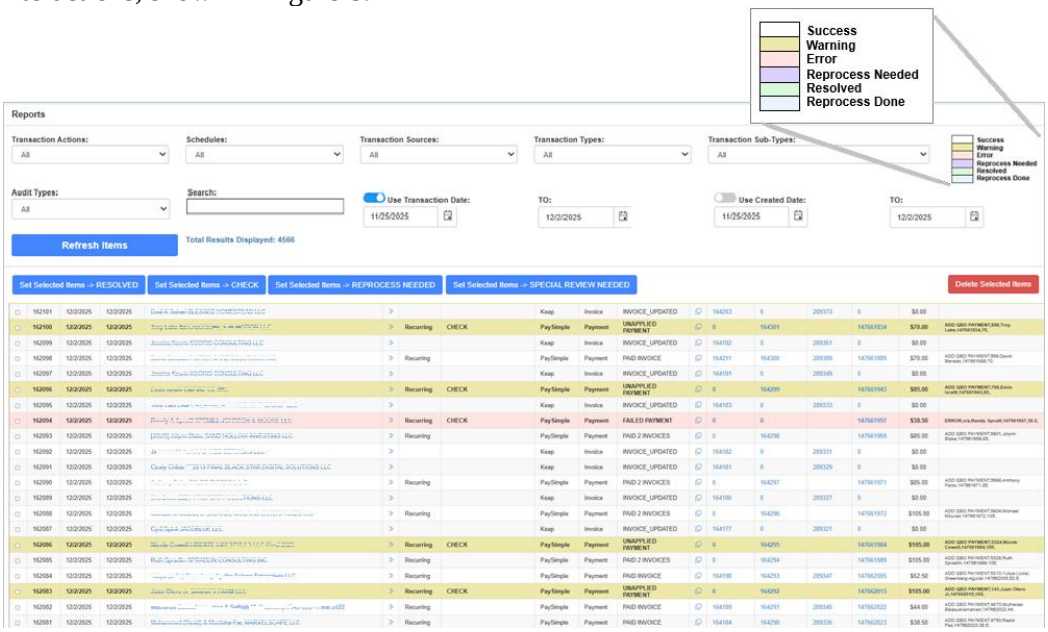


Fig. 8. DataSyncHub Web Application – Audit Results

Users can filter results based on errors or warnings, the source of the data (CRM, Payment, Financial), by type of data being synced (customers, invoices, payments), the subtype if an error or warning occurred, date range for both when the result log was created and when the transaction took place.

Result status can be updated from ‘Check’ to ‘Resolved’ by the user, or from any status to ‘Resolved’ or ‘Needs Special Review’. These records become color-coded for administrators to handle any special cases or issues that need resolving within individual IT systems.

Once an issue is resolved, users can set records to ‘Needs Reprocessing’. Customers, invoices, payments and any other datatypes that are synced, will be reprocessed the next time agent processing occurs. Once reprocessing is completed, the record status is updated to ‘Reprocessing Done’.

The various stations are controlled within the DataSyncHub web application as shown in Figure 9. Each station has setting from the source into the Central Hub and from the Central Hub out to the targets. Each can be individually turned on/off. Also, for this implementation, objects being sync can be limited to new records order or new and updated records. For each station, a process type is set of TIMER, SCHEDULE or ON-DEMAND. These can be set at any time. Once configured and testing is done, setting stations to run on a TIMER is the simplest. For the example, payments are synced according to a specific date. This is to allow the financial system control over what days are synced in.

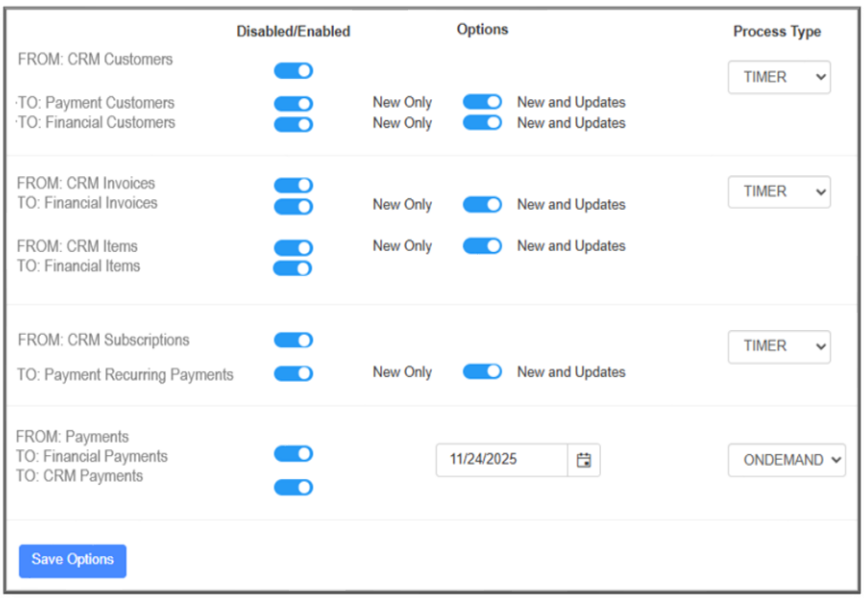


Fig. 9. DataSyncHub Web Application – Configure ADIN Agents rules and processing type.

5 Manufacturing Applications

In automated manufacturing applications, materials transform as they progress from station to station from raw materials to the finished product [6]. The state of the item being produced is its status and devices, such as pumps, crimpers, feeds, etc. are actions. Agents that sense real world status via sensors such as laser detectors and pressure switches are the source of triggering criteria for agents. The same processes in a multi-agent system used for datasyncs have been applied to automated manufacturing.

Automated manufacturing benefits from agent-based processing environments like our ADIN Automation Platform, where three approaches are presented. The first uses agent-based processing in an embedded microprocessor environment, the second shows how our traditional server-based container system can be applied to manufacturing applications and third, will show how to transform and improve existing manufacturing systems based on PLCs (Programmable Logic Controllers) configured using Ladder Logic by swapping in new approaches [7] using agent-based processing with IoT devices called Sensor Control Modules (SCMs). Figure 1. Types of AI Automated Manufacturing processing types

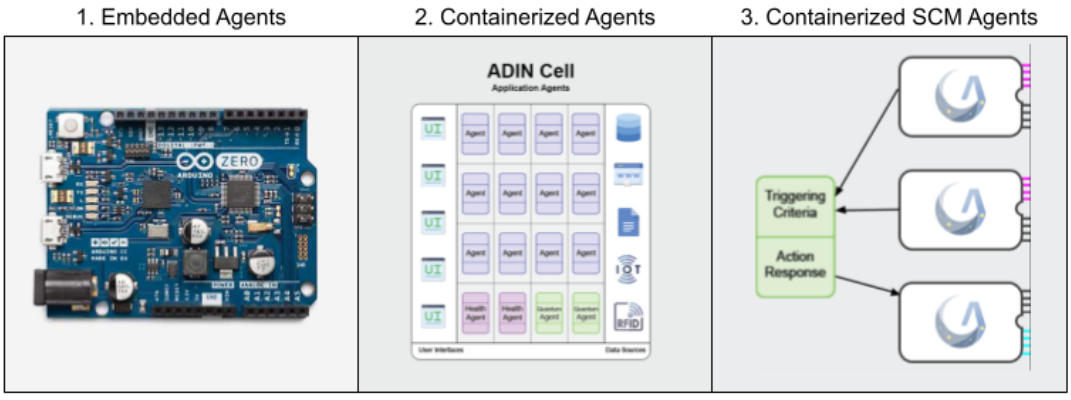


Fig. 10. Types of AI Automated Manufacturing processing types

The ADIN AI Automation Platform has a REST API which means programs written in any language can create and configure hundreds or thousands of agents across a large multidimensional parameter space. Agents run in Docker Containers. This means managing large sets of ADIN

agents is limited only to processing compacity and using systems like Kubernetes for container orchestration.

Once an Agent trigger is activated, agents execute action responses based on their configuration, which like triggering criteria is preset as well as expandable. Actions can be a new record in a database is inserted or updated. Notifications, where a templated message can be sent to specific people via any digital format, such as email or text message. Commands can be sent to IoT devices or can update and control other ADIN Agents. Actions can be combined, where many actions can be controlled from a single agent trigger or controlled via virtual state machine data graphs where the action first performs a function followed by a state machine update.

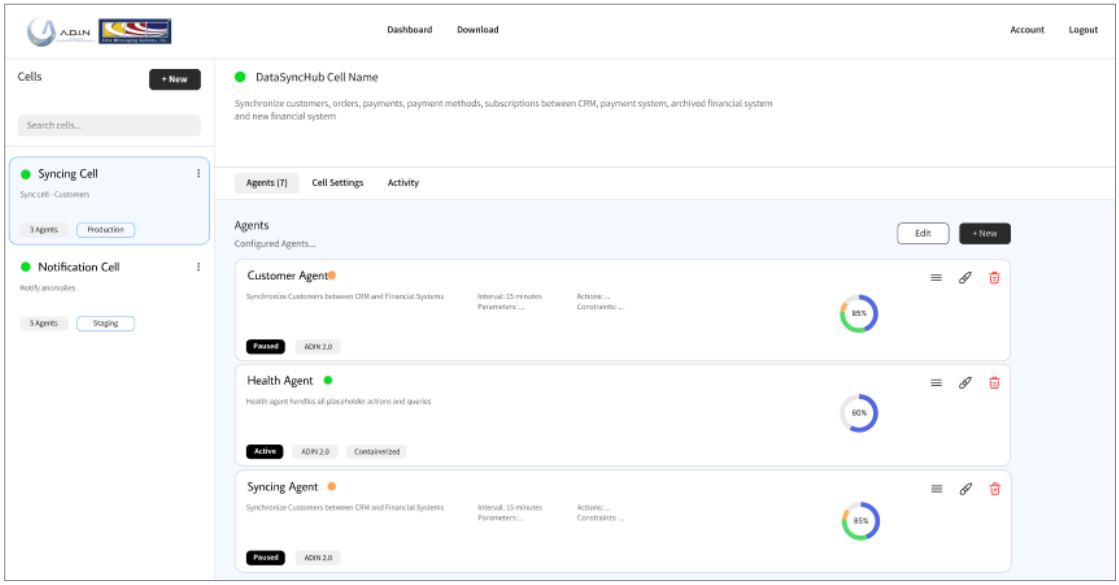
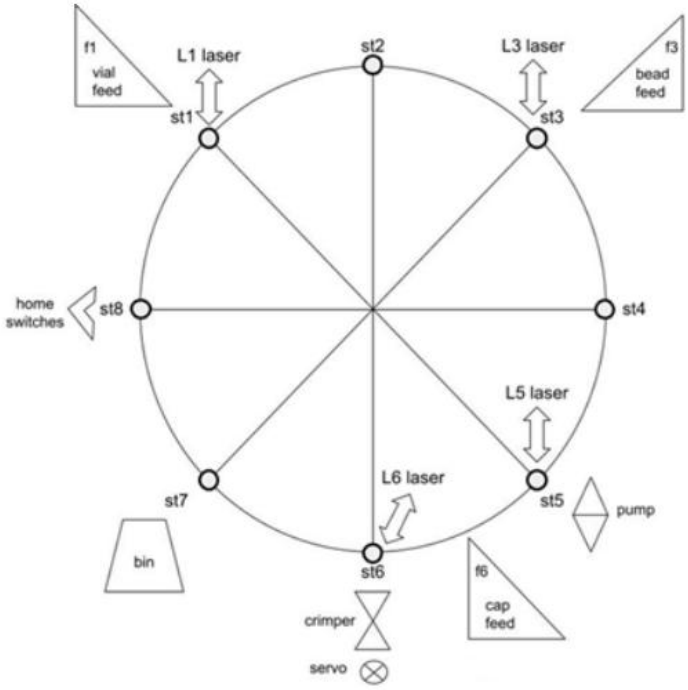


Fig. 11. ADIN™ User Interface

5.1 Embedded Agents

The ADIN agent source code was ported into the Arduino C++ variant where about a dozen agents were configured for trigger criteria and action responses and then run in



Fig, 12. Arduino ADIN Agents for each designated station

a single processing thread. This is a constrained version of the usual ADIN agent, where they run in a fully isolated containerized environment. On a typical Arduino single core board, only one processing thread can be run, and per Arduino's convention the function setup() is first called, followed by repeated calls to the loop() function. ADIN agents are configured in setup() where a separate ADIN Agent library is linked into the main program. Depending on the sensors and controls used by Arduino board design, each station (see 'st #' in Figure 3 below) has an associated ADIN agent C++ object, configure according to each stations' needs.

The Arduino board in this example had laser inputs and component feeds with motorized hoppers to keep components moving into a gravity-fed feed, several device motors, and a servo. An additional motor controlled a spinning platter turning it clockwise, and it had its own agent to monitor global information, where each station's actions needed to successfully complete to meet its trigger criteria.

Table 1 shows how this was accomplished in terms of each station's triggering criteria and action response.

Table 1. Configuration for Arduino C++ Embedded Agents for each station.

Station ID	Description	Trigger Criteria	Action Response
st1	Component 1 Feed	Is Feed full? L1 Laser switches closed?	No – turn on hopper, Yes- turn off hopper, F1 Motor until both L1 Laser switches open
st2	*Empty – Future use*		
st3	Component 2 Feed	All components present? L3 Laser switch	No – indicate missing component for future steps, F3 Motor
st4	*Empty – Future use*		
st5	Liquid pump	All components present? Previous station statuses and L5 Laser switch	Yes – run pump motor for preset time, P5 Pump
st6.1	Component 3 Feed	Is Feed full? L6 Laser switch	No – turn on hopper, Yes- Enable Servo to push vial into position for Crimper
st6.2	Motorized crimper	All components present? Previous station statuses	Yes – run motor for preset time, record crimp success/fail results, C6 Crimper
st8	Motorized platter	Crimp successful? Analyze feedback profile captured during Crimp	Yes – full $1/8^{th}$ turn platter forward; item drops into bin. No – partial platter forward, sound alarm for manual removal and inspection.

Every iteration through Arduino's loop() function, each agent's triggering criteria was evaluated. Some stations required timing based on time intervals where others had a continuous interval. For example, stations that operated a motor had separate on and off stages that were based on specific time intervals. Motors that fed component hoppers could turn on or off at any time based on the state laser switches points at component feeds. This meant that there were two main processing loops, one for continuous events and another for interval events.

This overall approach had many benefits. By identifying stations and treating each station individually, we successfully determined specific timing requirements to provide the optimal pumping action, servo control, and crimping control. Once each station was running optimally, we increased overall throughput resulting in greater quantities produced. This system replaced manual labor,

whose jobs transformed into overseeing the automated manufacturing devices, responding to ‘failed crimp’ events, and making sure feeds were fully stocked components.

6 Quantum Optimization

Quantum computers, especially those dedicated to addressing optimization specifically, like D-Wave Quantum [8] bring incredible opportunities for time saving functions that every application that involves resources and resource sharing can benefit from. Quantum computers are different from classical computers, like our desktop, laptops and servers in that the internal microprocessors operate on qubits (quantum bits) rather than bits (binary digits). Binary digits are operated on by Boolean logic gates that treat electrical signals as either a zero or a one.

Quantum computers have logic gates that allow for conditions where a gate’s value are both a zero and a one at the same time. D-Wave Quantum’s architecture is geared toward pragmatically reaching an optimum point over minimum overall energy. Imagine a landscape being surveyed and the lowest dip in the ground represents the best answer to a question that looks for the ‘lowest value’, like the least amount of time spent, or the lowest cost in materials or the shortest distance between all travel points.

The lowest point in a multi-dimensional set of parameters, such as cost vs. time vs. distance, can be represented as a long mathematical equation. This equation is submitted to the quantum computer designed to find minimum energy through a process called quantum annealing. Translating from a problem definition into the equation to find minimum energy is one of the tasks we programmed our agents to accomplish. The ability to optimize a data set, whether used for datasync or manufacturing control is available by adding agents that perform updates to data after the minimum energy state is found. Quantum computers perform this almost instantly, whereas classical computers increase exponentially in time requirements as data set size increases to where they cannot be used to solve optimization problems.

The focus of this paper is about how agentic processing solves the old, common, mundane problem of getting data from ‘Point A’ to ‘Point B’ where so work can get done. Quantum computers powered by agents running continuously review the shared dataset and depending on what needs to be optimized can feed data into data sharing systems for the best overall performance. Points ‘A’ and ‘B’ can be IT systems, but they can also be manufacturing stations. We continue to push the boundaries between the digital and the analog world with an eye toward overall optimization with the goal of improving productivity, reducing wasted time and making the best use of resources including human resources.

7 Conclusion

Every programmer eventually faces the problem of needing data from one system accessible in another system where built-in support isn’t available. IT systems often advertise a tight integration that is insufficient and lifecycle-based solutions are very difficult to achieve. Real-world applications usually resort to larger monolithic ERPs giving up best-in-class options and resign themselves to a limited menu of mediocre feature options. By devoting our research efforts to this stubborn, long-standing problem of allowing data to move unconstrained overcoming technical challenges, we open up possibilities in innovation by allowing companies to build solutions based on what is best suits their needs. Solutions are built by connecting subsystems, configuring rules and defining action responses in an agentic framework to form persistent data bridges. Agentic processing that incorporates quantum optimization reduces the complexity of sharing data while vastly expanding the value of data, in both software and hardware realms.

Acknowledgments. The authors thank Mr. Michael Izzo for his efforts in all phases of the technology developed described in this paper.

References

1. Armstrong TK.: Digital rights management and the process of fair use. Harv. JL & Tech, 20-49 (2006)Intelligent Automation with Quantum Optimization using Agent-Based Technology
2. Vertesi, J., Dourish, P.: The value of data: considering the context of production in data economies. In Proceedings of the ACM 2011 conference on Computer supported cooperative work, pp. 533-542. (2011).
3. Fuller, Tammy R., Deane, Gerald E.: Anomaly detection and intelligent notification. Future of Information and Communications Conference. (2018).
4. Hayes-Roth, Barbara: An architecture for adaptive intelligent systems. Artificial intelligence 72, no. 1-2: 329-365 (1995).
5. Briscoe, G.: Complex adaptive digital ecosystems. In Proceedings of the International Conference on Management of Emergent Digital EcoSystems, pp. 39-46. (2010).
6. Pulikottil, T., Estrada-Jimenez, L.A., Ur Rehman, H. et al.: Agent-based manufacturing — review and expert evaluation. In Int J Adv Manuf Technol 127, 2151–2180 (2023). https://doi.org/10.1007/s00170-023-11517-8
7. The Future of Ladder Logic, https://www.mroelectric.com/blog/the-future-of-ladder-logic/
8. Thom, M.: Solving problems with quantum annealing. In Digitale Welt 5, 70–73 (2021). https://doi.org/10.1007/s42354-021-0341-9