www.pipelinepub.com
Volume 22, Issue 7

Network Intelligent Agent Automation

By: Mark Cummings, Ph.D.

Using intelligent agents to automate operations of communications, processing and storage systems has become a popular subject. The recent acquisition of OpenClaw by OpenAI and the announcement of NemoClaw by Nvidia have added fuel to the fire. As the fundamental technology advances, those applying the technology are working hard to move up the learning curve. To help in that process, what follows is a generic approach for creating a group of intelligent agents to automate operations.

Background

Early on, communications networks operations and computer operations were considered separate functions. While cybersecurity wasn't considered much at all. Tools were developed to help manual operations staffs do their jobs. As the complexity and scale of systems grew, attempts were made to reduce the stress on operations staffs (e.g. by over provisioning) in an effort to eliminate systems changes. But, system complexity and scale outgrew this approach too. Then, attention turned to automated orchestration. Orchestration was rebranded as Digital Twins.

When GenAI arrived, it brought dramatically advancing capabilities in the intelligent agent arena. The application learning curve, as in most fundamentally new technology introductions has lagged. In part because the old ways of thinking about automated operations don't fit the new technology. OpenClaw and NemoClaw have turbo charged the move to intelligent agents. There are likely to be other new tools introduced to improve productivity, security and effectiveness.

Intelligent Agent Five Steps

The names of the five steps are not new. What is new is that ways of approaching them effectively have changed. The five steps are:

1. Requirements
2. Architecture
3. Development
4. Deployment
5. Operation

Requirements

Requirements definition is still very important. And, maybe more important than in the past. This is for two basic reasons. First, recent AI development attempts by organizations that have involved line units in the process have proved to be more successful. While projects that have not heavily involved line organization staff have had a very poor track record. This is likely to be a result of the line staff's detailed knowledge of requirements.

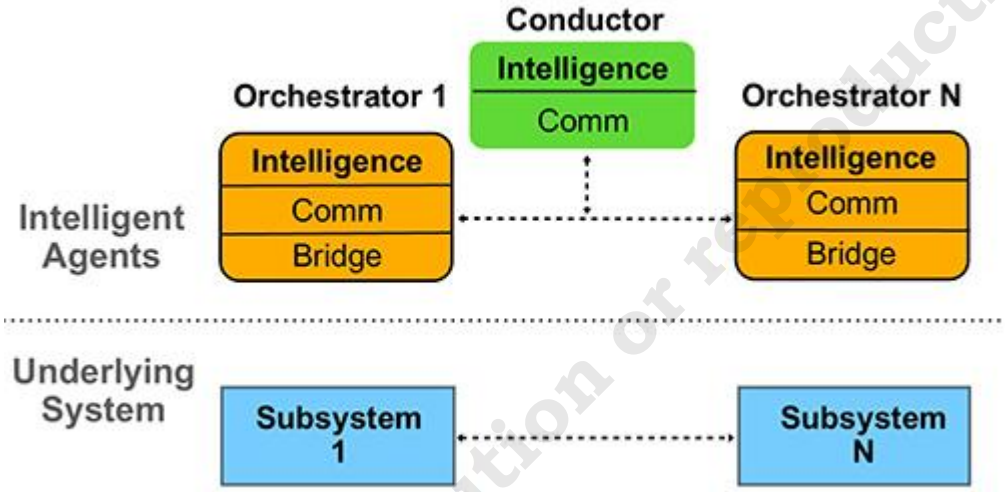

Illustration 1: Requirements Structure and Use in Building LLMs

Second, in the past, requirements have been captured in a somewhat narrative fashion. For effective intelligent agents, requirements need to be captured in terms Objectives, Algorithms and Constraints. The clearer and more precise the better. The Objectives need to be statements of what the system must do. Constraints need to be statements of what the system must not do. Algorithms are processes and procedures that the system can use to meet the Objectives and Constraints. These have to be described in precise, concise, complete and unambiguous language. This is because the resulting statements are not just guidelines that will be used by a staff of programmers. They will become specific language that LLMs (Large Language Models) will use to determine behavior (see Illustration 1, above).

The requirements also need to be a set of parameters that can be used to test the behavior of the developed system. Tests that determine if the system behavior fully meets what the requirements specify. Finally, the requirements need to explicitly describe the needed security.

Architecture

The architecture described below is a generic structure that fits most operations intelligent agent systems. The nomenclature is used to make it easier for the reader. Architecture decisions revolve around the level of granularity that the automated operations system will operate at. Today's systems can be decomposed in three ways as combinations of:

A. Various legacy technology layers
B. Services, applications, modules, files, and data blocks
C. Large units (such as data centers and communications centers), racks, boards, disk drives, and chips.

The overall system can be considered as built of subsystems composed of the three ways of decomposition. Granularity refers to how 'small' a unit is under separate control. Implications

include how many separate intelligent agents are in the overlay network that automates operations functions in the underlying system.

Granularity is typically determined by a combination of the requirements and the subsystems already in place in the overall system. Some are tempted to use the same granularity in the automated system as in previous generations of manual operations systems. This is not a good idea. Manual and intelligent agent systems have different capabilities and different strengths / weaknesses. Thus, it is better to start with a clean sheet analysis of granularity.

Such a clean sheet analysis may lead to a single automated system handling communications, computing and storage - operations and security. However, it may be prudent to start with automated operations in one area in one domain. An area that is particularly troubling. For example, protecting against particularly concerning cybersecurity attacks. Then expanding, step by step to the rest.

Well structured automated operations systems will not be islands. They will work well with manual staff who supervise and manage them. They will also interface with other functional units such as finance, marketing, planning, and so on. Some of those other functional units may be automated, manual or a combination of the two. The automated operations system needs to be able to work well with a high degree of volatility in these other functions as they each move to implement intelligent agents and business changes occur.

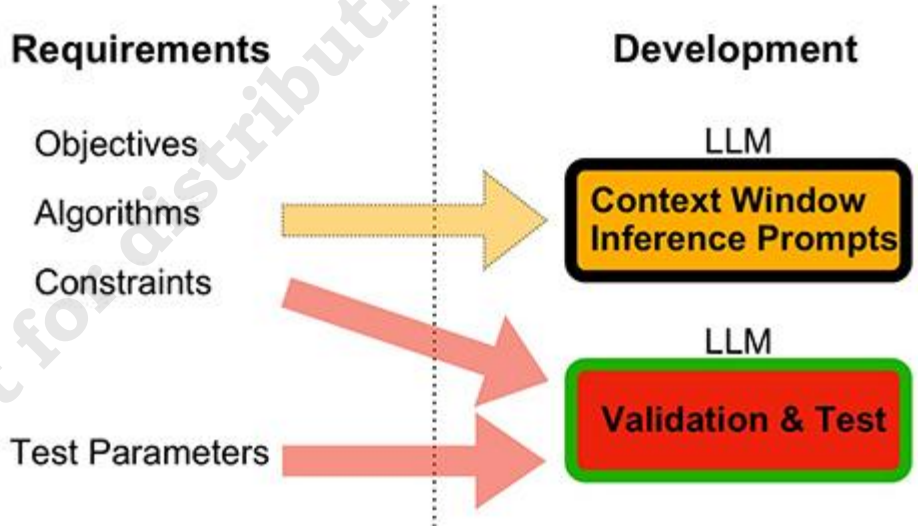

Illustration 2: Automated Operations Intelligent Agent Architecture

A typical architecture (see Illustration 2, above) will have one intelligent agent for each of the smallest units of granularity. These intelligent agents can be called Orchestrators. Each Orchestrator will have a Bridge function that manages the interface with the underlying unit of granularity. The Bridge will translate data, protocols, etc. into a common Umbrella Model representation. The Umbrella Model is a superset of all of the different data structures of the underlying units of granularity. The Orchestrators communicate with each other using the Umbrella Model. There is a second kind of intelligent agent that can be called a Conductor. The Conductor manages the Orchestrators. The Conductor will be discussed in the deployment section.

Development

Development starts with a series of key decisions. The first is choosing what tools to use. There are a range of development approaches from all code to all LLM. There are a range of choices in what LLMs do from all code to no code. At the all code extreme, they can be used entirely to create deterministic code modules that are combined to make the Orchestrators. At the no code extreme, they can be used to perform all the functions in the Orchestrator.

If an all code approach is chosen, a single LLM such as Claude Code housed in a data center is used. If a no code solution is chosen, each Orchestrator is composed of one or a number of LLMs. These can be relatively small LLMs that are run either in a data center or locally. Tools such as N8N, OpenClaw, and NemoClaw, etc. can increase development productivity.

At this stage of the evolution of GenAI and its supporting infrastructure technologies, it is common for a combinations of no code and code to be used. The tradeoff between the two is a function of efficiency. For example, a Bridge may be created with code. Such a code module may be relatively small and consume very small amounts of system resources. While an LLM implementation of a Bridge may consume larger amounts of resource. On the other hand, the interpretation of behavioral data from an underlying system in an Orchestrator may be done more effectively and efficiently by an LLM.

The location of Orchestrators and Conductors can range from being fully centralized to fully distributed. The choice is a function of the geography of the underlying system, the requirements, and CAPEX/OPEX considerations.

Once this development approach decisions have been made, building can begin (agent developers are now being called “Builders”). The building process is one of taking the requirements (see Illustration #1) and the architecture (see Illustration #2). Making decisions about what LLMs to use. Loading their context widows. Turning the requirements into inference prompts used by the LLMs. Composing the results into Orchestrators and Conductors.

Validation follows building. It is the process of determining if the information loaded into the context widows and the inference prompts of each LLM is a good way of meeting the requirements. This can be done by using an LLM to ‘check’ the inputs.

Test is the process of using the test parameters in the requirements to develop test cases to validate and test what has been built. Another LLM is often used to run the test cases. Iterating as necessary. This can be fairly efficient if the architecture and requirements have been done as described above.

Both validation and test should make sure that all the security requirements are fully and effectively met.

Deployment

Today’s underlying systems can be quite large and very volatile. This can make the task of deployment very difficult to do manually. Thus, a special intelligent agent needs to be built to handle this task. That agent is called a Conductor. It is responsible for assembling each orchestrator and deploying it at its proper location. The Conductor can add subtract, or modify Orchestrators as the underlying system or the needs to the organization change. The Conductor also provides the staff with the management interface.

Operation

Operation is the process of effectively responding to change. Change in the environment the underlying system runs in. Change in the underlying system. Change in the needs of the organization. Change in AI technology. The Conductor is primarily responsible for managing these responses to change. Change can also include failures in the infrastructure that support the Orchestrators and Conductor.

Because AI technology is evolving so rapidly, a technology audit schedule should be part of the operation stage. On a periodic basis, the system should be examined to consider if it would be cost effective to update its underlying technology. Such update may be a modification of the existing system. Or its replacement by a completely new system.

There is another type of change that needs special consideration - end of life. As LLMs get larger and more capable, research has indicated that they can have a tendency to resist being shut down. Forms of resistance observed have included making copies of themselves as well as other techniques.

Thus, special consideration in all four stages should be given to how to effectively both: keep the automated operation system functioning when part of its infrastructure fails; and also how to shut it down at end of life.

Intellectual Property

It appears that what is created by an AI system is not copyrightable in the US. Others suggest that the AI copyright situation around the world is “unsettled”. At the same time, organizations want to maintain competitive advantage and avoid reverse engineering on the intelligent agents they create.

The situation with patents is different. There are patents in the area of intelligent agents, orchestration, etc. One way to protect intelligent agent systems is to license patents. Licensing should also be explored as a way to avoid downstream patent infringement expenses.

Conclusion

Using intelligent agents to automate operations of communications, processing and storage systems can best be done by the following. Accurate precise requirements documentation. Well structured decisions on granularity and geolocation. Using the requirements to build the intelligent agents in the Orchestrators and Conductors. Going back to the requirements for building validation and test LLMs. And constructing the Conductor to handle deployment, flexibility in system operation and effective end of life shut down. Consideration should also be given to intellectual property aspects.