SPE 150285

Decision Support and Workflow Automation for the Development and Management of Hydrocarbon Assets Using Multi-Agent Systems

Amr S. El-Bakry, SPE, Michael C. Romer, SPE, ExxonMobil Production Company; Peng Xu, Anantha Sundaram, and Adam K. Usadi, ExxonMobil Research and Engineering; H. Lane Morehead, SPE, ExxonMobil Upstream Research; Mark L. Crawford, SPE, Bryce A. Holloway, and Charles A. Knight, ExxonMobil Information Technology

Abstract

Asset management teams face many challenges with increasing asset complexity, increasing data volumes, and staff in high demand while optimal asset performance remains paramount. More effective development and management of hydrocarbon assets may be achieved through an increased level of automation of work processes and decision-support technologies across the upstream value chain. Asset management workflow automation poses special challenges as such workflows are multi-disciplinary, cross-functional, and human expertise-intensive. Advancements in the fields of artificial intelligence, software engineering, and decision sciences have led to the development of Multi-Agent Systems (MAS) which have been the cornerstone of achieving higher levels of system and work process automation and autonomy. Here, the human expertise, obtained via knowledge elicitation of domain experts, is encoded into the software in an extensible and sustainable way. Each agent functions as an entity of a distributed computing system, performing a broad range of tasks which may include advanced analytics, data quality checks, and oversight of other agents. Although agents may have competing priorities, they can convey information to one another to broker a feasible decision through collaborative and coordinated decision making. This approach is useful for distributed real-time monitoring and capable of providing technical recommendations under open-loop environments and process control in closed-loop environments. This paper provides insights into a multi-agent development process for hydrocarbon asset management workflow automation and decision support.

Introduction

Modern hydrocarbon assets impose great management challenges due to their scale and complexity. The complexity manifests itself in many ways and derives from the spatial and temporal distribution of physical and production processes. Uncertainties inherent in the asset further add to the complexity. Assets may not respond to engineers' and operators' actions in consistent or predictable ways due to the dynamic and often incomplete information available from reservoirs, wells, facilities, and equipment. Critical information may not be available during decision making due to missing data or out-of-date predictive models. This problem can be compounded by network bandwidth constraints and the IT costs of integrating heterogeneous data sources.

** Exxon Mobil Corporation has numerous subsidiaries, many with names that include ExxonMobil, Exxon, Esso and Mobil. For convenience and simplicity in this paper, the parent company and its subsidiaries may be referenced separately or collectively as "ExxonMobil." Abbreviated references describing global or regional operational organizations and global or regional business lines are also sometimes used for convenience and simplicity. Nothing in this paper is intended to override the corporate separateness of these separate legal entities. Working relationships discussed in this paper do not necessarily represent a reporting connection, but may reflect a functional guidance, stewardship, or service relationship.*

Asset management best practice knowledge is highly distributed among engineers, geologists, and technicians. Some decisions require analysis drawing from multiple experts whose time is in high demand and who may be located in different geographic locations. Asset management includes routine activities such as well surveillance and time-critical activities such as fault mitigation. These activities, which may be interdependent, occur asynchronously at different time-scales and compete for attention and resources. As a result production uplift opportunities may be missed, equipment may be operated non-optimally, and urgent events may not be addressed in a timely manner.

To improve the efficiency and effectiveness of asset management it is essential to develop an integrated, flexible, and sustainable IT system that addresses these challenges. Such a system should integrate heterogeneous, distributed data sources and computer models (e.g., analytical models). It must process data automatically converting it into high-level human-understandable information. It must be capable of developing asset management plans, monitoring the execution of those plans, and adjusting them as and when necessary. Such a system should adhere to best practice work-flows and yet adapt to novel situations allowing for easy customization to address evolving asset needs.

A Multi-Agent System (MAS) is a technology and design approach that can satisfy these requirements in a scalable, flexible, and sustainable way. A MAS is composed of multiple interacting intelligent software agents, each of which is a computer entity that is capable of autonomous action within its environment in order to meet its objectives. Agents are persistent, autonomous, proactive, and social (Wooldridge, 2009). A MAS represents a natural progression and integration of several technologies from software engineering, decision sciences, and data analytics. MAS have been widely used in other industries for modeling and managing complex systems (Luck et al., 2003) such as power distribution (McArthur et al., 2007), transportation (Burmeister et al, 1997), and trading (Luo et al., 2002).

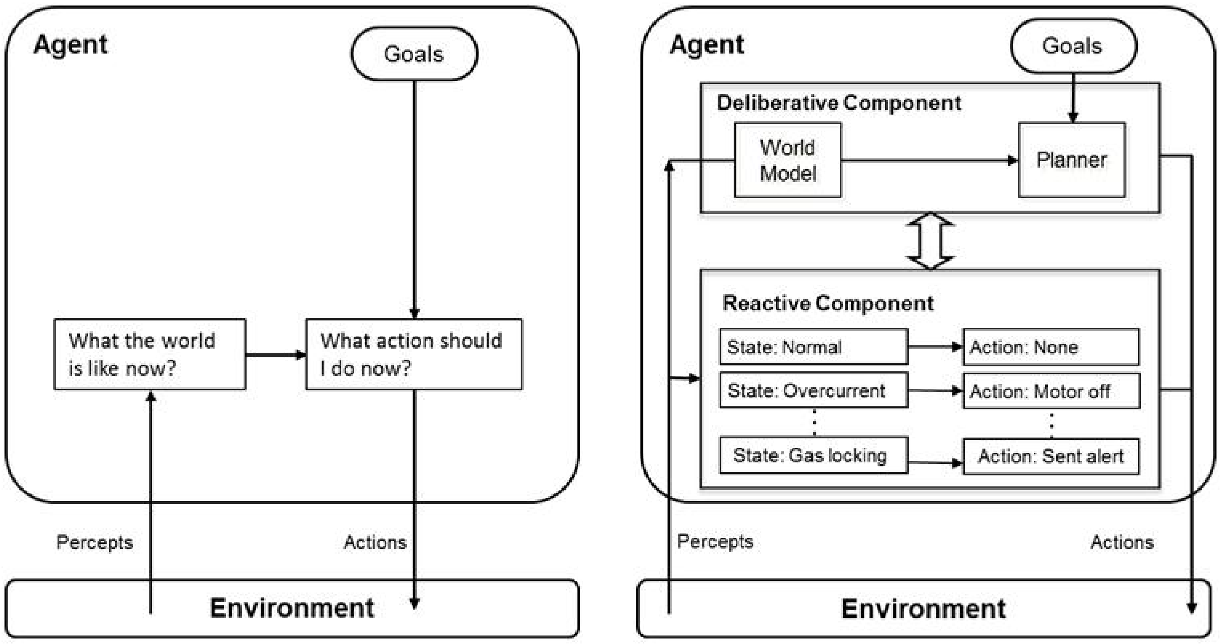
Early attempts to adopt multi-agent technology in the Oil & Gas industry can also be found in the literature. Engomo and Hallen (Engomo and Hallen, 2007) developed a proof-of-concept MAS that optimizes oil production for a simulated field. The agents analyze sensor data and control each well’s choke to meet their goals. They are able to satisfy all constraints, take actions to control sand content in wells, detect critical situations, and send alarms to human operators. Authors of (Olmheim et al., 2008) developed a similar system that manages field production under normal depletion conditions. Moreover, they reviewed previous StatoilHydro applications using multi-agent technologies, including a trading system and the Mongstad Jetty Planner. Multi-agent systems have also been used to monitor conditions in gas lift wells (Stephenson et al., 2010); the component agents collect and interpret data, assess wells’ conditions, recommend actions, and explain said recommendations.

In this paper we propose a process for developing a MAS for managing hydrocarbon assets and describe an example application of this process for select engineering surveillance workflows. The multi-agent system that perform analysis of sensor data and take actions based on its findings in order to optimize the production of a simulated oil field. The agent is able to adjust each well’s production, taking into account of local environment information. They also need to ensure that the global constraints are met. Local constraints (e.g., sand content limit) are examples.

Why a Multi-Agent System?

An intelligent agent is an autonomous, goal-driven, problem-solving computational entity capable of effective operation in dynamic and open environments. It is a natural extension of modern software design and development best practices. Figure 1(a) shows a schematic of an intelligent agent (Russell and Norvig, 2003). Agents can sense and perceive the states of the environment and certain agents have the capability to plan and conduct actions to achieve their goals. Figure 1(b) illustrates an agent using two methods to plan its actions. For simple scenarios an agent acts in a reactive mode; it directly maps the state of the environment to an action. For more complex scenarios an agent operates in a deliberative mode; it utilizes more sophisticated planning algorithms to identify a sequence of actions to achieve its goals. Note that the schematics in Figure 1 are just two examples of many agent schematics; readers are referred to (Russell and Norvig, 2003; Weiss, 1999; Wooldridge, 2009) for a more complete description of this topic.

Key enabling technologies for a MAS include 1) signal processing and pattern recognition for comprehension of data and situational assessment, 2) knowledge representation and inference for reasoning, planning, and decision making, and 3) communication and negotiation for constructing complex systems from smaller pieces. A learning capability is also crucial for many real world applications because it is difficult to be complete and accurate during the initial process of acquiring knowledge from engineers and domain experts. An agent with learning capability can update its knowledge by analyzing the results of planned actions. It ultimately corrects errors introduced in the knowledge acquisition phase and makes its knowledgebase more accurate and complete. Even if the initial knowledge acquisition is accurate and complete, learning is still important if the underlying environment (e.g., reservoir) is dynamically changing. Stone and Veloso (2000) provide a survey of multi-agent learning from a machine learning perspective.



(a) (b)

Figure 1: Illustration of an intelligent agent structure.

MAS provide a natural and easy-to-understand means of decomposing complex systems into manageable pieces. While each agent has clearly defined goals and capabilities to achieve those goals, agents may interact and cooperate to achieve system-level goals. This is true even if the individual agents have conflicting goals and compete for resources. A communication language and protocol are essential for agents to exchange information and resolve conflicts. A detailed introduction to agent communication is beyond the scope of this paper; readers are referred to documents on popular agent communication languages such as Knowledge Query Manipulation Language (KQML) (Finin et al., 1994), FIPA's agent communication language (FIPA), or the general introduction of MAS (Russell and Norvig, 2003; Wooldridge, 2009). In addition to communication language and protocol, a domain specific knowledge representation model is required for agents to understand each other. Knowledge representation, depending on application, ranges from simple forms such as a set of rules to more powerful representations such as ontologies (Allemang and Hendler, 2008). Brachman and Levesque (2004) provide a comprehensive introduction of various knowledge representation means.

To achieve its design objectives a MAS development team needs to decompose a complex system into smaller distributed systems, each with its own degree of autonomy. This requires a clear understanding of the underlying systems: e.g., their loci of control, their decision-making units, and their modes of interaction. For example, an asset management team works together to optimize hydrocarbon production through collaboration where each member assumes one or multiple roles with clearly defined responsibilities. One way to develop MAS is to analyze these roles and articulate the goals, capabilities, resources, and actions for each member of the team. The capabilities and actions that computers perform well may be encapsulated into one or more intelligent agents with corresponding goals and a means to interact with humans. One does not need to invent new concepts or procedures for asset management while implementing such a design; one just maps the appropriate existing functionalities into agents. Such a system may be constructed efficiently and, more importantly, is easy to understand and maintain.

Many other design approaches follow the same divide-and-conquer methodology. However, MAS possess several distinguishing features. Agents are autonomous and collaborate dynamically in ways that are not necessarily conceived at design. The autonomy of agents comes from the integration of sensing, reasoning, planning, and plan execution. An agent continuously assesses its environment, evaluates the outcome of its actions, and adjusts its plans accordingly. An agent with powerful planning algorithms (Ghallab et al., 2004) can assemble a limited number of actions to tackle various complex scenarios. More sophisticated behaviors emerge when multiple agents communicate and cooperate with each other. Consequently one does not need to pre-conceive all the possible scenarios the MAS may encounter during the design phase. A centralized software development approach, by contrast, generally requires one to conceive all the possible scenarios and hard-wire corresponding solutions. The flexibility of agents contributes significantly to the sustainability of a multi-agent system within a dynamic engineering and business environment.

The goal-driven nature of agents provides great flexibility for maintaining MAS. Agents send task-related requests to other agents when they are working together. The agent that receives the request will only perform such a task if it is useful for (or at least does not prevent it from) achieving its goal. The agent whose request is not fulfilled will develop an alternative plan to get the job done. MAS provide a natural extension of software encapsulation and a mechanism to scale up when new agents are added or refined—or to scale down when portions of the system are taken offline such as when deprecating old features or network glitches break the connectivity of the system. Mechanisms that allow new agents to register or broadcast their capabilities and health facilitate this process. Thus the system is more robust and maintainable than traditional software applications.

Agents run persistently, continuously, and proactively. They can automatically take action whenever new information is available, assets change behavior, or management goals are updated. And, as conceived in this paper, action is always in the form of decision support: a set of recommendations with explanations. The value derives from consistently elevating the quality of decisions to that of advanced domain expert—and applying that capability to a large number/variety of assets.

In summary, MAS provide an approach and a suite of technologies to satisfy the wide variety of capabilities needed for complex asset management.

Designing a Multi-Agent System for Workflow Automation

We view asset management from the perspective of a team of engineers who perform a variety of analytical workflows in order to optimize field performance parameters. Each workflow consists of a set of tasks that need to be executed sequentially or asynchronously. In this section we describe the methodology to identify workflows and develop agents to manage these workflows. We consider an example of a producing field where some wells require an artificial lift system such as a pump (e.g., electric submersible pump, rod pump). For the sake of simplicity some components such as gathering and separation facilities are not considered. The example is for illustrative purposes only and does not necessarily represent best practices for real world applications.

We consider the following workflows for an asset containing several hundred wells and associated pumps.

- Surveillance workflows: Predict and detect critical events, manage alerts, perform root-cause analysis, and propose mitigation procedures.
- Production optimization workflows: Maintain well production rates close to targets by performing online controls such as pump adjustments and offline enhancements such as well stimulations and pump upgrades.
- Maintenance workflows: Extend the life of the asset and reduce its operating costs through prompt execution and scheduling of routine maintenance workflows such as well tests and pump work-overs.

Design Methodology Overview

Our software system should be scalable, flexible, robust, adaptive to novel applications, and sustainable over time. That is, one should be able to minimize custom implementation for each and every asset, situation, and application. If the system itself is distributed, this requirement may be addressed by multiple technologies which pose certain software engineering challenges. These challenges include determining the required entities or components, their organization relationships, and their means of communication. In this section we describe an example MAS design methodology for a set of asset management applications.

The general design principle used here is loosely based on the Prometheus methodology (Padgham and Winikoff, 2004), though our emphasis is on decision making workflows, something not emphasized in the Prometheus methodology. Some of the steps have been modified to make them more appropriate for the artificial lift domain. The design process is structured towards an appropriate distribution of knowledge among the computational entities (agents) to achieve the design goals. Good asset management requires that decisions be based on sound practices and fundamentals; workflows and the knowledge required to address dynamic scenarios must be used as part of the design process as well. The following is the sequence of steps in the design process.

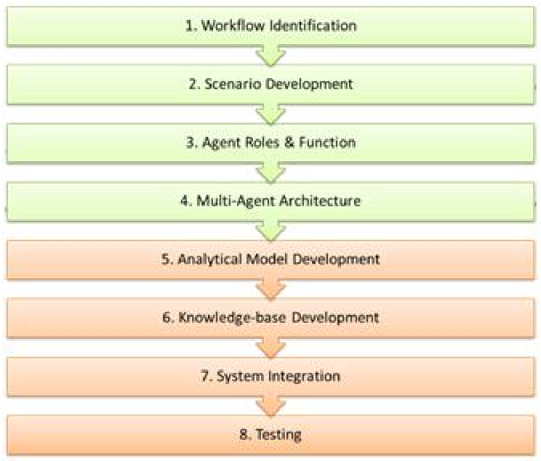

1. Workflow Identification
2. Scenario Development
3. Agent Roles & Function
4. Multi-Agent Architecture
5. Analytical Model Development
6. Knowledge-base Development
7. System Integration
8. Testing

Figure 2: A MAS design process for asset management workflows (only green-shaded boxes are discussed in this paper).

To illustrate the design process we will consider a set of workflows associated with a generic artificial lift mechanism. We will assume that such workflows are governed by generic practices which do not necessarily represent recommended best practices. In the following sections we describe a multi-agent implementation of Steps 1-4 from Figure 2. Figure 6 shows that scenario development (Step 2) and agent role development (Step 3) are integrated. This tight interaction will be demonstrated in the examples we describe as well. We will not describe Steps 5-8 in any detail. Analytical model development (Step 5) refers to the construction of supporting analytics that help the agents in decision support. While analytical model development (Step 5) is an important step, it is heavily customized to the asset and tied to possibly proprietary models. Analytical models in the artificial lift problem would include the well and reservoir flow models, the pump performance curves, and certain empirical models for root-cause analysis. The design of the knowledge-base (Step 6) can follow any of the well-documented methods in literature (such as belief-desire-intention graphs) and will not be described here. Finally, system integration and testing (Steps 7-8) relate to piloting the full application on a real asset and iteratively fine-tuning certain elements such as the roles of the agents and their collaboration.

Workflow Identification

Most asset management applications can be described as a set of workflows. These workflows include well surveillance, equipment surveillance, equipment maintenance, well test scheduling, etc. Most workflows have a hierarchical structure with nested subtasks that can be decomposed into low-level primary tasks. A primary task is the basic building block of a workflow that can be unambiguously executed by a computer entity (e.g., agent) or a human operator. Such a decomposition strategy has been widely used for many industrial applications. Example methodologies include Hierarchical Task Network (Ghllab et al, 2004) and Plan Goal Graph (Sewell and Geddes, 1990).

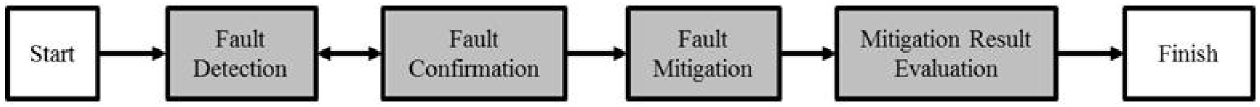
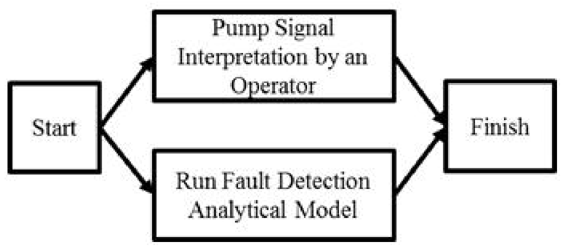


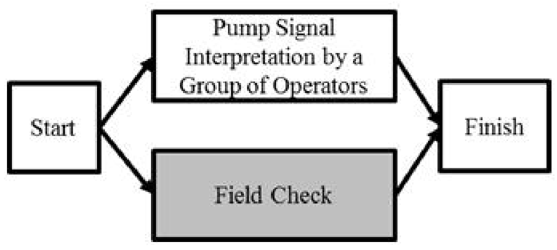
Figure 3: Pump fault detection and mitigation workflow (shaded boxes represent compound tasks).

Figure 3 shows a high level pump fault detection and mitigation workflow. The Start and Finish are pseudo tasks. The four shaded blocks between them represent compound tasks. Once a fault is detected it needs to be investigated to ensure that it is not a false alarm; this task is called Fault Confirmation. In case of a false alarm no other task but Fault Detection will be executed and the system will continue monitoring for faults. If a fault is confirmed a remediation recommendation will be made (Fault Mitigation) and follow up (Mitigation Result Evaluation) should be performed.

Figure 4 shows the further decomposition of the Fault Detection and Fault Confirmation tasks. Two methods to conduct fault detection are listed and only one of them will be executed: Pump Signal Interpretation by an operator and Run Fault Detection Analytical Model. The first one is conducted by an operator (human) and the second one is conducted by a computing entity. They are considered primary tasks (i.e., no further decomposition) because they represent unambiguous tasks that can be executed. Fault Confirmation can be decomposed into two alternative tasks; one involves human operators and the other conducts a field test such as a well test (Field Check). Field Check is a compound task that has to be decomposed into multiple subtasks.


Start
Pump Signal Interpretation by an Operator
Run Fault Detection Analytical Model
Finish

(a) Fault detection decomposition



(b) Fault confirmation decomposition

Figure 4: Task decomposition.

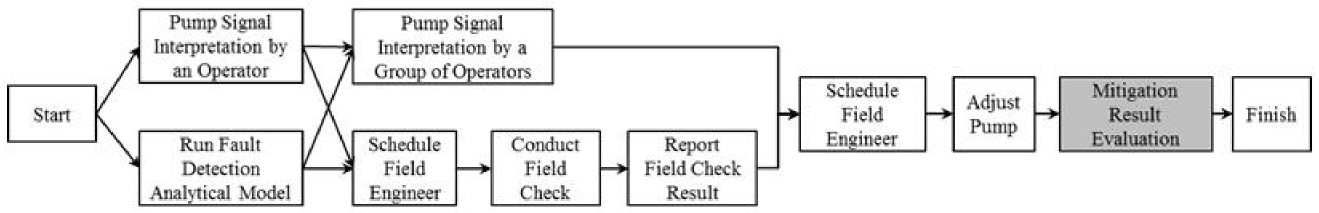
Several workflows may be active at one time during asset management and in an automated application they manifest as workflow instances. For example, multiple instances of Pump Fault Detection and Mitigation may exist in an asset management system; each instance would be dedicated to a single pump. The coexistence of these instances is not obvious if no faults are detected. However, they coexist (remain independently active) when multiple pumps encounter faults simultaneously or when a mitigation step is underway for a well with a previously confirmed fault. These workflows may depend on each other (e.g., the fault may persist as the well is being treated), compete for resources (e.g., mitigation treatment chemicals), and/or have conflicting goals. Scenario development is a natural way in which these dependencies and conflicts can be uncovered. It is an important step towards defining the agents, their interdependencies, and their collaborative capabilities.

Scenario Development

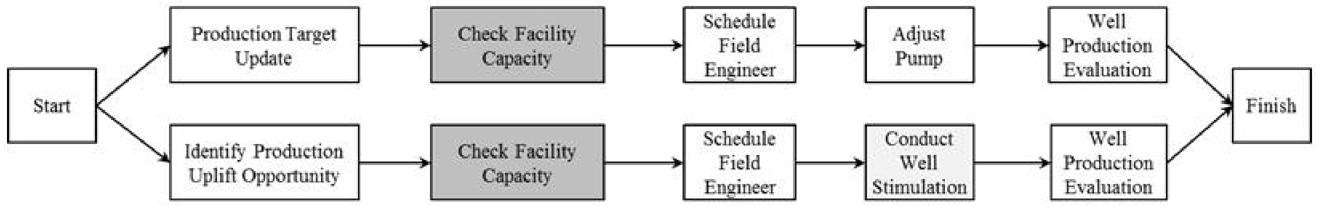
Scenario development helps consolidate and clarify workflow roles and the critical decisions they support. It uncovers domain dependent priorities that assist humans in making weighted compromises when resolving conflicts. Once the roles, priorities, and resolution paths are identified they can be modeled within a multi-agent system. The first step in scenario development is domain expert input, where different functional situations are laid out along with the sequence of tasks used to reach a resolution. For each scenario we consider the following items.

A. What workflows are involved?
B. What are the different roles that mitigate conflicts and help allocate resources?
C. What are the priorities to use in conflict resolution?
D. What new workflows will be triggered to achieve sub-goals?

To illustrate, we will provide an example where production targets have to be reconciled against fault mitigation goals. From Step A above we isolate two workflows with several compound tasks. These are shown in Figure 5. Figure 5(a) shows the Pump Fault Detection and Mitigation workflow with some of the primary tasks decomposed. Figure 5(b) shows a Well Production Uplift workflow which attempts to meet production targets and seeks uplift opportunities.


Start
Pump Signal Interpretation by an Operator
Run Fault Detection Analytical Model
Pump Signal Interpretation by a Group of Operators
Schedule Field Engineer
Conduct Field Check
Report Field Check Result
Schedule Field Engineer
Adjust Pump
Mitigation Result Evaluation
Finish

(a) Pump fault detection and mitigation workflow



(b) Well production uplift workflow

Figure 5: Two workflows sharing common tasks and competing resources (shaded boxes represent compound tasks).

Now consider the scenario in which a well with a fault detected by workflow 5(a) is also the well selected for a production increase by workflow 5(b). The Production Uplift workflow requests a speed increase from the Adjust Pump task while the Fault Mitigation workflow requests a speed decrease from the same Adjust Pump task. One way to resolve the resulting conflict is to detect that there is a conflict and defer to the human operator for resolution. In a real case there are hundreds of wells and numerous workflows and at any time one could be executing any branch of an active workflow. Deferring to the operator every time would leave him or her no better off than without the decision support system. We can try to identify certain independent roles that consolidate responsibilities in the domain so that priorities can be established. In this example the fault is detected at the well level, the production increase sought at the field level, and the mitigation plan includes other components such as field engineers and pump adjustments. To compensate we need different entities whose primary goal is the welfare of specific elements of the asset or the execution of critical functions in asset management: (a) Well (b) Production (c) Pump (d) Field Work Scheduling. Each task shown in Figure 5 has an entity responsible for providing input for and implementing the task as well as monitoring its progress. In Figure 5 it is clear that certain tasks can be consolidated as they refer to the same element or resource within the asset (such as a pump or field engineer). Each entity determines if it can initiate or perform the task it is responsible for and communicates this back to the requestor. Figure 6 shows the consolidation of these tasks and the responsible role assigned for executing that task.

The keys to resolving most conflicts are communication, prioritization, and collaboration. In this case an increase in pump speed will likely worsen the fault and damage the pump and therefore the well. We could prioritize that well health is more important than production uplift to resolve the conflict—the production management entity would either need to pick another production treatment plan (Conduct Well Stimulation) that does not worsen the fault or select a different but neighboring well for production increase (Adjust Pump). Figure 7 shows the chain of communication and one conflict resolution branch (Well Stimulation) accepted and implemented by the Well Agent.

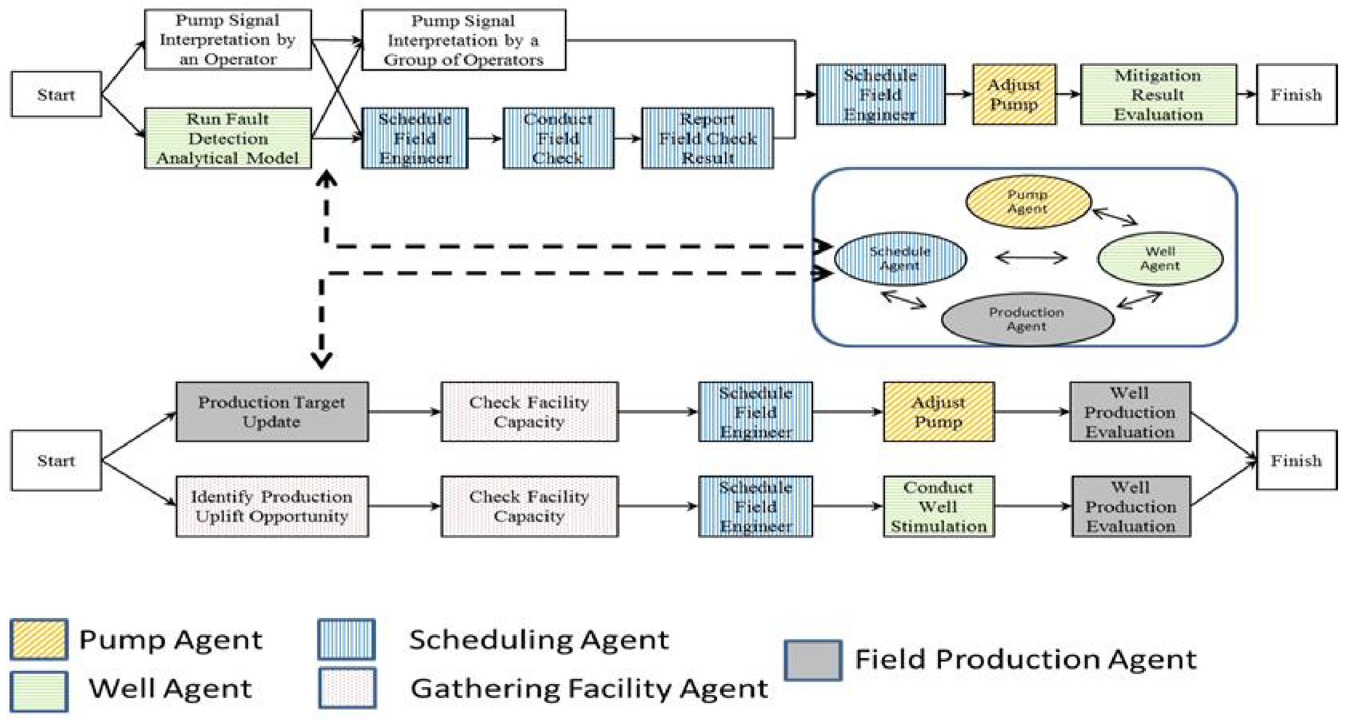


Figure 6: Consolidating tasks into roles for agents in production & fault mitigation scenarios.

The last step in this process is the addition of other roles and sub-goals as necessary. Figure 7 shows that Well Stimulation is triggered since the Adjust Pump request cannot be satisfied. This will include a new workflow which has the following tasks (a) Check available equipment (b) Check inventory for chemicals and equipment (c) Schedule injection (d) Monitor well injection schedule (e) Conclude workflow. Prior to requesting the Well Stimulation there may be a preliminary workflow to check if the facilities can handle the expected production uplift. This could be tasked to a different entity. These workflows now help recognize two more roles—(a) Well Work Scheduling Agent and (b) Gathering Facility Agent—and any related data-flows required to support them.

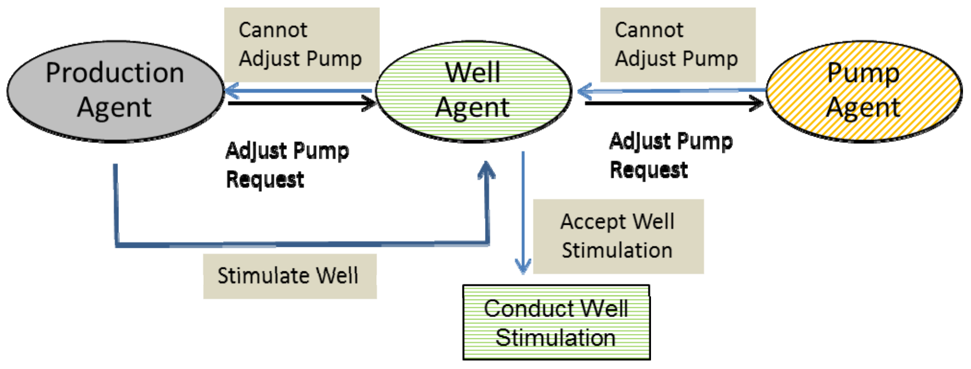


Figure 7: Identifying chain of communication and priorities to resolve conflicts.

Agent Design

A straightforward way to design agents for asset management is to consolidate different responsibilities determined during the workflow identification and scenario development steps. Role consolidation can be done by asset element or resource, by goal or function, or by some other criteria. In our example, aggregation by asset element leads to agents such as the Pump Agent and the Well Management Agent. These agents have access to all the knowledge required to operate and maintain the asset and the logic to make low-level decisions to resolve conflicts. Aggregation by goal or function leads to a Well Work Scheduling Agent and Production Management Agent. The Production Management Agent is responsible for one goal: “production uplift” from different sub-assets such as the gathering facilities, wells, or pumps. It decomposes its overall field level goal to a series of smaller goals that can be achieved by other agents in the system.

An agent uses various pieces of knowledge (e.g., analytics, physics, best practices) to arrive at decisions. Under special circumstances one may want to separately encapsulate certain proprietary or export sensitive know-how related to these specialties so they may be easily replaced with less sensitive ones. It makes sense under those circumstances to establish some agents as know-how providers so that only they need to be modified in applications exported to other countries.

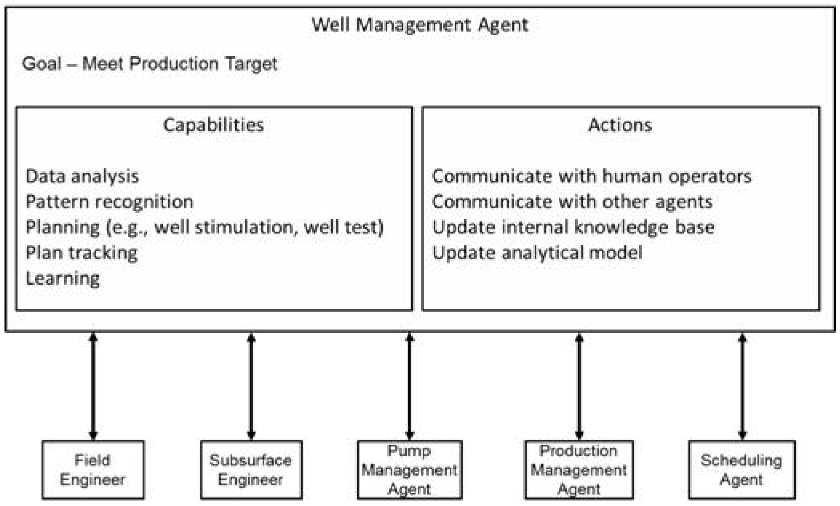

Well Management Agent
Goal – Meet Production Target
Capabilities
Data analysis
Pattern recognition
Planning (e.g., well stimulation, well test)
Plan tracking
Learning
Actions
Communicate with human operators
Communicate with other agents
Update internal knowledge base
Update analytical model
Field Engineer
Subsurface Engineer
Pump Management Agent
Production Management Agent
Scheduling Agent

Figure 8: A Well Management Agent.

An example Well Management Agent design is shown in Figure 8. This agent's environment is the state of the well itself as observed through a stream of data from an online data-base fed by sensors in the well. The Well Agent's overall goal is to ensure that the well's production target is met. Distinction must be made between the overall goal and the roles of the agent. The Well Agent will first need to ensure that the well is not under any abnormal condition to meet its production target goal. If there is a fault it needs to ensure mitigation is in place before planning to meet production targets. The Well Agent performs a dual role of fault mitigation and production uplift within the scope of the well. Even after aggregation multiple roles can coexist within an agent. The production target for a well is allocated by the field's Production Agent so the Well Agent needs to receive this target from the Production Agent. The Well Agent translates the target into a series of sub-goals to achieve that involve other agents such as the Pump Agent. The agent persists in activity as long as some of its goals are not met.

The agent must be able to monitor and determine the state of its environment, devise plans to mitigate abnormalities and/or meet its production goal, and communicate targets to/from other agents and operators. The Well Agent may need other supporting services to perform these tasks. For instance, pattern recognition methods, analytical well models, or empirical practices may be needed to confirm severe flow conditions such as gas-locking or sand migration. Each method can be designed as a separate web-service that is called by the Well Management Agent to confirm or refute the existence of specific well conditions.

Finally, the agents provide decision support to engineers and operators. The agents need to communicate results and provide reasoning to the human operators as well as solicit information. Figure 8 shows these components.

Conclusions

The design of effective multi-agent systems for asset management requires explicit documentation of corporate best practices, personal human experiences, and frequently experts' tacit knowledge. The lack of such documentation does not necessarily impact an experienced operator's performance, but it does pose great challenges for MAS developers who may not have in-depth domain knowledge. There are two approaches for MAS developers to overcome this challenge. One is to identify the implicit workflows from existing documents summarizing workflow activities. Another is to participate in the domain experts' routine surveillance meetings. Both approaches may need to be used in tandem to achieve the best knowledge capture results. In complex real-world applications the workflows likely interact with each other and make independent workflow identification challenging. Such a challenge stems from the complexity of interacting, and often dependent, nature of asset management workflows. Examples include the existence of multiple workflows that could achieve the same goal or parallel workflows with conflicting objectives. It is not trivial to identify an independent workflow to address a given situation. It is important to identify—and carefully document—the threads of knowledge and intentions underpinning different workflow steps. MAS can support human decision making by automating the management of such complex workflows. They achieve this through proactive, continuous, and autonomous processing of surveillance data, updating system models, analyzing system states, determining problem root causes, and providing targeted and informed updates to members of the asset management team. Furthermore, MAS provide a natural approach to both creating and sustaining implementation of best practice approaches.

In this paper we have discussed a design methodology for using multi-agent Systems for asset management workflow automation. Although we use one simplified artificial lift example to illustrate the idea, the approach can be applied to other production and development applications. We have applied the approach described to workflows including those to manage production where wells contain electrical submersible pumps.

References

1. Allemang, D. and Hendler, J. 2008. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Burlington, MA: Morgan Kaufmann.
2. Brachman, R. and Levesque, H. 2004. *Knowledge Representation and Reasoning*. San Francisco, CA: Morgan Kaufmann.
3. Burmeister, B., Haddadi, A., Matylis, G. 1997. Application of Multi-agent Systems in Traffic and Transportation. *IEE Proc. Software Engineering*, **144**(1): 51-60.
4. Engmo, L. and Hallen, L. 2007. Software Agents Applied in Oil Production. MS thesis, Norwegian University of Science and Technology (May 2007).
5. Finin, T., Fritzson, R., McKay, Don., and McEntire R. 1994. KQML as an Agent Communication Language. In Proc. 3rd International Conference on Information and Knowledge Management (November 1994).
6. FIPA ACL, http://www.fipa.org/repository/aclspecs.html.
7. Ghallab, M., Nau, D., Traverso, P. 2004. *Automated Planning: Theory and Practice*. San Francisco, CA: Morgan Kaufmann.
8. McArthur, S.D.J., Davidson, E.M., Catterson. V.M., Dimeas. A.L., Hatziargyrious. N.D., Ponci. F., and Funabashi T. 2007. Multi-agent Systems for Power Engineering Applications – Part 1: Concepts, Approaches, and Technical Challenges. *IEEE Trans. on Power Systems* **22**(4): 1743-1752.
9. Luck, M., McBurney, P., and Preist, C. 2003. Agent Technology: Enabling Next Generation Computing: A Roadmap for Agent-based Computing. Online: http://www.agentlink.org/roadmap/al2/roadmap.pdf.
10. Luo, Y., Liu . K., and Davis, D.N., 2002, A Multi-agent Decision Support System for Stock Trading. *IEEE Network* **16**(1): 20-27.
11. Olmheim, J., Landre, E., and Quale, E.A. 2008. Improving Production by Use of Autonomous Systems. Paper SPE 112078 presented at the SPE Intelligent Energy Conference and Exhibition, Amsterdam, The Netherlands, 25-27 February.
12. Padgham, L. and Winikoff, M. 2004. *Developing Intelligent Agent Systems*, Hoboken, NJ: Wiley.
13. Russell, S. and Norvig P. 2003. *Artificial Intelligence: A Modern Approach*, second edition, Upper Saddle River, NJ: Prentice Hall.
14. Sewell, D. and Geddes, N. (1990). A Plan and Goal Based Method for Computer-human System Design. In D. Diaper, et. al. (Eds). Human-Computer Interaction—INTERACT '90. Elsevier Science Publishers; North-Holland. pp. 283-28.
15. Stephenson, G., Molotkov, R., De Guzman, N., and Lafferty, L. 2010. Real-Time Diagnostics of Gas Lift Systems Using Intelligent Agents: A Case Study, *SPE Production & Operations*, **24**(1):111-123.
16. Stone, P. and Veloso, M. 2000. Multiagent Systems: A Survey from a Machine Learning Perspective. *Autonomous Robots* **8**(3):345-383.
17. Weiss, G. ed. 1999. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, Cambridge, MA: MIT Press.
18. Wooldridge, M.J. 2009. *An Introduction to Multiagent Systems* second edition, Hoboken, NJ: Wiley.