A General Model Based Engineering Approach To MRO Business Software Applications Using Acme

AbdElRahman ElSaid*‖, Hassan Reza†¶, Debesh Adhikari‡¶, Fatima El Jamiy§¶
Department of Computer Science, University of North Dakota, Grand Forks, North Dakota 58202¶
B. Thomas Golisano College of Computing and Information Sciences, Rochester Institute of Technology, Rochester, NY 14623‖
Email: aae8800@rit.edu*, hassan.reza@engr.und.edu†, debesh.adhikari@ndus.edu‡, fatima.eljamiy@ndus.edu§

***Abstract*—Maintenance, repair, and overhaul (MRO) services are an expanding market, which is expected to cross $61 billion in 2017. The market is expected to grow to about 28,000 aircraft by 2018. IT and software solutions should be at the heart of the automation and the optimization of such promising and sensitive industry. However, there is no generally-accepted software architecture to facilitate the MRO operations. The industry is dependent on architectures which are mostly propriety and have varying success and acceptance. A general MRO architecture could be of great significance, as modern aircrafts have a long life expectancy that can exceed 30 years, and definitely will need routine/none-routine MRO operations over their lifespan. Due to which, there is a growing need for a well-defined architecture to help reduce the costs, and standardize the overall MRO operations. Therefore, the goal of the stream of this research is to define and formalize a generalized software-solution architecture for MROs. This study proposes a general Client-Cloud architecture, reflecting the main MRO functional and quality requirements to enhance the business's overall performance and success. The study also made an effort to formally define the model, introduced by the proposed architecture, using Acme as an ADL[10]. This could lower the level of ambiguity and enhance the identification of the systems' components.**

I. Introduction

The unaddressed inherited needs for a time-saving, efficient, and cost-effective processing of the tasks and details of the business's parts is putting the business's stack holders in a restless sought for a modern and effective solutions. This target is achievable in todays technology accomplishments, where the digital evolution and the IT infrastructure can play a significant role. Those needs include, but not limited to: digital logging, information disseminating and security, accessibility, convenience, connectivity, communication and digital technology maintenance operations aid. In addition, the great AI achievements in other disciplines, as in image recognition [1], audio-visual emotion recognition [2], music composition [3] and other areas, are an inspiration for additions features in these solutions. Because solutions address reliability concerns, airworthiness evaluations concerns, and safety issues forecasting, as well as optimization solutions for the planning and scheduling tasks, AI's metaholistic software solutions can dominantly be efficient and productive.

Nonetheless, the 'Internet Of Things' (IoT) concept can be utilized in this industry to form a cyber-physical mesh that connects the physical assets of the business directly to the controlling application either for taking actions or for analysis to mitigate the risks. This latter step is actually starting to be a reality that will shape the future of the industry as large firms are adopting it.

However, to the best of our knowledge, no formal research conducted about a process-model or an architecture that blueprints an integral application dedicated to those MRO requirements other than those of proprietary nature, which claim coverage for some or most of those needs. Consequently, the MRO is striving for efforts to be done in this realm, which is a very rich research area supported by a thirsty market.

Embedding software solutions into the MRO organizational model will not only make the industry more accurate, precise, cost-efficient, time-efficient, and less error-prone, but definitely more safe, and this is achieved by eliminating the human factor error. Moreover, it will give the advantage of a more robust and liable reliability analysis. Consequently, considering all these parameters will reflect less-accident and more efficient civil-aviation operations, which shall be translated to more revenues and a broader budget margin for development and research.

II. Technical Background

Maintenance is classified into Routine (scheduled, preventive) Maintenance and Non-routine (corrective) Maintenance [4], [5]. Figure 1 shows a flowchart of the general aviation maintenance process. The description of the process is as follows:

A. Routine Maintenance

The routine maintenance is about a maintenance schedule or plan, the aircraft manufacturer designed for his product in a maintenance program. An aviation MRO adopts the manufacturer maintenance program and can customize it to its specific operations or regulations requirements. The program is arranged in particular checks that have precise intervals. The checks' intervals can either be *a)* Flight Hours *b)* Flight Cycle , or *c)* Calendar.

The checks are maintenance packages that hold maintenance tasks within. Each task is dedicated to a particular maintenance action (inspection/testing a system, replacement of a component, . . .). Maintenance checks are performed by executing the

[10]Architecture Description Language

978-1-5386-4112-5/18/$31.00 ©2018 IEEE

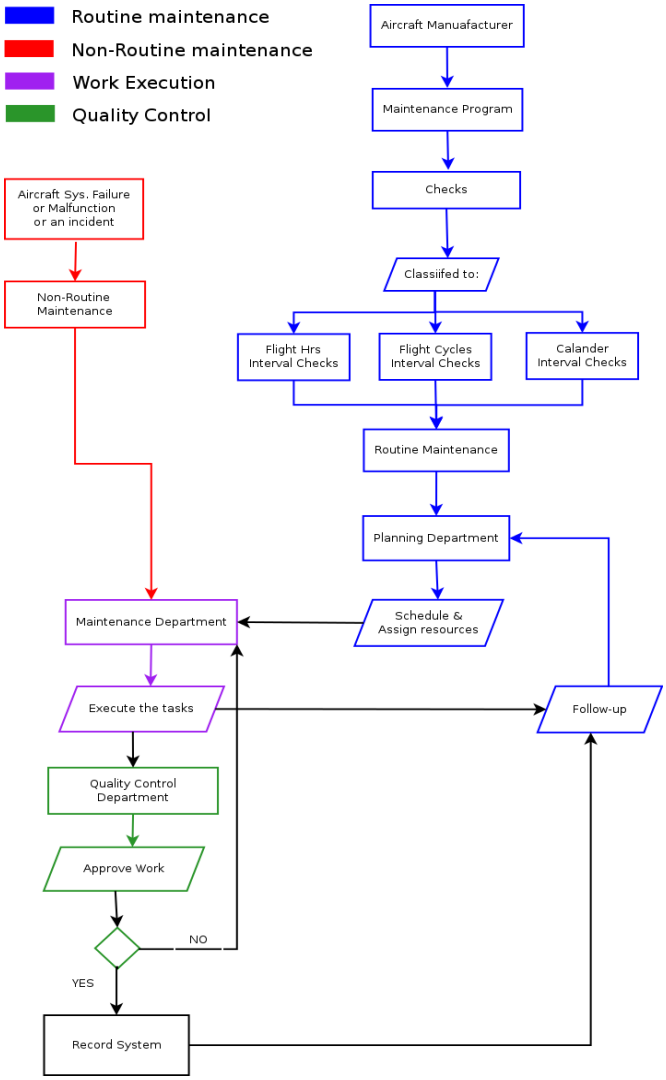


Fig. 1. Commercial civil aviation maintenance general process

tasks and signing it off. The planning sector in the aviation MRO organization prepares the work packages in the form of work orders that contain the maintenance checks and their tasks, according to the aircraft operating time, cycles, and/or time since in service, as specified in the aircraft maintenance program. As they release these packages, the maintenance sector moves to execute those work orders and sign the tasks off as they perform them.

When a task is signed off by maintenance staff, it should also be signed off by a quality control inspector to assure the quality of work done. This process is followed-up by the planning department to ensure its conformity with the check plan. Finally, the check with its tasks is then recorded. This type of maintenance is represented in Figure 1 in blue color.

B. Non-routine Maintenance

When pilots or maintenance staff report a technical problem and it requires some action, the maintenance department responds. The maintenance initiates a non-routine work order and handle the maintenance procedures and then closes the work order by signing it off by one of its certificated staff.

When the work order is signed-off by the maintenance department, it should be approved by the quality control department by signing it off by one of its inspectors. After that, the work order is recorded.

Sometimes the maintenance procedure needed to solve the technical problem does not need the aircraft to enter the hanger and it is performed on the apron. In such a case, the problem and the maintenance action is logged in the technical log book (TLB) of the aircraft, but also should be recorded in the record system after signed off by certificated maintenance staff.

This type of maintenance is represented in Figure 1 in red color.

III. Acme as an Architectural Description Language (ADL) Background

Architectural description languages are used to model a system in a formal way that is understandable to both humans and machines. They are developed to avoid the limitations of informal architectural representations which can be ambiguous, incomplete, inconsistent and unanalyzable [6]. ADLs can be used for the design of both software as well as hardware components [7]. Different types of ADLs for different purposes with different features and capabilities have been developed over the years. Some of the popular ADLs are Architecture Analysis & Design Language (AADL) [8], Acme [9], Rapide [10], Wright [11], Darwin [12], Aesop [13], Adage [14], etc. ADLs may differ vastly in representations. However, they all share a similar core and have a common foundation in the form of elements such as components, connectors, systems, properties, constraints, and styles [6]. Components can be used to represent both the high level and low-level representation of a system [7]. Connectors define the interaction among the components. Systems consist of components and connectors. Properties are the semantic information of elements. Constraints are the restrictions on elements. Style defines the design vocabulary and rules.

Acme is an architectural description language that was developed with the goal to ease the interexchange process between different ADLs. Thus, Acme consists of many of the common elements of various ADLs with an option for extensions. Despite that Acme not as powerful in expression as other ADLs, “the Acme language constructs do correspond to real-world software design concepts” [15]. Accordingly, the study mainly chose Acme for this significant properties. Acme is also used to model the high-level representation of MRO systems for the properties described below.

Acme consists of seven types of entities to define the formal architecture of a system. The seven core entities are [9]:

(a) **Components:** computational elements and data stores of a system
(b) **Connectors:** interaction among components
(c) **Ports:** interfaces to the components
(d) **Roles:** interfaces to the connectors

(e) **Systems:** graphs of components(nodes) and connectors (arcs) and are specified by attachments
(f) **Representations:** hierarchical detailed descriptions of components or connectors
(g) **Rep-maps:** correspondence between the internal representation and the external interface

Acme allows defining properties of Acme elements to complement the system description. A property can be defined by a name, an optional type, and value. Acme also supports defining constraints, which are considered a special types of properties. If a constraint is defined at the system level, all the elements of the system will also be under the constraint. Different domain-specific vocabulary can be defined by using types and families. The main limitations of using Acme is that there is no specific model to describe behavior and functional properties of elements. Also, it does not, as mentioned earlier, support direct mapping of elements to code. However, it can be translated to another ADL that can support this feature.

IV. Related Work

A. Proprietaries

1) Overview: Many corporations are involved in this intermediate growing industry. Frequently, there are new emerging companies that are offering new service which gives this industry other dimensions. Some of the very famous companies involved in this industry are IBM with MAXIMO, Swiss AviationSoftware with AMOS, and lately GE Digital with their cloud-based operating system, Predix.

Owned by IBM, this software was mainly used for service management but it was also applicable to maintenance management and was already being used by the jet engines manufacturing giant Rolls Royce. The software had some success but did not deeply impress the market [16].

Crossair's (former Swissair) IT division (later independent Swiss AviationSoftware Ltd.) was very active that it did not only manage hardware and software used by the airline but also made its own products. They developed an application to cover the airline's MRO management requirements, and later marketed this software as a product. They called their product AMOS, which became relatively successful as they have near ninety customers using their product. The software is based on client-server design and is considered one of the most successful MRO application [16].

General Electric, the giant industrial manufacturer, launched GE digital which leads the development of new breed of operational technology. By committing $1 billion to plant sensors on jet engines and other machines and then connecting them to the cloud, they will have a flow of online data flowing through those sensors so that they can analyze it to identify methods to boost airworthiness, reliability, and productivity. This kind of work is considered a type of operational technology where the software controls and monitors the machines except that now this work adds a new feature, which is data analysis. The cyber-physical mesh of sensors connected to the cloud is an IoT system. The IoT stream of data can be analyzed to forecast breakdowns and assess the monitored systems' overall health in fast pase.

The company's plan is to collect 50 million data variables from 10 million sensors installed on its machines, which requires a heavy platform that would be capable of connecting, securing, and analyzing data. They developed their own cloud-based platform; Predix, which is an operating system that can provide their customers with real-time information to plan the maintenance process and provide analysis results to improve the performance. Besides, this platform also allows the customers to use their own developed application to run on the system on their data. Eventually, this will open a new market for applications that can be hosted on the new operating system. Regarding data security, which is a concern for entities considering Industrial Internet, this aspect is addressed at all platform application layers: services enablement, data orchestration, and infrastructure layers [17].

2) Proposed Architectures: Regarding a reference architecture, which is the scope of this study, despite that there are quite some proposed architectures, none was accepted by the industry as a reference architecture. Two of those architectures found in the literature are:

- IBM MAXIMO architecture:
  The architecture is shown in Figure 2. Though the architecture is not accompanied with a description, the study of it shows that it is composed of two architectural styles, namely: modified blackboard style where a controller is orchestrating the other models besides the presence of communication between the modules, and client-server style. Although the system is used commercially, it looks like it will suffer from the inherited problems of the styles used [18]–[20]. On the one hand, the use of a blackboard architecture means that the system might get choked if the initiative module failed unless of course some sort of redundancy is used. On the other hand, the client-server style inherits the networking problems and a lot of consideration should be given while implementing such architecture, as whether the server is local and whether the network infrastructure can support the load of the work. Besides, the data module is not explicitly shown as part of the architecture, though it plays a vital role in the MRO systems.
- Swiss AviationSoftware AMOS architecture:
  The structure of the architecture shown in Figure 3 suggested that it is a mixture of layers and modules architecture. Though the architecture neatly shows the important sections of an MRO organization, the authors of this study believe that this architecture is very abstract and is not showing much of the system design. Based on practical experience with the system, the system can be accessed remotely, at least for data entry. Accordingly, at least this feature in the system is not showing in the architecture circulated in the literature. It is also worth mentioning that a pure layered architecture would suffer from issues related to performance, scalability,

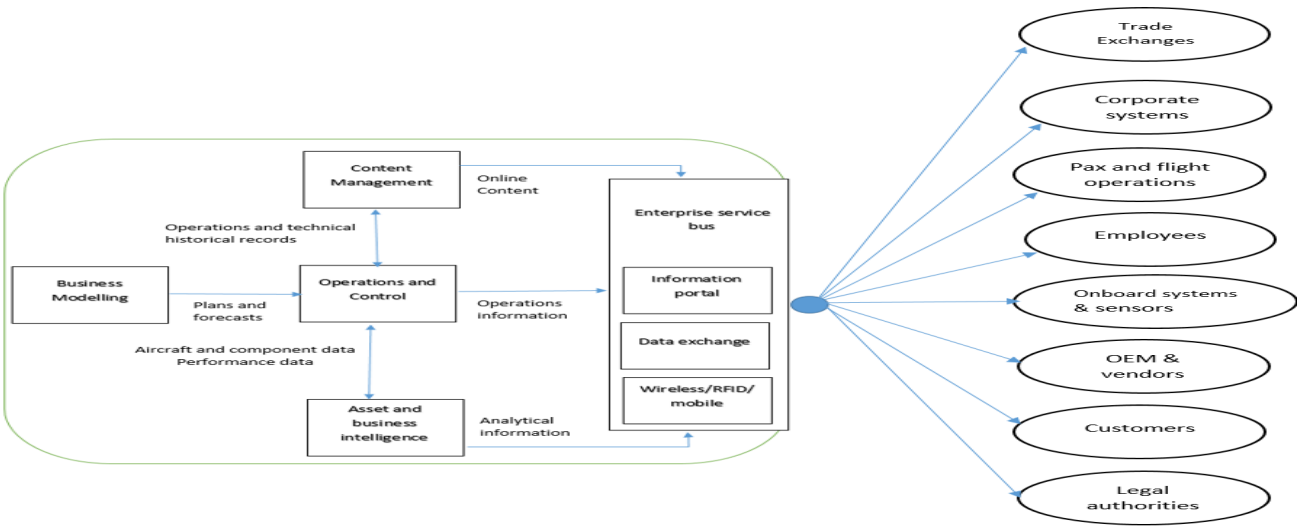


Fig. 2. IBM architecture (adapted from [16])

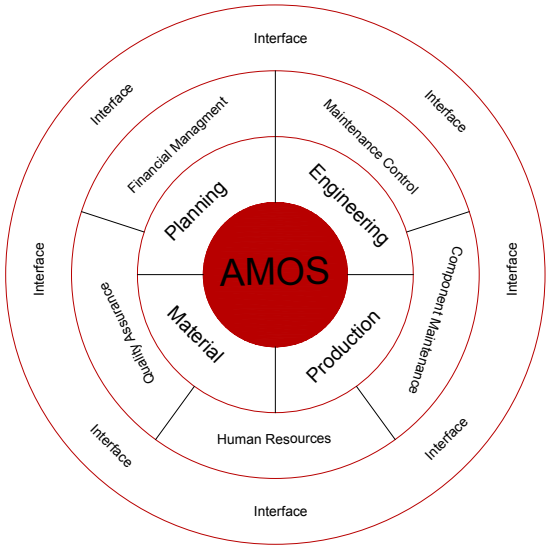


Fig. 3. AMOS architecture (adapted from [21])

availability, and modifiability [18], [20].

B. Research

Considering the effort done on the standardization of the industry, aviation MRO business models are quite practically standardized. However, it is not integrally documented and approved as industry standards. Federal Aviation Authority (FAA), other civil aviation authorities and regulatory bodies have set specific requirements for the how these business models would look like in their final form, but they did not specify a method(s) to reach to this final functional model.

According to Anant [16], MRO as an industry can be standardized if the following are precisely defined:*a)* Business Process Model, *b)* Enterprise Data Model and,*c)* Reference Architecture: *i)* Business, *ii)* Application, and *iii)* Technology.

An executive roundtable about MRO IT took place in 2010 [22] where representatives from both software development and aviation industries gathered to "define processes and tools on effective data management and generate possible strategies and collaborations to maximize aircraft maintenance data capabilities". This effort was originally done to address the contemporary and future need for solutions to address the ability of the new generation of aircrafts to provide a huge amount of data through built-in monitoring systems that are much sophisticated than the older generations. The attendees agreed that this new technology must be addressed by advanced ground-based technology that takes advantage of this new fleet generation and levels up the maintenance process through data availability and integration. The main points that were found as a challenge to the aviation MRO IT, and concerns its software architectures, were problems related to *i)* integration and migration of data, *ii)* lake of standards and formal description for the system, *iii)* security, flexibility, and availability of the system, *iv)* automation of the system.

Considering the effort required to propose functioning models, a relatively simple example of a cyber-physical system architecture is proposed by Lampe *et al.* [23]. The study presented the implementation of ubiquitous computation in the MRO business. Ubiquitous systems are an embedding computational capability (generally in the form of microprocessors) into everyday objects to make them communicate effectively and perform useful tasks in a way that minimizes the end user's need to interact with computers [24]. Such a system intended to maximize performance, and put costs to minimum. The paper realistically identifies the limitation of the study as mentioned that the focus of the study: "needs to generalize the presented architecture to be applicable in different asset management scenarios, which would include

the specifications for the base functions and services that are essential for optimization of asset management".

In [25], authors provide a review and benefits of product lifecycle management (PLM) in aviation MRO. They state that PLM is extensively used in the design phase of aircrafts, however, it is used nearly ten times less frequently in maintenance, repair and overhaul. This study shows that a software solution architecture dedicated to MRO should consider PLM to address maintenance as a product (service), as the aircrafts have long life spans of some thirty years.

Zhu *et al* proposed a web-based product service system (PSS) for aerospace MRO services [26]. The PSS was designed to better integrate product development with MRO operations. The integrated knowledge base into the model to reuse the previous experiences and reduce the ambiguity in MRO services which can act as a decision support tool.

Finally, artificial intelligence solutions and other solutions can be integrated into MRO systems' architecture as modules to perform some task like predicting maintenance faults before they occur during flight or even before the aircraft takes off. An example for using such methods in aviation maintenance is introduced in another study where Long Short Term Memory Recurrent Neural Networks were investigated to predict aviation turbine engines excessive vibrations before they actually occur, giving the pilots a chance to take remedy actions before safety wise unfavorable events happen [27].

V. Client-Cloud MRO Architecture

The reviewed work in the literature address the functional requirements of an MRO business system, but still, there are some issues and concerns that can be considered to enhance the experience and the outcome of using such system. Security is in the heart of this concerns as confidentiality plays a big role in the aviation business. Also, availability of the system is a very vital aspect of the industry. Moreover, current methods and technology as machine learning, artificial intelligence and, IoT, can be exploited to optimize the processes of the industry through predictions and forecasts using the huge data flow and data collection from either the aircraft or the records of the maintenance or operation processes. All of these are software technical aspects of the MRO software application. By achieving those aspects, the safety of industry will be boosted as the work environment shall be more optimized, and less stressful. Besides, business economics will show better performance as a system will be more efficient performance and cost wise.

A Client-Cloud MRO architecture is proposed in this paper as a reference architecture. The two main components of the proposed architecture are the Client component and the Cloud component. The main purpose of the system is to ease the main workload of an MRO typical organization. This workload consists of engineering designs and analysis for technical problems, plans for the technical work on aircraft, logistic preparations, maintenance actions, and management review and analysis. In addition to this, the system would take advantage of uprising technology like IoT and machine learning to offer a new service to the MRO industry through collecting data online and analyzing this massive amount of data automatically and computationally instead of manually. Furthermore, the system considers vital aspects of the industry as:

- **Security:** Data access should be limited in quantity and quality, meaning; data should not be accessible except for people who are associated with it and not for other and also for the purpose of the tasks assigned and not beyond that. To achieve this:
  - the study designed the data to be at the heart of the system's architecture and limit access to it through a data controller which takes care of directing data to other components by honoring requests from those components just as the work requires,
  - sensitive data related to each MRO organization should stay within the institution to ensure that it is concealed. On the other hand, data about certain make/type/model of aircraft can be collected as white data to be sent to a cloud where it can be used for performance analysis.
  - the data sent to the outside world also goes through the controller and is heavily filtered to just be white data that will not allow any inferences about the company, equipment and/or, personnel.
  - data flowing from the aircraft sensors should flow in a one-way direction to the data repository of the system.
- **Systems' performance analysis**[1]**:** Most MRO organizations might serve several aircrafts of certain make/type-/model at most. To make sound performance analysis it needs data and information from more than a limited number of aircrafts of the same made/type/model. In order to solve this problem, data can be collected from the fleet that represents this group of aircraft from all the possible MRO and operation organizations. With such a huge amount of data, machine learning and artificial intelligence methods can be implemented to make inferences and predictions about aircraft systems' performance and reliability.
- **Availability and Performance**[2]**:** System availability can be achieved through a well-designed architecture and a smart implementation of redundancy to guarantee availability and enhance performance (speed) at the same time.

The proposed architecture is depicted in Figure 4 and its components are described below.

A. Architecture components

1) The Client component: In the client, a central local data repository is used to store all information about the required for MRO operations. The data in this repository is made accessible to the only four teams who will be involved in MRO activities: management, logistics, engineering, and planning. Any other access should be through the four teams, through an interface similar to a layered style architecture.

[1]Aircraft systems' performance analysis

[2]Software solutions performance (speed)

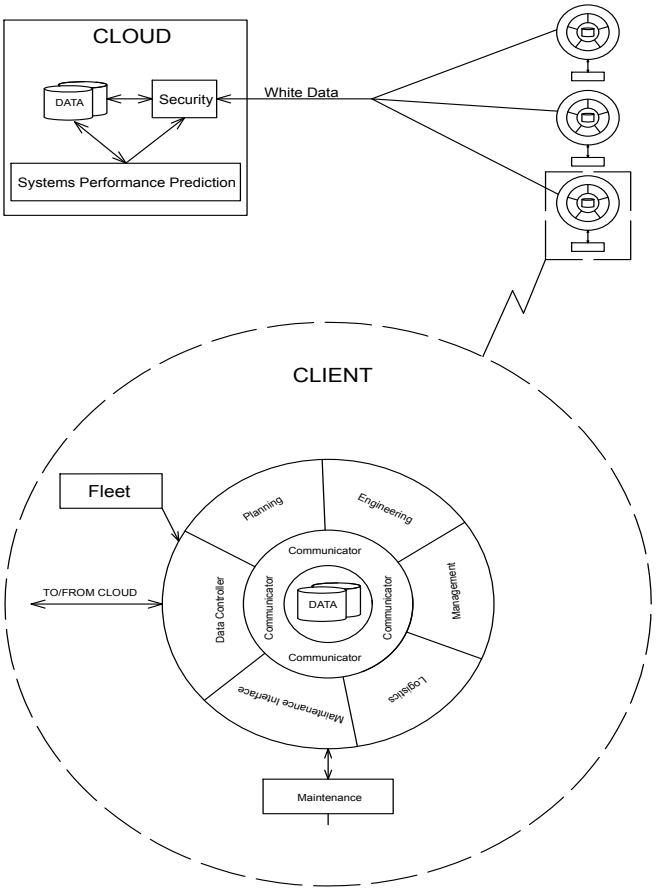


Fig. 4. Cloud-Client MRO architecture

The maintenance team is responsible for performing the actual physical MRO operations and they execute the tasks based on the work order. They are connected to the system through an interface. Thus, they will not have any direct read access to the data repository. They will, however, have limited write access after the completion of the work order, detailing the work for future references.

The subcomponent of this component are as follows:

- **The Business Divisions Component**: This component represents the core component of the MRO business model. It is further divided into sub-components:
  (a) Engineering: this component takes care of data analysis and data visualization to assist in the decision taking process. It should also have access to all technical information to design solution for the maintenance process.
  (b) Planning: this component prepares the plans for the maintenance checks and actions. It should have access to records and schedules beside of course access to data of the logistics.
  (c) Logistics: this component follows the planning orders regarding material and spare parts. It also follow-up the inventory entries and its service lives. Also, it arranges for transfer of material to the location for maintenance convenience.
  (d) Maintenance: the component serves as a portal to the maintenance sector in the MRO. Through it, the maintenance personnel would have access to work orders and checks plans. Through this component, they can also access the technical documentation so that they can perform their tasks and communicate with the logistics, planning, and/or engineering sectors. When work is done, technicians can record it through this component as well and store it in the Data component.
  (e) Management: this is the upper management and inspection portal where the managers can have their high-level information and charts and the inspectors would have reports about the quality of the work.

  This is a gross break down of the MRO business. It can still be further refined and put in more details as other sub-components, like for example Ground Equipment Services (GES), Inspection and Quality assurance, Quality control, Records management, Tools-Shop Services, ... ect. can also be added to this business structure, but this would be beyond the scope of the study.
- **The Data Controller Component**: This component orchestrates the communication between the business component, fleet component, and cloud component from one side and the Data component from the other side. It takes care of the data transfer rates, security of data transfer and data access, and concurrency of data access.
- **The Communicator Component**: The Business Division component's sub-components not only need to communicate with the Data components, but they also need to communicate with each other to pass information and instruction. For instance, the Planning sub-component needs to talk to the Logistics sub-component to know about the inventory of the material and spare parts before releasing the maintenance plan. This is done through the Communicator component.
- **The Data Component**: The component lies in the heart of the system. Access to it is very limited to the Data control component which also restricts the communication to the Business component's sub-components with data repository (each for certain specific data), the Fleet components for data dumping and cloud for dumping analysis data and sending certain previously specified data to the cloud.

2) The Cloud component: The cloud serves as a data repository of all the clients, without identifying them. The encrypted data from the controller will pass through a security module to verify the authenticity of the information. The security module will also encrypt and decrypt data. The goal of the security layer is to prevent any unauthorized access.

A central cloud repository is used to gather data about aircrafts from different clients to employ statistical and artificial intelligence methods to make system's performance analysis/predictions. The main idea is to collect enough data to make reliable predictions. Almost all the reliable data analysis methods need ample data, and it is unlikely that only one client will have such amount of data. Therefore, having a central data repository will be beneficial to all the clients. The forecasts will be communicated back to the clients through the controller. These forecasts can be used by the clients as an early warning to plan in advance for future MRO operations.

The subcomponent of this component are as follows:

TABLE I
COMPARISON OF THE DIFFERENT ARCHITECTURES

	AMOS	*MAXIMO*	*PREDIX**	*ClientCloud MRO*
Style	Layered + Client-Server	Pipe-filter + Client-Server	Client-Server (cloud)	Client-Server + Blackboard + Cyber-Physical
Availability	Medium	Medium	High	High
Maintainability	High	High	Medium	High-Medium
Reuse	Medium	High	N/A	High-Medium
Performance	Low	Medium	High	High-Medium
Usability	High	Medium	N/A	High
Modifiability	High	High	N/A	High-Medium
Scalability	Low	Medium	High	High

*The structure of the architecture is not available. Assumptions are made regarding a cloud based system requirements

- **The Data Component**: this component represents the database on the cloud side. This would receive white data about the fleet of the clients and categorize them according to the aircraft make, type, and model. The access to this component is limited to write right for the security component and a read/write right for the Systems Prediction component.
- **The Security Component**: the component is a filtering security layer for the cloud's outside world communication.
- **The Systems Prediction Component**: the component uses statistical methods along with AI and machine learning techniques to make performance analysis and prediction about a certain type of aircraft. Results are then broadcasted to the clients.

VI. ANALYSIS

A comparison between the different presented architectures is shown in Table I. The decisions made in the table are based on the known properties of the standard architectural styles reviewed in the literature of software architecture.

Since the architecture for the GE Digital's PREDIX is not available, assumptions are made about cloud-based system quality requirements. Furthermore, the available architectures are very abstracted and details are hidden in published diagrams. Thus, some assumptions are also made regarding these architectures based on the general outlook of the architecture and the architectural styles it seems to use. The architectures of AMOS and MIXMO are decided based on the available diagrams and figures in the literature. AMOS is considered a combination of a layered and client-server architectural styles. MAMIMO is considered a combination of pipe-filter and client-server architectural styles. These decisions lead to the decisions shown in Table I based on the inherited properties of those styles [18]–[20].

The designed Client-Cloud architecture exploits the blackboard style in the communication between the Business component sub-components, the layered style between the Business/Controller/Fleet components and the Data component, the client-server style between the Client and Cloud components, and the pipe filter architecture between the Cloud component sub-components.

The advantage of using the blackboard style [20], [28] is that the extendability of the Business Division component can easily be done by just adding new sub-components. Also, change and maintenance in the Client Component can easily be done under the blackboard style. However, it has a downside with the Controller sub-component being a bottle-neck in the system and hence can affect the performance and availability of the system [18], [19]. This is a very serious problem in a vital system like the MRO system but it can be mitigated by using a type of redundancy with the Controller component consisting of two or more components utilized to actively perform the same exact tasks but one of them would do the work as a checker and once the main controller is down, the shadow controller(s) take(s) over. The same analogy can be done with the communicator except that the shadow communicator can also take care of some of the communications to reduce the traffic on the main communicator. Redundancy is also required with the database while distributing the data on the data units can also enhance the performance since the data accessing components are not always accessing the same data at the same time.

The advantage of using the layer style [18], [20] is that the data access is very limited to a specific component and hence security is promoted. The disadvantages of this style are the reusability and performance, but this problem does not exist in this case since the layers are just two.

The advantages of the client-server style are the modifiability, scalability, and accessibility which are very good for such a dynamic growing and changing business. Redundancy can still be used here in the data repository on the Cloud (server) side. The accessibility problem associated with the networking infrastructure can be considered a minimal risk as the data provided by the Cloud is not of operational immediate necessity nature. The security concern can be addressed using encryption and authentication measures beside the fact that the data transferred to the cloud is white data that can not lead to any information about the entities or personnel. Maintainability, scalability, and modifiability are also counted as advantages of this type of style.

VII. ACME REPRESENTATION OF THE CLOUD-CLIENT ARCHITECTURE

The Acme representation of the proposed architecture of this study is illustrated in Figure 5. The tool used to build this representation is AcmeStudio [29].

The main components of the architecture (Cloud, Client) are shown in Figure 5a connected through RemoteSendReceiveCntT connectors (Table III) between the Controller Division component (client side , from Security_Manager component) and the Security Manager component (cloud side). The subcomponents of the Client component are shown in Figure 5b: the Fleet (representing aircraft own/served by the organization), the Business Division (representing the main sectors of the MRO organization: Engineering, Planning, Logistics, Management, and Maintenance and the Communicator components), the Repository (Data) and, the Controller com-

ponents which controls all data communication. The Business Division components are connected to the Communicator component through OneToManySendRevieveCntT and ManyToOneSendRecieveCntT connectors (Table III). Business Division component is connected from the Communicator component to the Controller Division component (from Conn_Mangaager component) through LocalSendRecieveCntT connectors (Table III). The Control Division component (from Conn_Mangaager component) is connected to the Repository component through LocalSendRecieveCntT connectors (Table III). The Fleet component is connected to the Controller Division component (from Security_Manager component) through special LocalSendRecieveCntT (Table III) connectors that allow data traffic in only one way.

The sub-components of the Business Division component are shown in Figure 5d: the Engineering, the Planning, the Logistics, the Management, the Maintenance and, the Communicator components. The sub-components of the Cloud component are shown in Figure 5c: the Repository (Data), the AI and, the Security Management components. Cloud subcomponents are connected through LocalSendRecieveCntT (Table III).

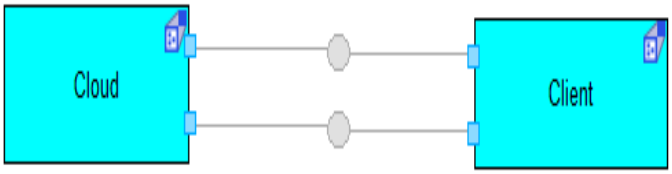
An MRO family of connectors, ports and, connectors were created for the MRO architecture to define a domain-specific vocabulary. The family extended the built-in families in AcmeStudio: *DaflowFam*, *RepositoryFam*, *CallReturnFam*, and *PubSubFam*.

A. Components Types

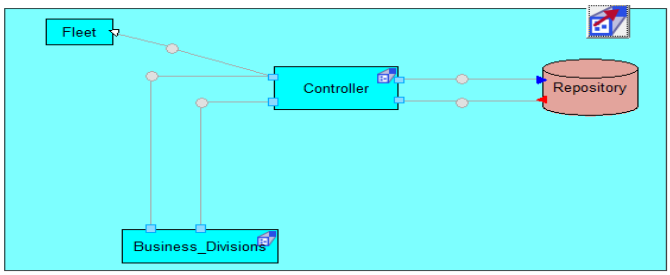
Seven MRO component types are defined in the MRO family:

(a) *BDCompT:* This component type represents different business divisions involved in MRO operations in a business organization. The business divisions involved in MRO are logistics, engineering, planning and management.
(b) *ClientCompT:* This component type is defined to represent the client side of the architecture.
(c) *MROCloudCompT:* This component defines the cloud side of the architecture.
(d) *FleetCompT:* This component type is used to represent the fleet of aircrafts.
(e) *RepositoryCompT:* This component type is used to represent the data repository. Our architecture uses two data repositories, one in the client side and one in the cloud.
(f) *CommunicatorCompT:* The business division in our architecture talk to each other and other components of the architecture using a communicator.
(g) *ControllerCompT:* The architecture uses a controller and this component type is used to represent the controller.
(h) *SecurityManagerCompT:* This component type is created to manage security of client-client connection.
(i) *ConnManagerCompT:* This component type is defined to be used for the management of the connections.
(j) *AICompT:* This component type is used to in the cloud for data analysis.

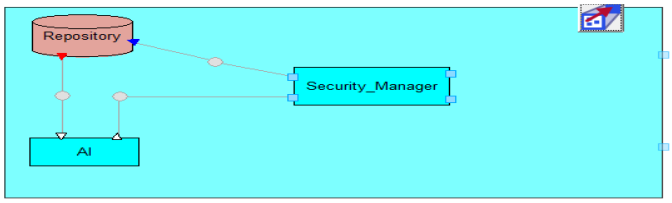
Table II shows the list of Acme declared components used in the architecture and the associated port(s) to each of the components.



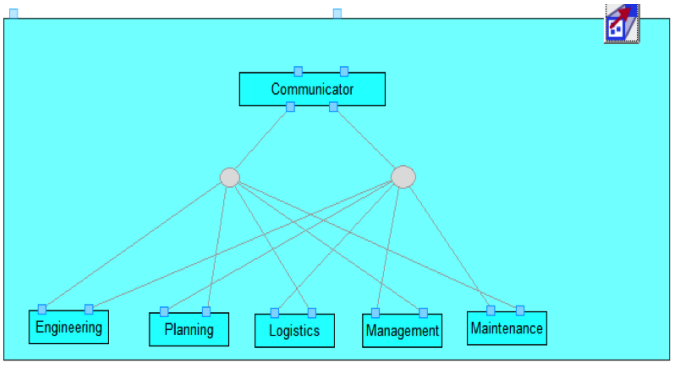
(a) Main Architecture



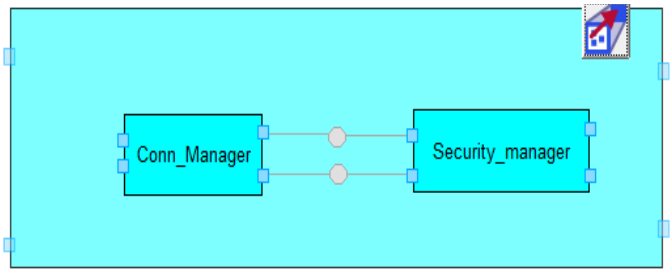
(b) Client Architecture



(c) Cloud Architecture



(d) Business Division Architecture



(e) Controller Division

Fig. 5. Cloud-Client Acme representation

TABLE II
ACME USED COMPONENTS TYPES

Component Types	Ports	Port Quantity
ClientCompT	sendData: SendPortT	1
	receiveData: SubscribePortT	1
CloudCompT	sendData: PublishPortT	1
	receiveData: ReceivePortT	1
BDCompT	sendRequest: ReuqestPortT	1
	receiveRequest: ResponsePortT	1
RepositoryCompT	dataIn : DataInputPortT	1
	dataOut: DataOutputPortT	1
ControllerCompT	send: SendPortT	2
	receive: ReceivePortT	2
CommunicatorCompT	receive: ReceivePortT	1
	send: SendPortT	1
	sendRequest: ReuqestPortT	1
	receiveRequest: ResponsePortT	1
FleetCompT	flightData: OutputPortT	1
AICompT	dataIn: InputPortT	1
	sendPrediction: OutputPortT	1
ConnManagerCompT	receive: ReceivePortT	2
	send: SendPortT	2
SecurityManagerCompT	receive: ReceivePortT	2
	send: SendPortT	2

TABLE III
ACME USED CONNECTION ROLES

N	Role
1	Sender
2	Receiver
3	Request
4	Respond
5	Publisher
6	Subscriber

B. Port Types

Along with components, different port types are defined in the family. They are:

(a) *DataInputPortT and DataOutputPortT:* These ports are designed for general data input and output.
(b) *SendPortT and ReceivePortT:* These ports are designed for sending and receiving information and instructions.
(c) *InputPortT and OutputPortT:* These ports are used to receive inputs to the components or deliver outputs of the components.
(d) *SubscribePortT and PublishPortT:* These ports are used to publish predictions by the cloud to the client subscribers.
(e) *ReuqestPortT and ResponsePortT:* These ports are used to request and respond in communicator/business-divisions communication.

Table III shows the list of the defined roles for the connections that connects the components of the architecture through the ports. Table IV defines the connections used and the roles associated with each connection.

C. Connector Types

Different connectors types are defined in the family. They are:

(a) *RemoteSendReceiveCntT and LocalSendReceiveCntT:* These connectors are designed for the local and remote sending and receiving of data and instructions.
(b) *OneToManyRequestRespondConnT and ManyToOneRequestRespondConnT:* These connectors are designed for requesting and responding to data requests and instructions between the Communicator component and the Business Division subcomponents. The roles associated with these connectors are requester and responder.
(c) *CCPublishSubscribeConnT:* This connector is used in the cloud to client communication. The cloud acts as a publisher and after making predictions, it will broadcast the predictions to all the subscriber clients. The roles associated with this connection type are publisher and subscriber.

TABLE IV
ACME USED CONNECTIONS

Connector	Roles*	
RemoteSendReceiveCntT	1	2
LocalSendReceiveCntT	1	2
ManyToOneRequestRespondConnT	3	4
OneToManyRequestRespondConnT	3	4
CCPublishSubscribeConnT	5	6

*From Table III

Table III shows the list of the defined roles for the connections that connects the components of the architecture through the ports. Table IV defines the connections used and the roles associated with each connection.

Script 1 depicts the declaration of Business Divisions component type (BDCompT). Script 2 depicts the declaration of data input and data output port types. Script 3 depicts the declaration of data input and data output connection types.

Listing 1. Declaration of Business Divisions component type (BDCompT)

```
Component Type BDCompT = {
  Port inputport : InputPortT =
                new InputPortT extended with { }
  Port outputport : OutputPortT =
                new OutputPortT extended with { }
}
```

Listing 2. Declaration of data input and data output port types

```
Port Type DataInPortT extends
        FilterInputPortT with { }
Port Type DatOutPortT extends
        FilterOutputPortT with { }
```

Listing 3. Declaration of data input and data output connection types

```
Connector Type LocalSendReceiveConnT = {
  Role sender : SenderRoleT =
                new SenderRoleT extended with { }
  Role receiver : ReceiverrRoleT =
                new ReceiverrRoleT extended with { }
}
```

D. Properties and constraints

Properties and constraints in Acme are used to enhance the system architecture. Properties are generally useful while implementing the system. Examples of some of the properties for this architecture are explained below.

The client-cloud architecture is different from a normal client-server style because the cloud component can also initiate communication with the client after making predictions from data analysis. The connector connecting the cloud and

client will use the internet as an infrastructure. Thus, the communication protocol property will be TCP/IP besides the ability to use some kind of secure connection as an intranet connection. One of the constraints at the system level could be that the cloud is able to broadcast its analysis of the raw data results to all the clients in the system. All the cloud-client communication should be encrypted for security.

The connection from the fleet to data controller is a strictly one-way connection. The controller should not be allowed to send any data/instructions to the fleet. The controller should limit access to data, grabbing data to users who hold the right privileges. The communicator should not be able to initiate communication on its own and should only match specific requests to specific responses. Furthermore, the business divisions component should not be allowed to communicate with more than one other business divisions component at a time.

VIII. Conclusion and Future Work

The study proposed an architecture as a structure of an MRO software application. The proposed work represented here is trying to take advantage of the key properties related to the lend of architectural styles, such as cloud-client, layer, and modified architectural styles to enhance security, availability, performance, scalability, modifiability, and maintainability beside of course being able to address the functional requirements of the industry. A higher level illustration of the architecture design is introduced using figures and Acme as an ADL.

The work has an abstract nature as it does not address an exhaustive list of the technical properties of the components, ports and, connections of the system in its Acme model. This is left for further considerations and study as this should take more industrial and operational details into consideration to tailor the system to suit the practitioners of the industry. Such an effort needs a work effort at the level of a steering group or something similar to count all the opinions and expertise of the stack holders. The study also addressed the basic futures of the MRO business model as engineering, planning, logistics, management beside of course the basic one, maintenance. This can be expanded to add more components like quality assurance, quality control, safety, record control, ground equipments, ... etc.

A future work would consider an implementation of the system and a closer study of the safety, performance and security risks.

References

[1] Donahue, Jeffrey and Anne Hendricks, Lisa and Guadarrama, Sergio and Rohrbach, Marcus and Venugopalan, Subhashini and Saenko, Kate and Darrell, Trevor, "Long-term recurrent convolutional networks for visual recognition and description," June 2015.

[2] Linlin Chao et al, "Audio visual emotion recognition with temporal alignment and perception attention," Mar 2016.

[3] D. Eck and J. Schmidhuber, "A first look at music composition using lstm recurrent neural networks," *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, vol. 103, 2002.

[4] D. R. Vieira and P. L. Loures, "Maintenance, repair and overhaul (mro) fundamentals and strategies: An aeronautical industry overview."

[5] H. A. Kinnison and T. Siddiqui, *Aviation maintenance management*, 2012.

[6] D. Garlan, "Formal modeling and analysis of software architecture: Components, connectors, and events," *Formal Methods for Software Architectures*, pp. 1–24, 2003.

[7] S. Björnander, "Architecture description languages," Ph.D. dissertation, PhD thesis. Mälardalen University. URL: http://www. mrtc. mdh. se/˜ han/FoPlan/ass2-bjornander. pdf.

[8] P. H. Feiler, D. P. Gluch, and J. J. Hudak, "The architecture analysis & design language (aadl): An introduction," DTIC Document, Tech. Rep., 2006.

[9] D. Garlan, R. Monroe, and D. Wile, "Acme: An architecture description interchange language," in *CASCON First Decade High Impact Papers*. IBM Corp., 2010, pp. 159–173.

[10] D. C. Luckham, J. J. Kenney, L. M. Augustin, J. Vera, D. Bryan, and W. Mann, "Specification and analysis of system architecture using rapide," *IEEE Transactions on Software Engineering*, vol. 21, no. 4, pp. 336–354, 1995.

[11] R. Allen and D. Garlan, "A formal basis for architectural connection," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 6, no. 3, pp. 213–249, 1997.

[12] J. Magee, N. Dulay, S. Eisenbach, and J. Kramer, "Specifying distributed software architectures," *Software Engineering—ESEC'95*, pp. 137–153, 1995.

[13] D. Garlan, R. Allen, and J. Ockerbloom, "Exploiting style in architectural design environments," *ACM SIGSOFT Software Engineering Notes*, vol. 19, no. 5, pp. 175–188, 1994.

[14] L. COGLIANESE and R. Szymanski, "Dssa-adage: an environment for architecture-based avionics development," *AGARD, Aerospace Software Engineering for Advanced Systems Architectures 8 p(SEE N 94-29315 08-61)*, 1993.

[15] A. Kompanek, "Modeling a system with acme," "http: //www.cs.cmu.edu/~acme/html/WORKING-%20Modeling%20a% 20System%20with%20Acme.html".

[16] A. Sahay, *Leveraging information technology for optimal aircraft maintenance, repair and overhaul (MRO)*. Elsevier, 2012.

[17] L. Winnig, "Ge's big bet on data and analytics," *MIT Sloan Management Review*, vol. 57, no. 3, 2016.

[18] B. FERRO CASTRO, "Pattern-oriented software architecture: A system of patterns," *Computación y Sistemas*, vol. 1, no. 002, 1969.

[19] D. D. Corkill, "Blackboard systems," *AI expert*, vol. 6, no. 9, pp. 40–47, 1991.

[20] M. Shaw and D. Garlan, *Software architecture: perspectives on an emerging discipline*. Prentice Hall Englewood Cliffs, 1996, vol. 1.

[21] ...., "Swiss aviationsoftware," "http://www.swiss-as.com/modules.do".

[22] "Aviation week executive roundtable: Mro it: Extracting mro intelligence from a data intensive aircraft," *Aviation Week*, "http://mromarketing.aviationweek.com/ExecutiveRoundtable/ downloads/MRO%20IT%20roundtable%20-%20Whitepaper.pdf".

[23] M. Lampe, M. Strassner, and E. Fleisch, "A ubiquitous computing environment for aircraft maintenance," in *Proceedings of the 2004 ACM symposium on Applied computing*. ACM, 2004, pp. 1586–1592.

[24] "Techtarget: Iot agenda," "http://internetofthingsagenda.techtarget.com/ definition/pervasive-computing-ubiquitous-computing".

[25] S. Lee, Y.-S. Ma, G. Thimm, and J. Verstraeten, "Product lifecycle management in aviation maintenance, repair and overhaul," *Computers in industry*, vol. 59, no. 2, pp. 296–303, 2008.

[26] H. Zhu, J. Gao, D. Li, and D. Tang, "A web-based product service system for aerospace maintenance, repair and overhaul services," *Computers in Industry*, vol. 63, no. 4, pp. 338–348, 2012.

[27] A. ElSaid, B. Wild, J. Higgins, and T. Desell, "Using lstm recurrent neural networks to predict excess vibration events in aircraft engines," in *e-Science (e-Science), 2016 IEEE 12th International Conference on*. IEEE, 2016, pp. 260–269.

[28] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy, "The hearsay-ii speech-understanding system: Integrating knowledge to resolve uncertainty," *ACM Computing Surveys (CSUR)*, vol. 12, no. 2, pp. 213–253, 1980.

[29] A. Team, "Acme, carnegie mellon university," "http://www.cs.cmu.edu/ ~./acme/AcmeStudio/index.html".