

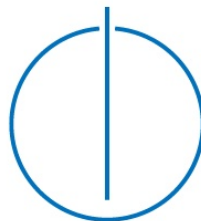
**Technische Universität
München**

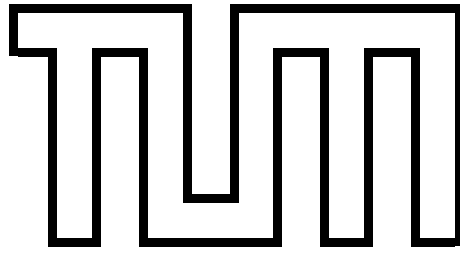
Fakultät für Informatik

Master's Thesis in Informatik

Development of a Web Based Application for the Implementation
of an Integrated System Health Management (ISHM)
Certification Process

MD Mushfiquir Rahman





Technische Universität
München

Fakultät für Informatik

Master's Thesis in Informatik

Development of a Web Based Application for the Implementation
of an Integrated System Health Management (ISHM)
Certification Process

Entwicklung einer webbasierten Anwendung für die Abwicklung
eines ISHM-Zertifizierungsprozesses

Author: MD Mushfiquir Rahman

Supervisor: Prof. Dr. Florian Matthes

1st Advisor: Adrian Hernandez-Mendez, M.Sc.

2nd Advisor: Milton Amador, Dipl.-Ing.

Submission: 15.01.2016

I assure the single handed composition of this master's thesis only supported by declared resources.

München, 15.01.2016

(MD Mushfigur Rahman)

Acknowledgment

First and foremost, I would like to thank Prof. Dr. Florian Matthes for giving me the opportunity to write my Master thesis in the Software Engineering for Business Information Systems (SEBIS) chair at TU Munich.

I earnestly thank my two advisors, Milton Amador (Airbus Group Innovations) and Adrian Hernandez-Mendez for their continuous motivation, extensive advice and endless support throughout this research work. In addition, I would like to thank Airbus Group Innovations TX5 team members. I enjoyed working in such a friendly place with nice colleagues.

Last but not least, I thank my parents for always supporting me and pushing me towards my dream.

Abstract

The goal of this master thesis is to develop a Web-Based Application for the Implementation of an Integrated System Health Management (ISHM) Certification Process. Developed Mx Credit Framework (MCF) Tool can be used for aircraft health monitoring and prediction capabilities to support diagnosis and prognosis of aircraft systems and thereby to optimize the maintenance process. One of the challenges to developing such systems are a complex process environment, different stakeholders having diverse responsibilities and maintaining process flow navigation with varying contents.

Design science methodology is used to overcome these challenges. Additionally, a state of the art study was conducted to find out the right solution approach. We have made our own approach to design and implement the MCF Tool. As a part of the solution, we have used a model-based user interface to cope with the changeable process contents, and a process modeling concept is introduced to generate the process flow navigation properly. This newly developed application provides functionality to retrieve information, to add or link new content to a given process step, to trace and monitor the process and to report the status at any time. Furthermore, we have performed an evaluation of the tool by demonstration and validation of defined use cases. Last but not least, we have also presented the new functionality that can be added to MCF Tool as a future extension.

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 13 |
| 1.1 | Motivation | 13 |
| 1.2 | Problem Description | 14 |
| 1.3 | Research Questions | 15 |
| 1.4 | Research Method | 15 |
| 2 | Environment | 19 |
| 2.1 | Status Quo of Aircraft Maintenance | 19 |
| 2.1.1 | Task-oriented Maintenance Program | 20 |
| 2.1.2 | Maintenance Plan | 22 |
| 2.1.3 | Aircraft Maintenance Regulations | 23 |
| 2.1.4 | Challenges of Aircraft Maintenance | 24 |
| 2.2 | Requirements Overview | 26 |
| 2.2.1 | Applicability | 26 |
| 2.2.2 | Stakeholder and Technical Requirements | 27 |
| 3 | Knowledge Base | 31 |
| 3.1 | Existing Process Modelling and Execution Tools | 31 |
| 3.1.1 | Activiti | 32 |
| 3.1.2 | jBPM | 36 |
| 3.1.3 | Bonita BPM | 37 |
| 3.1.4 | Model Based UI Approach | 40 |
| 3.2 | Comparison of identified Technologies and Tools | 43 |
| 3.3 | Existing Technologies | 45 |
| 3.3.1 | AngularJS | 45 |
| 3.3.2 | Node.js | 45 |
| 3.3.3 | Express.js | 46 |
| 3.3.4 | JSON | 46 |
| 3.3.5 | Neo4j | 47 |
| 3.3.6 | REST | 48 |

| | | |
|----------|---|------------|
| 4 | Conceptual Design for Mx Credit Framework | 50 |
| 4.1 | Common Approach | 50 |
| 4.2 | Description of Mx Credit Framework | 51 |
| 4.3 | Description of Mx Credit Process | 52 |
| 4.4 | Involved Stakeholders | 55 |
| 4.5 | Technical Concept for Mx Credit Application | 56 |
| 4.5.1 | Proposed Architecture | 56 |
| 4.5.2 | Management of Process Flow | 57 |
| 4.5.3 | Approach for Authoring and Publishing of Mx Credit Process content | 58 |
| 5 | Design and Implementation | 62 |
| 5.1 | Mockups for Mx Credit Framework | 64 |
| 5.2 | Mx Credit Application Architecture | 68 |
| 5.2.1 | Graph Data model for MCF Tool | 69 |
| 5.2.2 | RESTful API | 70 |
| 5.3 | Implementation of Process Modeller | 73 |
| 5.3.1 | Single Page Application Implementation | 74 |
| 5.4 | Model Based UI | 76 |
| 5.4.1 | Process Schema Generation | 77 |
| 5.4.2 | User Interface Generation from JSON Schema | 80 |
| 5.5 | Process Flow Engine | 81 |
| 6 | Evaluation of the MCF Tool | 83 |
| 6.1 | Definition of a representative Use Case | 84 |
| 6.2 | Short EHA Description | 85 |
| 6.3 | Considered Failure Modes and Maintenance Concepts for EHA | 86 |
| 6.3.1 | Visual Inspection | 86 |
| 6.3.2 | Leakage Check | 86 |
| 6.4 | User Acceptance Tests | 86 |
| 6.5 | Evaluation results | 87 |
| 6.5.1 | Component View | 87 |
| 6.5.2 | Manage Group & User View | 88 |
| 6.5.3 | My Tasks View | 89 |
| 6.5.4 | Process Step Detail View | 90 |
| 6.5.5 | Process Execution View | 91 |
| 6.5.6 | Tabular Process View | 93 |
| 6.5.7 | Model Based UI Generation | 94 |
| 6.5.8 | Process Steps View | 95 |
| 6.6 | Analysis of MCF Tool Evaluation | 100 |
| 7 | Conclusion and Future Work | 102 |

| | |
|--------------------------------|------------|
| <i>CONTENTS</i> | 8 |
| Appendices | 107 |
| A List of Abbreviations | 108 |
| B EHA Use Case Details | 109 |

List of Figures

| | | |
|-----|--|----|
| 1.1 | Information Systems Research Framework [16] | 16 |
| 1.2 | Resoning of Design Cycle | 17 |
| 1.3 | Design Research in this Master Thesis. | 18 |
| 2.1 | Maintenance Strategies [39] | 20 |
| 2.2 | Maintenance plan [39] | 22 |
| 2.3 | Aircraft maintenance regulations [27] | 24 |
| 3.1 | Components of Activiti [8] | 32 |
| 3.2 | Activiti explorer [9]. | 33 |
| 3.3 | BPM Eclipse Editor for process | 36 |
| 3.4 | Bonita BPM Studio - Graphical BPMN 2.0 Workflow Designer [13] | 38 |
| 3.5 | Bonita BPM Portal [12] | 39 |
| 3.6 | The Cameleon Reference Framework [2]. | 41 |
| 3.7 | Neo4js is an ACID-compliant database | 48 |
| 4.1 | Process Environmet of MCF. | 51 |
| 4.2 | General Mx Credit Process [39]. | 54 |
| 4.3 | User group association with process steps [39]. | 55 |
| 4.4 | Process Flow with Model Based UI | 57 |
| 4.5 | Process Flow with Model Based UI. | 58 |
| 4.6 | Relationship of the Components, Sub Components and Items of the MCF Tool [39]. | 59 |
| 4.7 | Relationship of contents to Item level [39]. | 60 |
| 4.8 | Relationship of contents to Item level [39]. | 61 |
| 5.1 | Design and Implementation approach of MCF Tool | 63 |
| 5.2 | Proposed architecture for Mx Credit Application | 64 |
| 5.3 | Dashboard Mock-up for Mx Credit Application | 65 |
| 5.4 | Process Steps Mock-up for Mx Credit Application | 66 |
| 5.5 | Process Steps Mock-up for Mx Credit Application | 67 |
| 5.6 | Process view Mock-up for Mx Credit Application | 68 |

| | | |
|------|---|----|
| 5.7 | Deployment diagram of Mx Credit Application | 69 |
| 5.8 | graph For Mx Credit Application | 70 |
| 5.9 | MCF Tool REST API services | 71 |
| 5.10 | Example of neo4j-js connection to Neo4j database | 72 |
| 5.12 | Class diagram of Mx Credit Application | 73 |
| 5.13 | Single page architecture of MCF Tool | 74 |
| 5.14 | The Model View Controller Software Pattern as described by Syromiatnikov and Weyns [15]. | 75 |
| 5.15 | Model based Approach for UI development based on JSON [5] | 77 |
| 5.16 | Including JSON schema for Component Details process step | 79 |
| 5.17 | Including JSON schema for Component Details process step | 81 |
| 5.18 | Communication diagram for Process Flow Engine | 81 |
| 5.11 | REST API Request/Response Data Flow | 82 |
| 6.1 | MCF web application user interface. | 84 |
| 6.2 | Mx Credit Process Description | 85 |
| 6.3 | Component View with list of components with details | 87 |
| 6.4 | Groups View | 88 |
| 6.5 | Users View | 89 |
| 6.6 | All Tasks | 89 |
| 6.7 | Assigned Task to User | 90 |
| 6.8 | Upload and View Links to process step | 91 |
| 6.9 | Upload and View Documents to process step | 91 |
| 6.10 | Process Step <i>Component Details</i> | 92 |
| 6.11 | Process Step <i>System FMEA</i> | 92 |
| 6.12 | Process Step <i>Component Details</i> | 93 |
| 6.13 | Process Step <i>Component Details</i> | 93 |
| 6.14 | Process Step <i>System FMEA</i> | 94 |
| 6.15 | Process Step <i>Manage UI with JSON Schema</i> | 94 |
| 6.16 | Process Step <i>Componet Details process step with generated UI from JSON Schema</i> | 95 |
| 6.17 | Process Step <i>Componet Details Process Step evaluation using Use case tableB.1</i> | 96 |
| 6.18 | Process Step <i>System FMEA Process Step evaluation using Use case tableB.2</i> | 97 |
| 6.19 | Process Step <i>System FMECA Process Step evaluation using Use case tableB.3</i> | 98 |
| 6.20 | Process Step <i>System FMECA Process Step evaluation using Use case tableB.3</i> | 98 |

| | | |
|------|---|-----|
| 6.21 | Process Step <i>System FMECA</i> Process Step evaluation using Use case tableB.4 | 99 |
| 6.22 | Process Step <i>System FMECA</i> Process Step evaluation using Use case tableB.4 | 100 |

List of Tables

| | | |
|-----|---|-----|
| 2.1 | Technical requirements list. | 30 |
| 3.1 | Pros and cons of Activiti | 34 |
| 3.2 | Important features of Activiti | 35 |
| 3.3 | Important features of jBPM | 37 |
| 3.4 | Pros and cons of Bonita BPM | 39 |
| 3.5 | Important features of Bonita BPM | 40 |
| 3.6 | Pros and cons of Model-Based User Interfaces Approach | 42 |
| 3.7 | Comparison of Existing Tools based on the requirement list 2.1. | 44 |
| 5.1 | HTTP Verbs available in MCF Tool API | 71 |
| A.1 | Abbreviations list. | 108 |
| B.1 | Use Case Data for System Component Selection process step | 109 |
| B.2 | Use Case Data for System Component Selection process step | 110 |
| B.3 | Use Case Data for System Component Selection process step | 111 |
| B.4 | Use Case Data for RCM Maintenance Concept process step | 112 |

Chapter 1

Introduction

In 2010 a survey was performed to identify future aviation safety risks. They include, among others, in-flight loss of control. Loss of control during flight may occur as a result of a stall, an icing-related event, a severe atmospheric turbulence or a malfunction, or failure of a flight-critical system or equipment [18]. Analysis of National Transportation Safety Board (NTSB) accident data and Federal Aviation Administration (FAA) incident data has established that system/equipment failures and malfunctions are significant contributing factors to aviation safety risk. In addition, the National Aeronautics Research and Development Plan [21] cited several fundamental safety challenges that are relevant to preventing loss of aircraft control accidents caused by system or subsystem malfunctions or failures.

The factors contributing to safety of air travel are of two type, namely, the human factor and the machine factor i.e. the aircraft [22]. It is one of the main objective of the manufacturer to see that when an aircraft is delivered to a customer, it is airworthy, meaning that it meets the requirements and conforms to type certificate and is in a safe condition for operation. One of the key factors to provide continuous airworthiness to aircraft being operated efficiently, is the application of appropriate maintenance measures.

1.1 Motivation

Maintenance of a large fleet of aircraft poses significant challenges for a business in terms of achieving the multiple, and in some ways conflicting, goals relating to maintenance and operation costs and desired service levels, including safety [23]. Given the increasing technical sophistication of the asset and the complexity introduced by different types of aircraft that make up a typical fleet, planning and scheduling of aircraft maintenance can be demanding. The impact of capital equipment downtime, regulatory compliance and the

value of spare-parts inventory needed further underline maintenance costs. Historically, there have been considerable efforts directed towards developing approaches that help minimize the downtime by more effective planning and control of maintenance operations, as well as predicting spare-parts usage and other resources requirements using forward planning [24].

Industries in the aviation sector are developing health management systems towards the enhancement of vehicle availability, mission reliability and safety, longer system life, and reduced ownership costs. There are in present ongoing efforts to integrate widely such systems into new aircraft for the health monitoring of components and the prediction of their remaining useful life.

In the past, health management systems have usually been considered as a subsequent measure, when it became possible that systems' safety was not good as expected. It would be beneficial, and probably cost effective, to consider health management systems during the design process to optimally divide the authority of safety-enhancing health management systems methods and safety-enhancing design modifications at the conceptual design stage. While some manufacturers are starting to accept this idea, an industry-wide adoption is still in its beginning [28].

One of the main challenges of introducing aircraft health management systems for new developed aircraft is how to deal with the achievement of airworthiness approval for their installation, for the validation of Maintenance Credits (Mx Credits) and for the full compliance with international safety standards, including continuing airworthiness and operations. Therefore an appropriate framework needs to be implemented to trace and monitor all relevant process steps and related inputs and outputs for the certification of aircraft health management systems and correspondent Mx Credits during aircraft development.

1.2 Problem Description

The aim of the project at Airbus Group Innovations (AGI ¹) is the development of a framework, that supports the introduction of health management applications, named Integrated System Health Management (ISHM) applications, for new developed aircraft. This so called Mx Credit Framework (MCF) shall provide all necessary inputs, information and documentation along the development process to the respective stakeholders, specially to the certification authorities, for the certification of the ISHM applications and correspondent Mx Credits ².

¹Airbus Group Innovations is the corporate research and technology Centre of Airbus Group.

²A Maintenance Credit (Mx Credit) is characterized by an approval to an Integrated System Health Management application (ISHM), that adds to, replaces, or intervenes in industry accepted maintenance

The MCF shall cover the aircraft development process, the maintenance concept development and the development and certification of ISHM applications with correspondent Mx Credits. Furthermore the MCF shall also provide a cost-benefit-analysis for the introduction of ISHM applications from the point of view of the manufacturer and towards the reduction of ownership costs.

Based on the defined MCF this work will provide a concept of an appropriate tool to support the stakeholders involved in the design, integration and certification of ISHM applications and correspondent Mx credits.

1.3 Research Questions

The overall research aim of this thesis is to implement a Maintenance Credit Framework(MCF) web application. In order to obtain this objective, important research questions were defined. That are listed below.

1. What are the state-of-the-art tools to model and execute process for modeling Mx Credit process ?
2. What are the requirements for modeling the Mx Credit process ?
3. How the Mx Credit Framework looks like ?
4. What are the specifications to evaluate proposed Mx Credit Framework adaptable with the requirements ?

These research questions figure out two main elements. First element is to find the procedure to implement rule based process flow navigation. Second, associating contents with it. These two elements create the initiation for this thesis and these will be discussed in detail with the development of the MCF Tool.

1.4 Research Method

To achieve the creation of MCF Tool, design science research methodology is used. This methodology is used for conducting design science research in information systems [16]. Figure 1.1 presents our conceptual framework for understanding, executing, and evaluating Information Systems(IS) research combining behavioral-science and design-science paradigms.

practices or flight operations. Introduced Maintenance Credits shall provide health monitoring and prediction capabilities to support diagnosis and prognosis of aircraft systems/components.

The environment defines the problem space in which the phenomena of interest resides[17]. IS research is composed of people, organizations, and their existing or planned technologies. Goals, tasks, problems and opportunities that define business needs reside in IS research scope. The knowledge base provides the raw materials through IS research. The knowledge base is composed of foundations and methodologies.

Above-mentioned IS research and results from reference disciplines provide foundational theories, frameworks, instruments, constructs, models, methods, and instantiations used in the development phase of a research study. Methodologies present the guidelines used in the justify/evaluate phase. In behavioral science, methodologies are typically rooted in data collection and empirical analysis techniques [16].

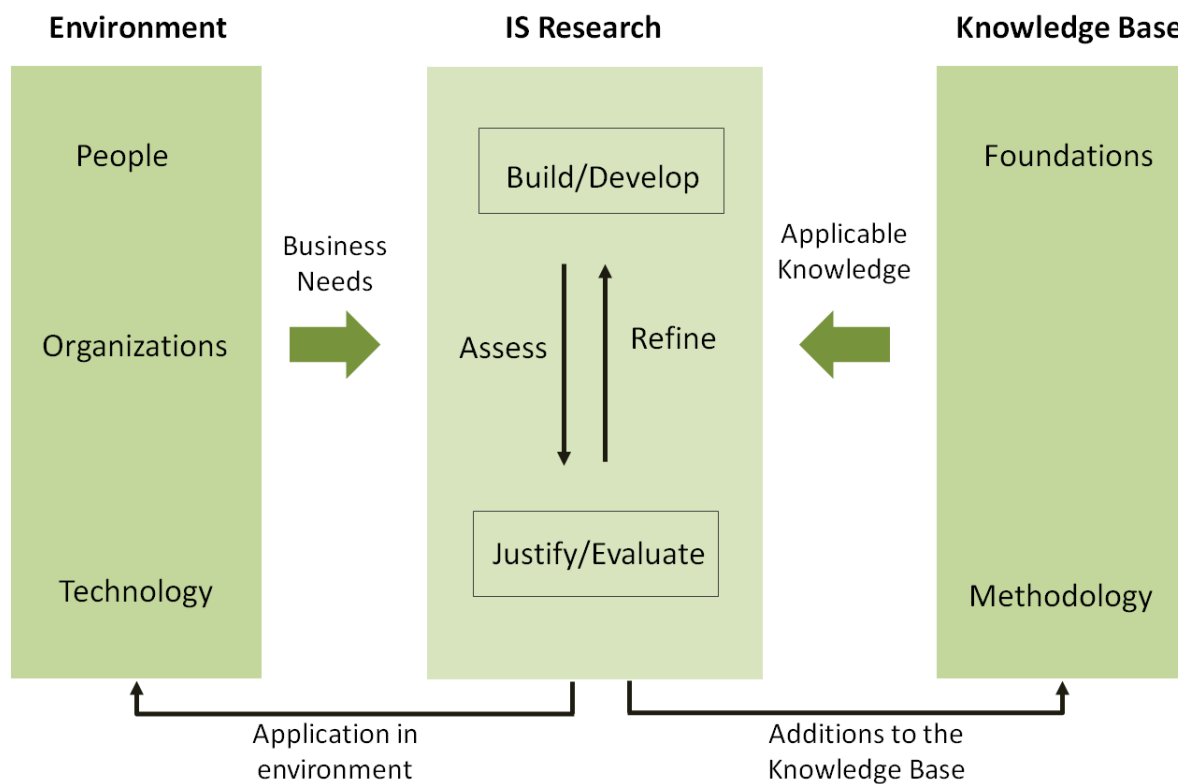


Figure 1.1: Information Systems Research Framework [16]

The work presented here is structured following the main ideas of the Design Research methodology. As stated in [1], Design Research "involves the analysis of the use and performance of designed artifacts to understand, explain, and very frequently improve on" those artifacts.

Fig 1.2 illustrates the course of a general design cycle as Takeda et al. suggested in [3]. It begins with the *Awareness of a problem*. Using the existing knowledge, Suggestions are drawn. Then, an artifact that implements the proposed solution is built. After the

implementation, the solution is evaluated. *Suggestions*, *Development* and *Evaluation* are performed iteratively.

In the *Conclusion* step results are shown and also further scope of improvement is described here.

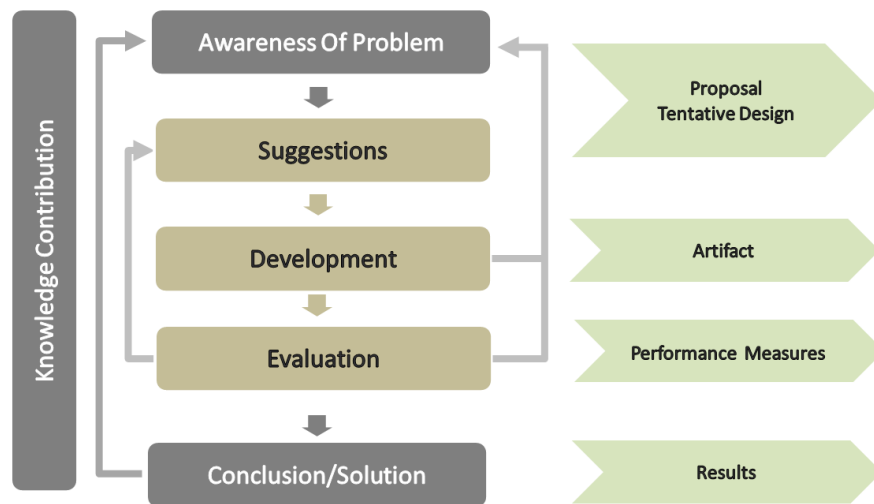


Figure 1.2: Resoning of Design Cycle

Fig 1.3 shows graphically how design research was applied to this master thesis. The first stage is Design as an Artifact. In our outline we found that there is shortage of process modeling application available that meets the current requirements of the MCF. To follow and find out more from the context of problem description we made the concrete find out of problem statement. In chapter 2 we described the context of the problem and concrete requirements.

Problem Relevance step is the defined step for making the research questions. From the problem definition, we found out the main problem relevance areas are, Navigation of process steps, finding out the important Task and Domain and improvement of UX while executing the MCF web application.

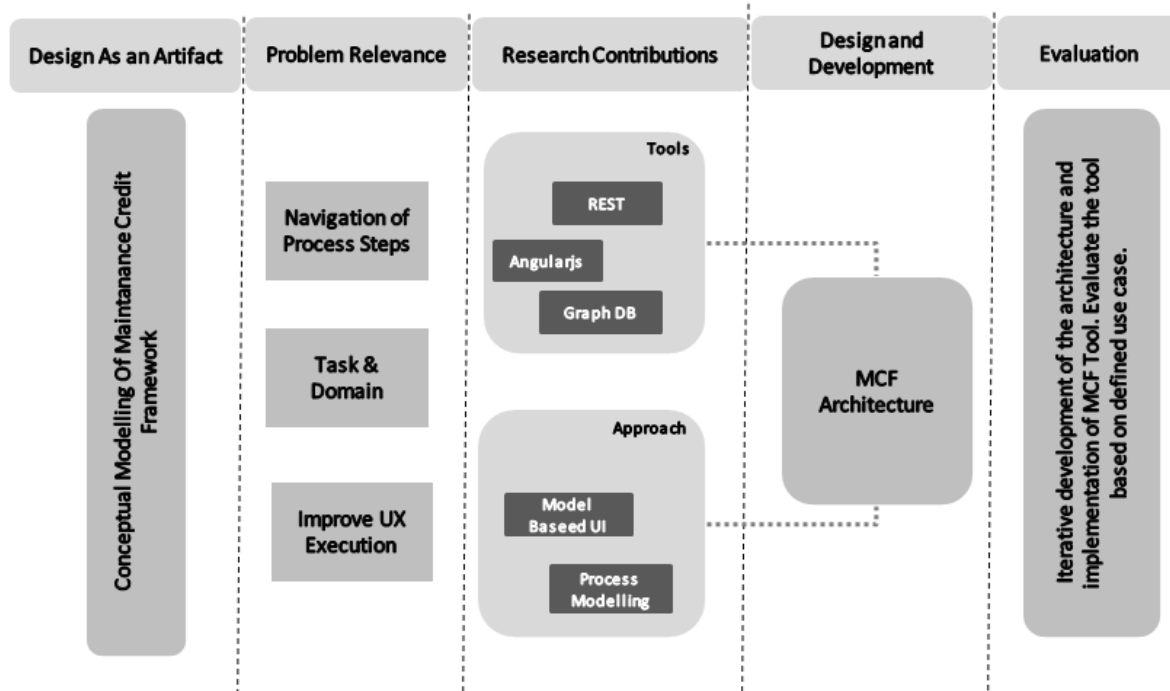


Figure 1.3: Design Research in this Master Thesis.

After finding out the problems we started to work on a solution approach. In *Research Contributions* step we explored the existing tools that are available for the solution and the identification of the approach to solve the problem. In the Design and Development phase we created the MCF Architecture. In chapter 5 we described detail of design of the MCF tool with implementation details. In chapter 6, we have evaluated the tool with a defined use case.

Chapter 2

Environment

Based on the Design Research Framework that was specified in Chapter 1, this chapter carries out analysis and definition of business needs of organizations that are impacted by the research questions. Henver et al.[33] stipulate that: “In it are the goals, tasks, problems, and opportunities that define business needs as they are perceived by people within the organization.” Thus, in analyzing this basis, we can settle the environment. We briefly describe the current status of the aircraft maintenance, maintenance regulations and maintenance plan. Lastly in section 2.2, we figure out the requirements for MCF Tool.

2.1 Status Quo of Aircraft Maintenance

Aircraft maintenance is characterized by the overhaul, repair, inspection or modification of an aircraft or aircraft component. The industry definition of maintenance generally includes those tasks required to restore or maintain aircraft’s systems, components, and structures in an airworthy condition. Maintenance is required for three main reasons [25]:

- **Operational:** To keep the aircraft in a serviceable and reliable condition so as to generate revenue.
- **Value Retention:** To maintain the current and future value of the aircraft by minimizing the physical deterioration of the aircraft in-service life.
- **Regulatory Requirements:** The condition and the maintenance of aircraft are regulated by the aviation authorities of the jurisdiction in which the aircraft is registered. Such requirements establish standards for repair, periodic overhauls, and alteration. This requires that the owner or operator establish an airworthiness maintenance and inspection program to be carried out by certified individuals qualified to issue an airworthiness certificate.

Maintenance (Mx) is divided into two major categories: scheduled and unscheduled. Figure 2.1, illustrates how these two categories are further broken down into specific tasks, Preventive Mx, On Condition Mx, Corrective Mx and Deferred Corrective Mx.

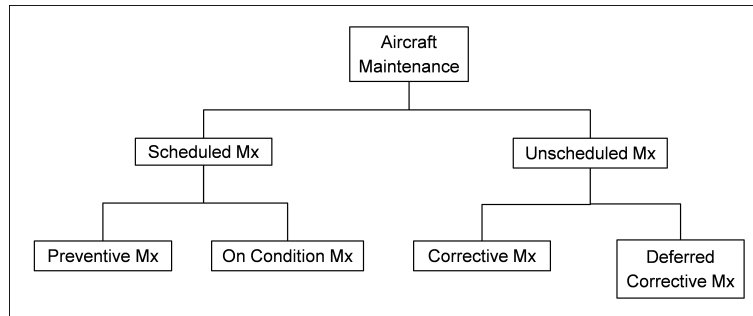


Figure 2.1: Maintenance Strategies [39]

There are two industrial processes which develop specific scheduled maintenance[25] tasks:

1. RCM (Reliability Centred Maintenance)
2. MSG-3 (Maintenance Steering Group)

RCM is a logical, structured framework for determining the optimum mix of applicable and effective maintenance activities needed to sustain the operational reliability of systems and equipment while ensuring their safe and economical operation and support. RCM provides the use of a decision logic tree to identify applicable and effective preventive maintenance requirements [40].

MSG-3 is an analytical methodology, officially recognized based on RCM. It is a process used by the airlines to develop maintenance programmes for commercial aircraft. The maintenance programme is the building block ensuring the continued airworthiness of an aircraft. The maintenance programme defines what scheduled maintenance tasks should be carried out and when those tasks should be carried out [40].

MSG-3 follows a task-oriented approach for the development of a maintenance program for commercial aircraft before entering into service. For each potential failure cause, the MSG-3 guidelines provide task-oriented logic to determine the appropriate scheduled maintenance tasks [25].

2.1.1 Task-oriented Maintenance Program

A task-oriented program consists of specific tasks, selected for a given functional failure consequence based on actual reliability characteristics of the equipment they are designed to protect.

Tasks are selected in a hierarchy of difficulty and cost, from lowest to highest [25]. Depending on the consequence of failure (safety, operational, economic, hidden safety and hidden non-safety) a single or combination of tasks will be selected. The following is the generic list of tasks to be selected:

1. Lubrication / Servicing: for the purpose of maintaining inherent design capabilities.
2. Operational / Visual Check: a failure finding task to determine if an item is fulfilling its intended purpose.
3. Functional Check / Inspection: functional checks are quantitative checks to determine if one or more functions of an item perform within specified limits. There are three levels of inspections to determine if an item is fulfilling its intended purpose:
 - General Visual Inspection
 - Detailed Inspection
 - Special Detailed Inspection
4. Restoration: reworking, replacement of parts or cleaning necessary to return an item to a specified standard.
5. Discard: the removal from service of an item at a specified life limit.

Under the MSG-3, maintenance tasks are categorized into three program groupings consisting of:

- Systems & power plant program
- Structural inspection program
- Zonal inspection program

The purpose of the systems & power plant program is to perform functional and/or operational checks on typical airplane systems i.e. flight controls, pneumatics, electrical power, etc. The structural inspection program is designed to provide timely detection and repair of structural damage during commercial operations. Detection of corrosion, stress corrosion, minor damage and fatigue cracking by visual and/or Non Destructive Testing (NDT) procedures are considered [25].

The objective of the zonal inspection program is to check the general condition of all systems and structures items involved in each zone by use of defined zonal inspection tasks. The zonal inspection tasks involve visual checks of electrical wiring, hydraulic tubing, water/waste plumbing, pneumatic ducting, components, fittings, brackets, etc., associated with the systems which are included within the zone boundaries [25].

2.1.2 Maintenance Plan

The maintenance plan details maintenance requirements and resources needed to maintain a specific piece of equipment or an overall system. Its developed from the maintenance concept and is refined over the life cycle of a piece of equipment or system [36]. The maintenance plan describes the following issues:

- Maintenance concept will be implemented
- Prescription of actions for each significant maintenance task that will be required for the system/ equipment during its life cycle
- Description of technical requirements (where and how maintenance will be performed)
- Incorporation of detailed support concepts and resource requirements
- Listing of the significant consumable items
- Listing of the supply, maintenance and recoverability requirements/sources for each repairable item

The maintenance plan is covered by a number of activities carried out by different departments.

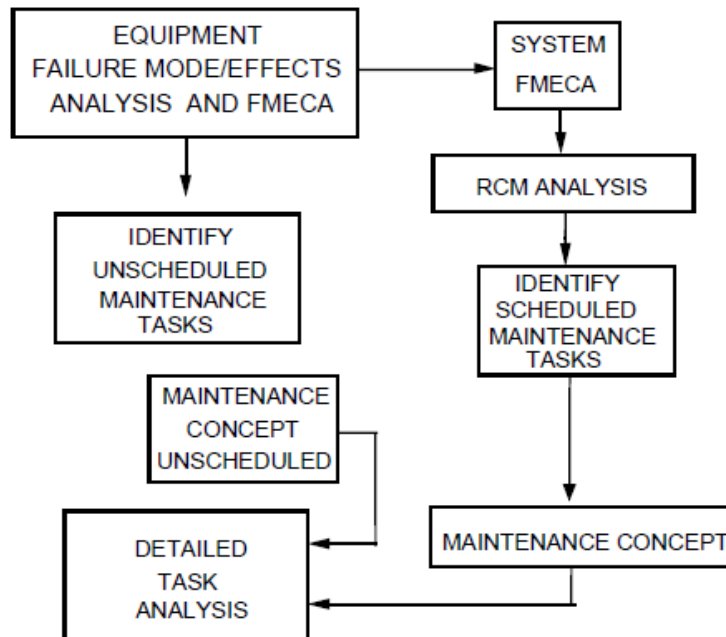


Figure 2.2: Maintenance plan [39]

First the maintenance task requirements and preliminary support resources are being identified and recorded, based on both the Failure Modes Effects Analysis (FMEA) and Failure Modes Effects and Criticality Analysis (FMECA) for unscheduled tasks. The FMECA provides a basis for establishing unscheduled maintenance actions and, as the driver for the RCM analysis, it identifies scheduled maintenance [39].

Based on the identified maintenance tasks maintenance concepts for the scheduled and the unscheduled maintenance actions are being defined. This step develops the maintenance tasks, to describe procedural steps required to complete the task and details the requirements for training, support equipment, spares, etc. The maintenance concepts conclude with the maintenance planning for the item under analysis [39].

2.1.3 Aircraft Maintenance Regulations

It is the prime objective of the manufacturer to see that when an aircraft is delivered to a customer, it is airworthy, meaning that it meets the requirements and conforms to type certificate and is in a safe condition for operation [22]. The type certificate reflects a determination made by the regulating body that the aircraft is manufactured according to an approved design, and that the design ensures compliance with airworthiness requirements. The regulating body compares design documents and processes to determine if the design meets requirements established for the type of equipment [19].

Maintenance regulations for commercial aircraft establish common technical requirements and administrative procedures for ensuring the continuing airworthiness of aircraft, including any component for installation. Continuing airworthiness covers all processes which ensure that at any time in its operating life, the aircraft complies with the airworthiness requirements in force and is in a condition for safe operation.

Aircraft maintenance is highly regulated. There are various airworthiness authorities around the world [29]. The major airworthiness authorities include:

- European Aviation Safety Agency (EASA) Europe
- Federal Aviation Administration (FAA) United States
- Civil Aviation Authority (CAA) United Kingdom
- Civil Aviation Safety Authority (CASA) Australia
- Transport Canada (TC) Canada
- Civil Aviation Administration of China (CAAC) China
- Directorate General of Civil Aviation (DGCA) India

In European civil aviation, EASA ensures that all civil aircraft operating within Europe are airworthy and safe. This legal framework arising from the convention of the International Civil Aviation Organisation (ICAO) is detailed in EU regulations [26]. This means that EASA issues aviation safety rules which are implemented the same way in all Member States following the agency’s mission of achieving a “high uniform level of civil aviation safety in Europe”. A summary of the regulations and their connections are shown in the following figure 2.3.

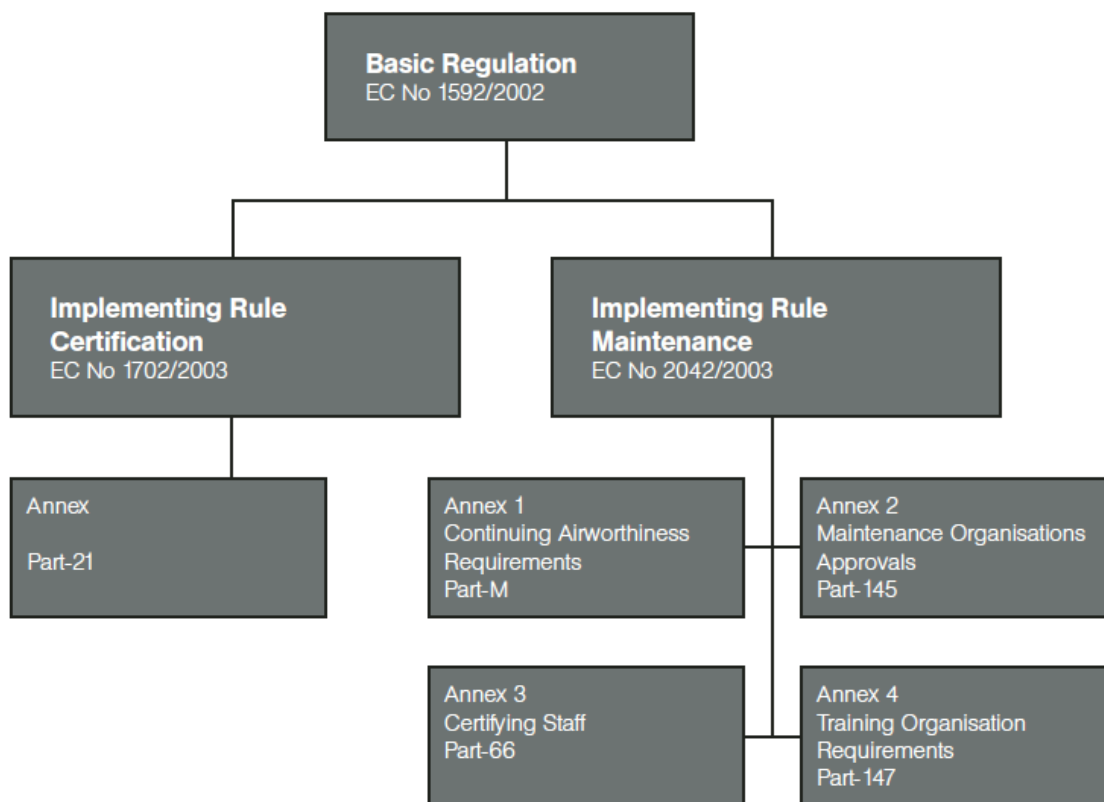


Figure 2.3: Aircraft maintenance regulations [27]

The aircraft maintenance regulations defined by the EASA cover continuing airworthiness requirements , approved maintenance organisations, maintenance personnel licensing and maintenance training organizations [27].

2.1.4 Challenges of Aircraft Maintenance

A survey was performed in 2010 to identify future aviation safety risks [28]. They include, among others, in-flight loss of control. Loss of control during flight may occur as a result of a stall, an icing-related event, a severe atmospheric turbulence, or a malfunction or failure

of a flight-critical system or equipment. Analysis of NTSB accident data and FAA incident data has established that system/equipment failures and malfunctions are significant contributing factors to aviation safety risk. In addition, the National Aeronautics Research and Development Plan cited several fundamental safety challenges that are relevant to preventing loss of aircraft control accidents caused by system or subsystem malfunctions or failures [21]. Among these are:

- To predict, monitor and check health of aircraft, at the material, subsystem, and component level, more efficiently and effectively.
- To integrate rapidly but safely technological advances in avionics, software, automation, and aircraft and airspace concepts of operation and operating procedures, by assuring their safety through a precise verification and validation process in a cost- and time-effective way.
- To create aircraft-level health management systems that can determine problems before accidents occur. Research in health management requires not only monitoring and detecting, but also confident prognostics of hidden potential failures before they arise.

Operation of aircraft involves a number of factors which contribute to the safety of aircraft. Although for every mode of travel there is an element of risk and danger to the traveling public it is more so in the case of air travel for obvious reasons.

The factors contributing to safety of air travel are of two type, "namely, the human factor and the machine factor i.e. the aircraft. It is the prime objective of the manufacturer to see that when an aircraft is delivered to a customer, it is airworthy, meaning that it meets the requirements and conforms to type certificate and is in a safe condition for operation" [22].

To overcome these challenges new technologies and ISHM applications for the health monitoring of aircraft components and the prediction of their remaining useful life need to be introduced in a broader way into new and legacy aircraft [31]. As a consequence, new approaches for the achievement of airworthiness approval for the installation of ISHM applications, for the certification of Mx Credits and for the full compliance with international safety standards, including continuing airworthiness and operations need to be established. Finally, an appropriate framework, so called MCF, needs to be defined to support all stakeholders in implementing all concerned activities.

The goal of ISHM is to provide tools and technologies for the automated failure detection, diagnosis and prognosis to mitigate undesirable events during flight. Undesirable events include those that arise from system, subsystem, or component faults or failures due to damage, degradation, or environmental hazards that occur during flight.

An essential objective of ISHM is to implement an advanced prognostics and health management (PHM) strategy that enables continuous monitoring and real-time assessment of vehicle functional health, predicts remaining useful life near failure components, and uses this information to improve operational decisions. Here, maintenance operations benefit from reduced occurrences of unexpected faults as the health management system shall provide early identification of failure precursors ewhile, simultaneously, Condition-Based Maintenance (CBM) is enabled, which can nhance availability, mission reliability, system life, and affordability [30].

Significant R&D progress has been done in the area of ISHM technologies in recent years. However real ISHM applications on aircraft board is still challenging and puts specific requirements on the ISHM application design and operation. These challenges include assurance of reliable and provable damage detection capabilities, taking over decision-making responsibilities instead of a human inspector and other challenges related to on-board installation and operation during the flight. Further, minimal weight and dimension, and system reliability and durability should be considered. Due to these and other challenging requirements the ISHM has not been widely implemented in aerospace industry yet [32].

2.2 Requirements Overview

This section gives a briefly overview of the collected requirements from the stakeholders involved in the development of maintenance concepts with ISHM and in the design, validation and certification of ISHM applications and correspondent Mx Credits. Departing from the needs towards the certification of ISHM applications and correspondent Mx Credits, stakeholder requirements and technical requirements have been identified.

2.2.1 Applicability

The stakeholder- and technical- requirements have been prioritized by the following criteria:

- Mandatory (M): absolutely important and necessary and so have to be performed
- Important (I): important to be realized or considered, but not mandatory
- Optional (O): useful or pleasant and should be fulfilled depending on feasibility, costs and so on

The use of "shall", "should", "must" and "will" shall observe the following rules:

- The word **SHALL** in the text denotes a mandatory requirement. Departure from such a requirement is not permissible without formal agreement.
- The word **SHOULD** in the text denotes a recommendation or advice on implementing such a requirement of the document. Such recommendations or advice is expected to be followed unless good reasons are stated for not doing so.
- The word **MUST** in the text is used for legislative or regulatory requirements (e.g. Health and Safety) and shall be complied with. It is not used to express a requirement.
- The word **WILL** in the text denotes a provision or service or an intention in connection with a requirement.

2.2.2 Stakeholder and Technical Requirements

The first set of collected requirements is derived from the needs of the stakeholders and focuses on the following areas:

- Development of aircraft maintenance concepts with ISHM
- Design of ISHM applications
- Achievement of airworthiness approval for the installation of ISHM applications
- Validation of Maintenance Credits
- Compliance with international safety standards, including continuing airworthiness and operations

In a next step technical requirements are derived from the collected stakeholder requirements, which form the basis for the specification and implementation of the MCF. The complete list of the collected stakeholder- and technical- requirements can be found in Table 2.1 The main issues addressed in the collected requirements are as follows:

- Definition of an adaptable Mx Credit Process model
- Integration of all ISHM related activities into Mx Credit Process coming from the following sub processes:
 - Development of New Aircraft
 - Development of Standard and ISHM-based Maintenance Concept
 - Development and Certification of ISHM applications
 - Certification of Mx Credit

- Access to the description of the modelled Mx Credit Process
- Access to complete Mx Credit Process history, including complete generated results per process step.
- Collection and easy access of all relevant data to certification authorities for approval of ISHM applications and correspondent Mx Credits
- Execution of the Mx Credit Process at runtime, respectively rule-based process execution
- Adding of content, annotations, data/documents links and results at runtime for each process step
- Traceability and Tracking of all relevant information during the execution of the Mx Credit Process
- Reporting of results for specific process steps or grouped activities
- In any of the above mentioned cases an access of the Mx Credit Process and of the content through a graphical representation should be also provided

Table 2.1 shows the technical requirements for MCF Tool. Priority for completion defined from 1 till 3:

1 - Mandatory

2 - Important

3 - Optional

| No. | Requirement | Priority |
|-----|--|----------|
| R1 | The Mx Credit Process Application shall provide during process execution access to all data and information related to the performed activities in the Mx Credit process | 1 |
| R2 | The Mx Credit Process Application shall provide access to all descriptions (all required regulations and standards per activity) of the modeled Mx Credit process | 1 |
| R3 | The Mx Credit Process application shall support the adding of content, annotations, data/documents (Upload) links and results at runtime for each process step | 1 |
| R4 | The Mx Credit Process application shall provide rule-based process execution (e.g. after submit, application guides user to the next correspondent step) | 1 |

| | | |
|------|--|---|
| R5 | The Mx Credit Process application shall provide access to complete Mx Credit Process data, including complete generated results per process step (after data submission in a process step no more changes are possible). | 1 |
| R6 | The Mx Credit Process application shall offer the functionality to implement a given Mx Credit Process model | 1 |
| R7 | The Mx Credit Process application shall provide for specific process steps a functionality to add documents | 1 |
| R8 | The Mx Credit Process application shall provide for specific process steps a functionality to add links (web links or links to data sources with a description) | 1 |
| R9 | The Mx Credit Process application shall provide for specific process steps the creation of tables, respectively rows, to modify and to delete them, if required. | 1 |
| R10 | The Mx Credit Process application GUI shall be easy and intuitive to use | 2 |
| R11. | During process execution the stakeholders shall be easily and intuitively guided along the Mx Credit Process at runtime | 2 |
| R12 | The Mx Credit Process application shall provide a graphical process view to retrieve current process status (submitted =>completed, active =>saved, open =>step not initiated) | 2 |
| R13 | The Mx Credit Process application shall provide a tabular process view to retrieve descriptions of the available Mx Credit Process | 2 |
| R14 | Users (Stakeholders) shall be associated to their respective process steps. | 3 |
| R15 | The Mx Credit Process application shall provide reporting of results for specific process steps or grouped activities | 3 |
| R16 | The Mx Credit Process application shall provide access to complete Mx Credit Process history (Login Date, Process Step ID, Process Step Access Date) | 3 |
| R17 | The Mx Credit Process application GUI shall be web based | 3 |
| R18 | The Mx Credit Process application shall provide an easy and intuitive modeling of the Mx Credit Process | 3 |
| R19 | The access to the description of the modelled Mx Credit Process shall be easy and intuitive. | 3 |
| R20 | Users (Stakeholders) shall be classified by user groups. | 3 |
| R21 | The Mx Credit Process application shall include a global model of related components, sub components and items | 3 |

| | | |
|-----|--|---|
| R22 | The Mx Credit Process application shall provide for specific process step views text fields. | 3 |
| R23 | The Mx Credit Process application shall provide for specific process step views correspondent drop down menus. | 3 |
| R24 | The Mx Credit Process application shall provide a login functionality to the users. | 3 |
| R25 | The Mx Credit Process application shall provide a simplified user rights management | 3 |

Table 2.1: Technical requirements list.

The collected requirements form together with the State-of-the-Art study, described in chapter 3 and the concept for the implementation of the MCF, described in chapter 4, the basis for the specification and the implementation of the Mx Credit application, described in chapter 5.

Chapter 3

Knowledge Base

In this chapter, we will check out the current knowledge base. This will include the research currently conducted in the field of process management. Based on the requirement mentioned in the requirement overview section in chapter 2, following the exploration of different approaches in academic literature, we focus at the proposed models in research specifically those process management type systems.

First, we analyze current Business Process Modeling tools in order to find out which features can be included in our application. Also there is an initiative to find out the solution approach of the existing modeling tools in contrast to the problems mentioned in chapter 2.

Second, we study a set of tools mainly focused on current web technologies, database server and important technologies that are related with the development of MCF Tool, in section 3.3 .

3.1 Existing Process Modelling and Execution Tools

In this section, we analyzed the current Business Process Modeling tools in order to know the features they provide. We focused on the tools that are open source. After a brief overview of features each tool provides, we documented the advantage and disadvantage in contrast with the requirement of MCF tool. Also, we have made the comparison list of the tools we have analyzed. In section 3.2, we presented the comparison of technology study and tool analysis.

3.1.1 Activiti

Activiti is a light-weight workflow and Business Process Management (BPM) platform targeted at business people, developers and system admins. Its core is a super-fast and rock-solid BPMN 2 process engine for Java. It provides an advanced Web interface shown in Figure 3.2 which can be used to start or manage processes and ad hoc task and forms support. Figure 3.1 explains different Activity components:

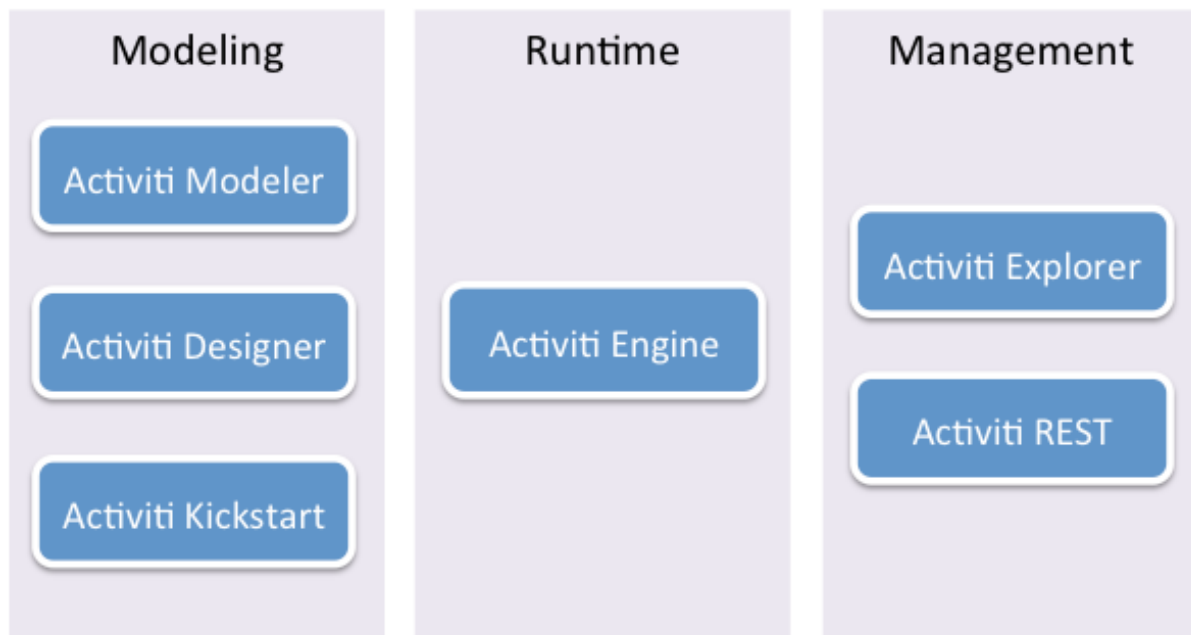


Figure 3.1: Components of Activiti [8]

- Modeler, a web-based graphical workflow authoring interface.
- Designer, an Eclipse plug-in for developing workflows.
- Engine, the core workflow processor. property, value and the operation.
- Explorer, a web tool to deploy process definitions, start new process instances and carry-out work on workflows.

It is possible to create process steps and apply the rules accordingly in the eclipse plug-in, the Activiti designer. For the web form of the process steps it has to be added manually. So the web form creation part is not available in the eclipse plug-in. After creating the Process steps a .war file can be exported. And that .war file has to be imported in the activity explorer. Table 3.2 gives the list of features provided by Activiti.

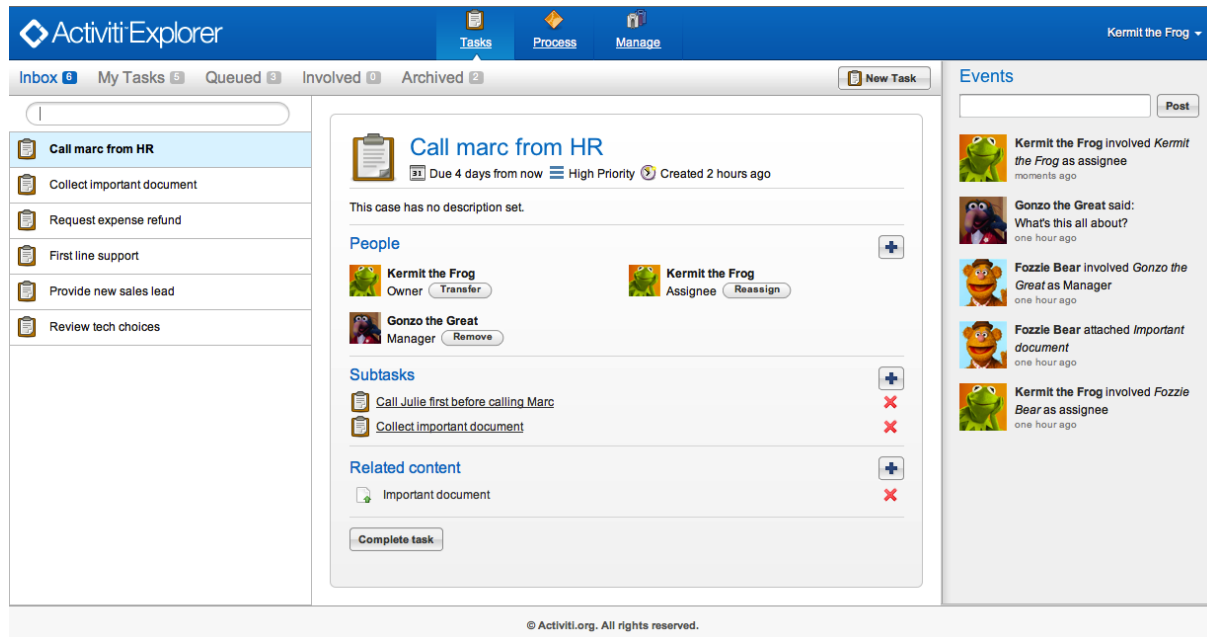


Figure 3.2: Activiti explorer [9].

Activiti benefits and problems are listed in table 3.1. We can create process steps and apply the rules accordingly in the eclipse plug-in. Process steps user interface i.e. HTML files, has to be created manually. And then it has to be linked with the process steps in eclipse plug-in. So the web form creation part is not available in the eclipse plug-in. After creating the process steps, a .war file can be exported. And that .war file has to be imported in the activity explorer. Based on the user access the user can access the process steps.

| Activity | |
|--|---|
| Pros | Cons |
| <ul style="list-style-type: none"> 1. It's an open source workflow engine. 2. easy to install and has a good documentation 3. Provides a good eclipse plug-in to create process flow. 4. Provides a Front end “Activity explorer” to manage tasks and User groups. 5. Activity Engine can be accessed by REST interface to query and control the application. 6. Supports database MySQL,Oracle. | <ul style="list-style-type: none"> 1. Eclipse plug-in does not provide a way to create forms. So, creating web forms with dragging and dropping the widgets in the web forms is not possible in the plug-in. 2. It is complex to customize Activity explorer. |

Table 3.1: Pros and cons of Activiti

So for Activiti, a stand-alone application has to be created for the MCF tool and it will use REST API to communicate with the Activiti engine. To create forms for the process steps, a web form generator is needed. As mentioned in the table 3.1 cons section, it is difficult to customize the activity forms. Also it is not possible to add new web controls inside the page. In the eclipse plug-in only limited number of web elements are available to use in the pages.

| Activiti | |
|------------------------------------|---|
| Vendor | Alfresco and the Activiti community. |
| License | Apache License |
| Supported Database | h2, mysql, oracle, postgres, mssql, db2 |
| Framework | Written in Java. |
| Process Modeling | There is a eclipse plug-in to design the process. |
| Visualization of Process execution | It provides a generated image and highlights in red the current state of the workflow. But from the requirements the states has to be clickable. Once user selects on a particular state the details of the process should be available |
| Rules | Rules can be applied in the eclipse plug-in process designer. |
| User Roles Management | In the eclipse designer groups with user can be created and can be assigned to the process steps. |
| API | It includes a REST API to the Activiti Engine that can be installed by deploying the activiti-rest.war file to a servlet container like Apache Tomcat. Using the REST API we can create and access Process steps details, insert Forms data(user given values) and get forms data. |
| Reporting | Activiti Explorer ships with some report examples and has the capability to easily add new reports to the system. For more customizable report , there we have to use some third party report generators. |
| Web Interface/Portal | Provides a web interface. |
| Content Search | Available |
| Adding content,forms,hyperlinks | Activiti supports two strategies to work with forms: Build-in form rendering with form properties and external form rendering. |
| User Community | Strong, gives a good on-line support through forums and blog. |

Table 3.2: Important features of Activiti

3.1.2 jBPM

jBPM is a Business Process Management (BPM) Suite. It is light-weight, open-source and written in Java. It allows to model, execute, and monitor business processes throughout their life cycle ¹. It provides Eclipse-based and web-based editor to model the business process. Figure 3.3 shows the Eclipse editor of jBPM.

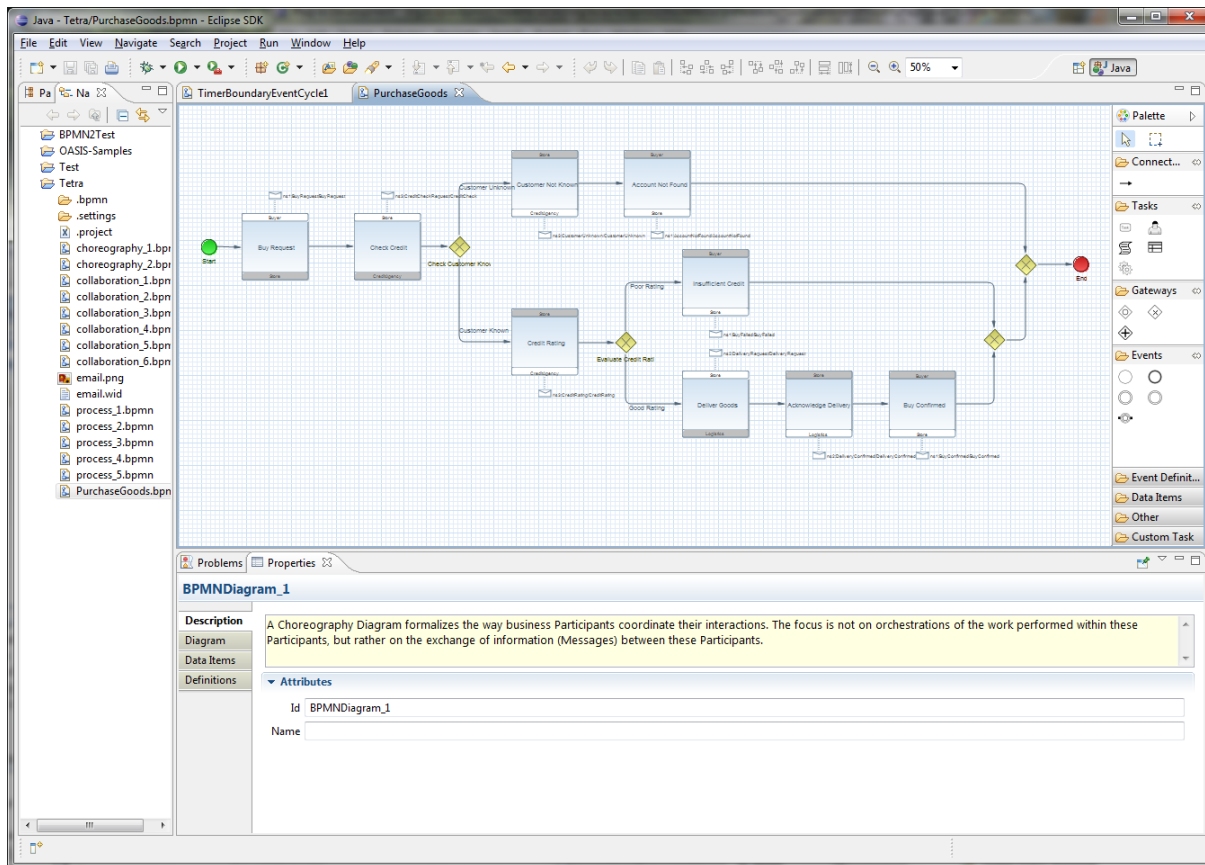


Figure 3.3: BPM Eclipse Editor for process

Table 3.3 gives the list of features provided by jBPM.

¹<http://docs.jboss.org/jbpm/v6.2/userguide/>

| jBPM | |
|------------------------------------|--|
| Vendor | jBoss |
| License | Apache License 2.0 |
| Supported Database | h2, mysql, oracle, postgresql, db2, derby, hsqldb, sqlserver |
| Framework | Java EE |
| Process Modeling | Provides an Eclipse plug-in to design the process. |
| Visualization of Process execution | Provides a Business Activity Monitoring (BAM). It allows to visualize and monitor process execution |
| Rules | Rules can be applied in the eclipse plug-in process designer. |
| API | Provides REST API to manage processes and tasks |
| Reporting | By adding a history logger to the process engine, all relevent events are stored in the database. This history log can be used to monitor and analyze the execution of your processes. |
| Web Interface/Portal | Provides a web interface. |
| Adding content,forms,hyperlinks | The jBPM Form Modeler is a form engine and editor that enables users to create forms to capture and display information during process or task execution. |
| User Roles and Rights Management | In the eclipse designer groups with user can be created and can be assigned to the process steps. |
| User Community | Strong, gives a good on-line support through forums. |
| Logging | Logging framework can be added as per choice. |

Table 3.3: Important features of jBPM

3.1.3 Bonita BPM

Bonita BPM is an open-source business process management and workflow suite. Bonita BPM is open source and can be downloaded under GPL. Bonita BPM is supported and developed by the company Bonitasoft. Bonita BPM has three major components:

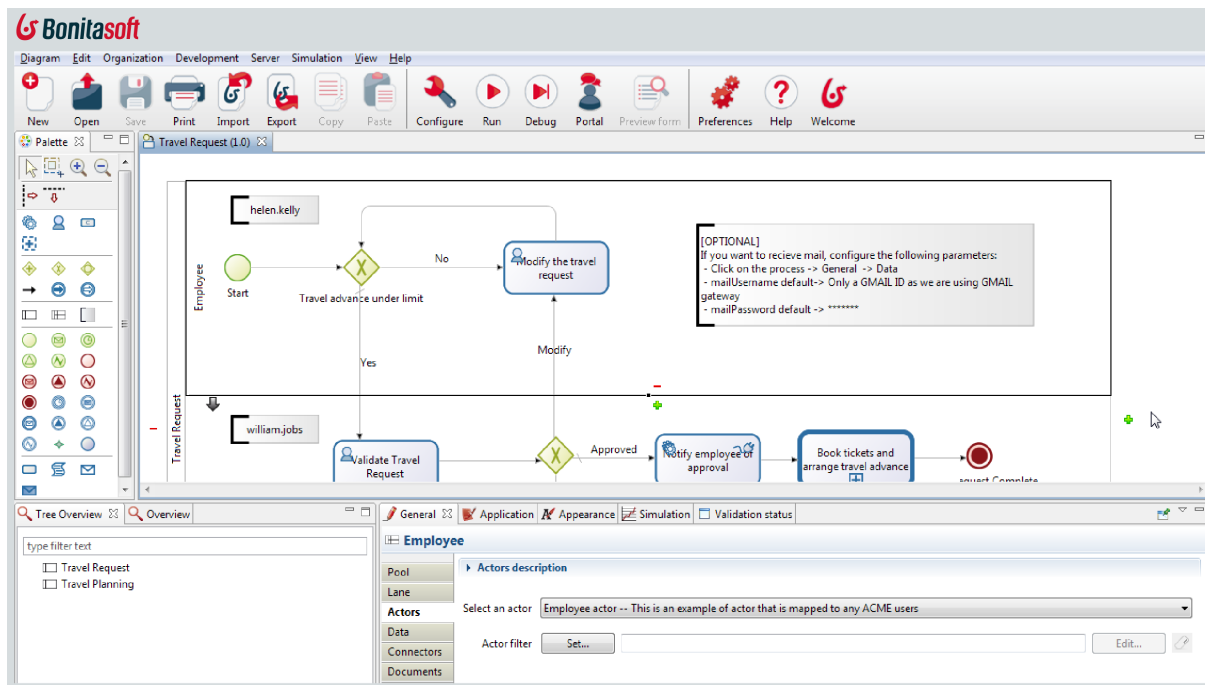


Figure 3.4: Bonita BPM Studio - Graphical BPMN 2.0 Workflow Designer [13]

Bonita Studio

Bonita Studio allows the user to graphically modify business processes following the BPMN standard. The user can also connect processes to other pieces of the information system (such as messaging, enterprise resource planning, enterprise content management, and databases) in order to generate an autonomous business application accessible as a web form. Bonita Studio also allows the user to design graphically the forms that will be shown to the end user in order to interact with the process. Moreover, the Studio allows the user to get started with processes designed with other standards and technologies such as XPD or jBPM. It relies on Eclipse.

Bonita BPM Engine

The BPM engine is a JAVA API that allows you to interact programmatically with your processes. It is available under LGPL. It relies on Hibernate.

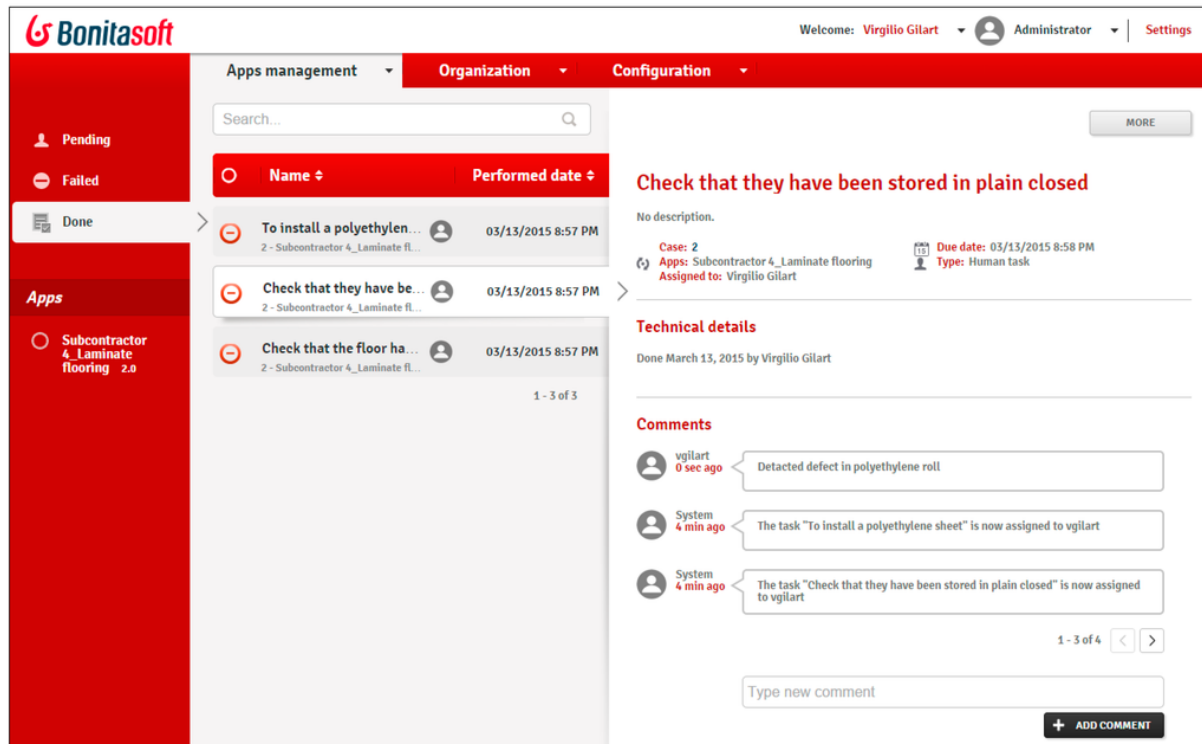


Figure 3.5: Bonita BPM Portal [12]

Bonita Portal

Bonita portal allows each end-user to manage in a webmail-like interface all the tasks in which he or she is involved. The portal also allows the owner of a process to administer and to get reports about processes. It relies on GWT. Table 3.5 gives the list of features provided by Bonita BPM.

| Bonita BPM | |
|--|--|
| Pros | Cons |
| <ol style="list-style-type: none"> 1. Provides a good studio to create process flows and create forms according to the process steps. 2. Create forms and drag and drop capability of widgets into the forms is available in the Bonita studio. 3. Provides a good API for the Bonita BPM Engine. | <ol style="list-style-type: none"> 1. Supporting external web form feature is not available in Bonita community edition 2. As the codes for process and associated web forms are generated by Bonita, it gives a very little control over that to customize. |

Table 3.4: Pros and cons of Bonita BPM

We can create process steps and apply the rules accordingly in the Bonita Studio. For the web form of the process steps it is possible to create from Bonita Studio or add it manually as a redirected URL. It is difficult to customize the Bonita Portal with given requirements, as it gives little control to do that. There is an option to create a standalone web application using the REST API provided by Bonita Engine.

| Bonita BPM | |
|------------------------------------|---|
| Vendor | Bonitasoft |
| License | Bonita BPM is open source and can be downloaded under GPL. |
| Supported Database | h2, mysql, oracle, postgres, mssql, db2 |
| Framework | Written in Java. |
| Process Modeling | Bonita studio provides a studio where process can be modeled |
| Visualization of Process execution | Visualization of process has to be done manually as Bonitasoft does not provide a graphical way to visualize the process flow. |
| Rules | Rules can be added in Bonita studio |
| API | Bonita provides Web API for Web clients of Bonita BPM Engine |
| Reporting | For reporting third-party software components can be added |
| Web Interface/Portal | Bonita portal provides a dashboard in the Bonita portal. |
| Adding content,forms,hyperlinks | Using Bonita studio it is possible to create forms and there is another option to assign the externally created web form to work steps. |
| User Roles and Rights Management | Groups with user can be created and can be assigned to the process steps |
| Logging | Logging framework can be added |
| UI Customization | It is difficult to customize the UI of the Bonita portal |

Table 3.5: Important features of Bonita BPM

3.1.4 Model Based UI Approach

A model-based approach is to identify useful abstraction highlighting the main aspects that should be considered when designing interactive applications. Central to all model-based approaches is that all aspects of an user interface design are represented

using declarative models. The main component is the Interface Model which includes different declarative models.

As depicted in the Figure 3.6 the Cameleon framework describes different layers of abstractions.

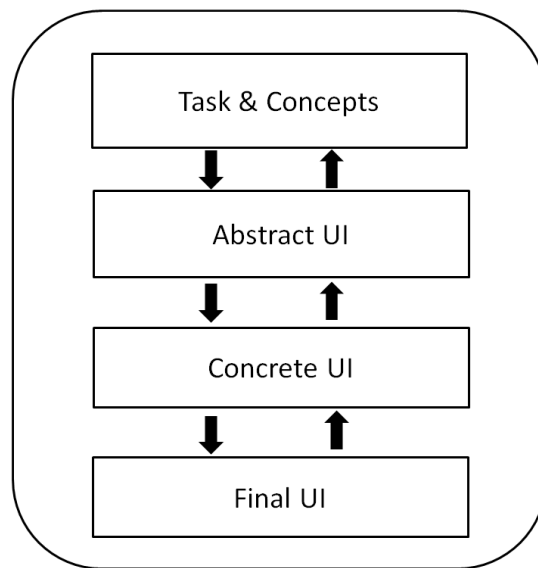


Figure 3.6: The Cameleon Reference Framework [2].

- **Tasks and Concepts**, finds out the hierarchies of the task that need to be performed in a specific order to carry out user goals
- **The Abstract User Interface**, expresses the UI in terms of abstract interaction objects. These objects are independent of any platform.
- **Concrete UI**, expresses the UI in terms of Concrete Interaction Objects. It defines more concretely how User Interface perceived by the users. The property, value and the operation.
- **Final User Interface**, can be represented in any UI programming language or markup language. So a final User interface can be compiled.

Model Base UI benefits and problems are listed in table 3.6.

| Model-Based User Interfaces | |
|---|---|
| Pros | Cons |
| <p>1.This approach reduces gap between requirements and implementation. Modelling phase focuses to ensure in advance that the implementation addresses the user-centered requirements. MBUID defines models related to the UI that are captured and updated throughout the development life cycle [20].</p> <p>2. A portion of the final UI can be directly generated and used in the final system</p> <p>3. Improves quality as generated code has always the same elements in the same places. Repeating tasks exactly in the same way every time is easier.</p> <p>4. Generated code contains minimal generation errors.</p> | <p>1.More code generators are needed to produce UIs for different kind of applications. Few tools are available on the market and are limited to specific context of usage.</p> <p>2 Lack of standards like UML for software design or architecture. There is no standard for UI development. At least in software design there is a common agreement in the notation: UML.</p> |

Table 3.6: Pros and cons of Model-Based User Interfaces Approach

To have the model-based UI benefits to our application, we decomposed the tasks into subtasks. For the MCF Tool, hereinafter also called Mx Credit application, we define the tasks that had to be performed on the process steps. In the abstract user interface level, we identify the interaction objects to support the objects. For the concrete user interface, we have used JSON schema to consider specific interaction objects supported. For example, to give the process step description we need to have a text box, so in the JSON schema. we define the text box. For the final user interface, we have used a transformer component that will transform JSON schema based Abstract UI to Final UI. We describe the flow of transformation details in section 5.4.

3.2 Comparison of identified Technologies and Tools

Based on the requirements in table 2.1, we have compared the tools. The summary of comparison is shown in table 3.7. From the comparison, we can observe that, BPM tools like Activity, Bonitasoft, jBPM does not provide all the functionalities that are necessary for Mx Credit Framework. Although, these BPM tools provides extensive rule-based process execution but does not give the much flexibility for requirements like customized user interface generation [R3], customized process steps status initialization [R12], process navigation i.e. to access executed and not executed process steps of Mx Credit process steps [R1].

[R5]- Access to executed process steps is an important requirement for MCF tool. Activity does not allow to access the process steps that are executed. It is also same for Bonita and jBPM. Accessing all the process steps of a modeled process step is not possible with these BPM tools. Model-Based UI gives a convenient way to manage this requirement. Likewise, for the requirement, provide reporting results for specific process steps [R15], BPM tools does not present a good option. Also, from the MBUI approach, we could not find out a proper solution.

From the table 2.1, we can identify that Model Based User Interface approach supports most of the requirements of the Mx Credit Framework. MBUI gives the flexibility to model the process execution, user interface generation for the process steps. Although, it provides little support for rule-based implementation for Mx Credit process [R4], we can observe from the requirements described in section 2.2, that Mx Credit process does not require a significant rule-based implementation [R4]. Simple rule-based implementation is possible with MBUI. Furthermore, MBUI provides adaptive user interface generation, that helps to cope with the complex process modeling and execution environment of MCF.

Hereafter, an overview of the existing BPM tools was obtained through this survey. A couple of tools were analyzed in detail. Also, we have analyzed Model-Based User Interface approach in depth. From the comparison mentioned in table 3.7 and several articles that are summarized in this review, it can be seen that there is a limited option to select a specific tool for the Maintenance Credit Framework. On the other hand, MBUI approach provides a much more flexibility. We choose to follow the Model-Based UI approach to implement the MCF Tool. For the implementation of MCF Tool, we have also studied available technologies. These are described in detail in the section 3.3.

| Requirements | Activiti | jBPM | Bonita | MBUI |
|---|----------|------|--------|------|
| [R1] -Provide process navigation functionality over modeled Mx Credit process steps | No | No | No | Yes |
| [R2] -Access all descriptions of the modeled Mx Credit process | No | No | No | Yes |
| [R3] -User Interface customization | No | No | No | Yes |
| [R4] -Rule based implementation | Yes | Yes | Yes | No |
| [R5] - Access to executed process steps | No | No | No | Yes |
| [R6] -Functionality to Model Mx Credit Process steps | Yes | Yes | Yes | Yes |
| [R7] -Add document for specific process step | No | Yes | No | Yes |
| [R8] -Add links for specific process step | No | No | No | Yes |
| [R9] -Create tables for process steps to view and modify data | No | No | No | No |
| [R12] -Provide customized process status | No | No | No | Yes |
| [R13] -Tabular process view for Mx Credit Process status | No | No | No | Yes |
| [R15] -Provide reporting results for specific process steps. | Yes | Yes | No | No |

Table 3.7: Comparison of Existing Tools based on the requirement list 2.1.

3.3 Existing Technologies

The objective of this section is to list out the potential technologies that are available and can be used to develop the MCF Tool. We study a set of tools that helps to improve the development. As it is a web based application we concentrate on the web related technologies.

3.3.1 AngularJS

AngularJS.² is an open source JavaScript framework that imposes the Model-View-Controller software pattern[14] onto client web-applications. It supports the creation of single-page applications that only require HTML, CSS, and JavaScript on the client side. AngularJS extends HTML by providing directives that add functionality to markup and that allows to create powerful dynamic templates. It is also possible to create customized directives, which is a one of the most powerful features of AngularJS. It also gives the ability to define the binding between the data in the scope and contents of the views. With ngModel directive bidirectional binding in AngularJS is possible ³. Below, a list of important features of AngularJS is shown:

1. Two way data binding
2. Templates
3. Dependency injection
4. Directives
5. Services

AngularJS provides RESTful functionality in the ngResource module. Using this functionality we can access the API of Neo4js to access data. Angular is maintained by Google and its engineers [34].

3.3.2 Node.js

Node.js⁴ is a runtime system for creating mostly server-side applications. It is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. It's also a runtime system that makes it easy to build a network or other event-driven application servers.

²<https://angularjs.org/>

³<https://docs.angularjs.org/guide/concepts>

⁴<https://nodejs.org/en/>

It's created to manage asynchronous I/O from the ground up and is a good match to a lot of common web- and network-development problems. To scale to large volumes of clients all I/O intensive operations in Node.js are operated asynchronously.

Same language JavaScript can be used on the back end and front end. So it breaks down the boundaries between front and back end development. In addition to its good features Node.js has a good open source community which provides good modules to add additional capabilities to Node.js applications.

3.3.3 Express.js

Express.js⁵ is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates a rapid development of Node based Web applications. It allows to set up middle wares to respond to HTTP Requests. It defines a routing table which is used to perform different actions based on HTTP Method and URL.

3.3.4 JSON

JSON (JavaScript Object Notation) ⁶ stands for JavaScript Object Notation which is light-weighted designed to easily read and write. Its easy to generate and parse for machines. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages.

JSON is on two structures:

- A collection of name/value pairs. In different languages, this is executed as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

However, it does not require JavaScript to read or write because it is made in text format which is language independent and can be run everywhere. JSON notation contains these basic elements which are also used in javascript:

1. Object: n array is an ordered collection of values. An array begins with [(left bracket) and ends with] (right bracket). Values are separated by , (comma).
2. Arrays: Arrays begin and end with braces and contain different values.

⁵<http://expressjs.com/>

⁶<http://www.json.org/>

3. A value can be a string, can be an object, an array, or the literals.
4. Strings are surrounded by double quotes and contain Unicode colon (:)

3.3.5 Neo4j

Neo4j is an open-source graph database which implements Graph Model conveniently to the storage level ⁷. It has a built-in REST web API interface that can be accessed by programming language. Neo4j uses Cypher query language. Cypher is a declarative, pattern-matching query language that makes graph database management systems understandable [6]. It has many features like, supporting full ACID(Atomicity, Consistency, Isolation and Durability) rules and exporting of query data to JSON format.

Graph structure means that we will be using vertices and edges or nodes and relationships, to store data in a persistent manner. Graph database represents data without some of the distortions of relational data model and also apply various types of graph algorithms on these structures.

Some of the key characteristics of Neo4j specifically as a graph database management system are:

- It is a transactional, ACID-compliant database. As listed out in Figure 3.7 it provides features like Atomicity, Consistency, Isolation and Durability.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.
- It is made for On-line Transaction Processing. It supports systems where it is needed to retrieve data from the database management system in an on-line system environment.

Cypher is one of the defining features of Neo4j graph database. Cypher is a declarative, pattern-matching query language that makes graph database management systems understandable and workable for any database user. It is easy to formulate queries based on relationships. For JavaScript, there are drivers that provide an option for connecting to Neo4j. In our application we have used *neo4j-js* ⁸. It is a Node.js client library for accessing the database with batch support.

⁷<http://neo4j.com/developer/graph-database/>

⁸<http://neo4j.com/developer/javascript/>

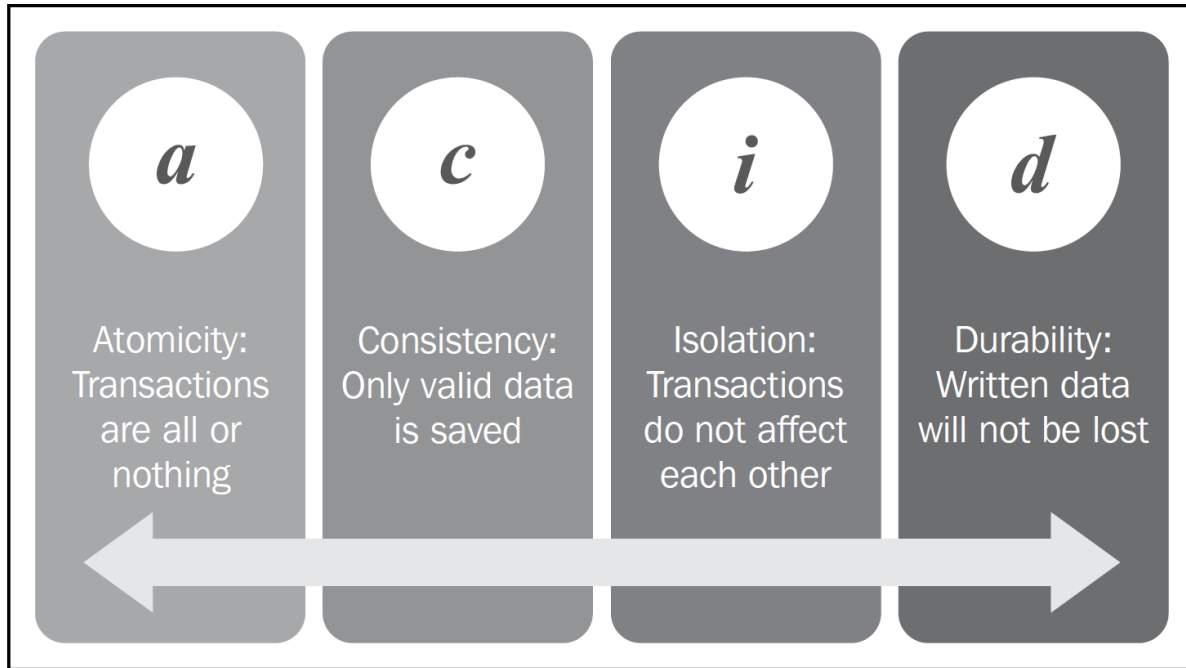


Figure 3.7: Neo4js is an ACID-compliant database

3.3.6 REST

REST, meaning representational state transfer, is a software architecture style targeted at distributed information systems. It ensures the scalability of web services. REST is defined by Roy Fielding in his dissertation, which is used as the main theoretical REST reference [38]. It goes through the constraints of REST and the properties they generate.

REST is formally described as a set of constraints applied to elements within the architecture. If followed, it ensures better scalability for web services:

Client-Server

The functionality of a server and a client should be separated. This improves portability of the client as it does not have to be worried of server specific things as data storage [38].

Stateless

The nature of the communication between clients and servers must be stateless. No client state should be stored on the server, instead clients should include all information needed to process a request.

Cache

Data within a server response must specify, implicitly or explicitly, whether it is cacheable or not. If a response is cacheable, a client is allowed to reuse that data for later requests. This can improve efficiency, as some interactions between the client and the server can potentially be eliminated.

Uniform Interface

The interface between components should be uniform to simplify overall system architecture. REST defines four interface constraints for this: *Identification of resources*, *Manipulation of resources through representations*, *Self-descriptive messages* and *Hypermedia As The Engine Of Application State*.

Layered system

Components should only see other components within the layer that they are interacting with. This constraint controls the complexity of the system.

Code-on-demand

A client can receive and execute code from a server, enabling extension of the client. This can simplify clients and improve system extensibility, as the number of features that have to be implemented on the client is minimized [38].

Although REST does not specify any specific protocols, HTTP is the most regularly used application protocol in association with REST. When using HTTP, resources are identified by URLs. Actions are performed on these resources in a uniform way by using a set of predefined verbs like GET, POST, PUT and DELETE. HTTP messages also contain headers that allow descriptions of the representations that they contain. HTTP headers can contain information about media type, cacheability, client state in the form of cookies, etc.

Chapter 4

Conceptual Design for Mx Credit Framework

4.1 Common Approach

Since the certification of ISHM applications and correspondent Mx Credits have unique challenges, these must be addressed during the early stage of technology development in order to successfully meet the requirements and fulfill the objectives for certification. To meet these challenges a concept for the implementation of the MCF has been developed by Airbus Group, which describes all necessary activities and provides involved stakeholders all required information along the development, the certification and the in-service life of the aircraft, equipped with ISHM applications.

The main approach allows to easily map the complex process environment and to guide involved stakeholders along the defined process steps (see figure 4.1). Just the required data related to the certification of Mx Credits and to the design, certification and usage of ISHM applications is being collected and processed.

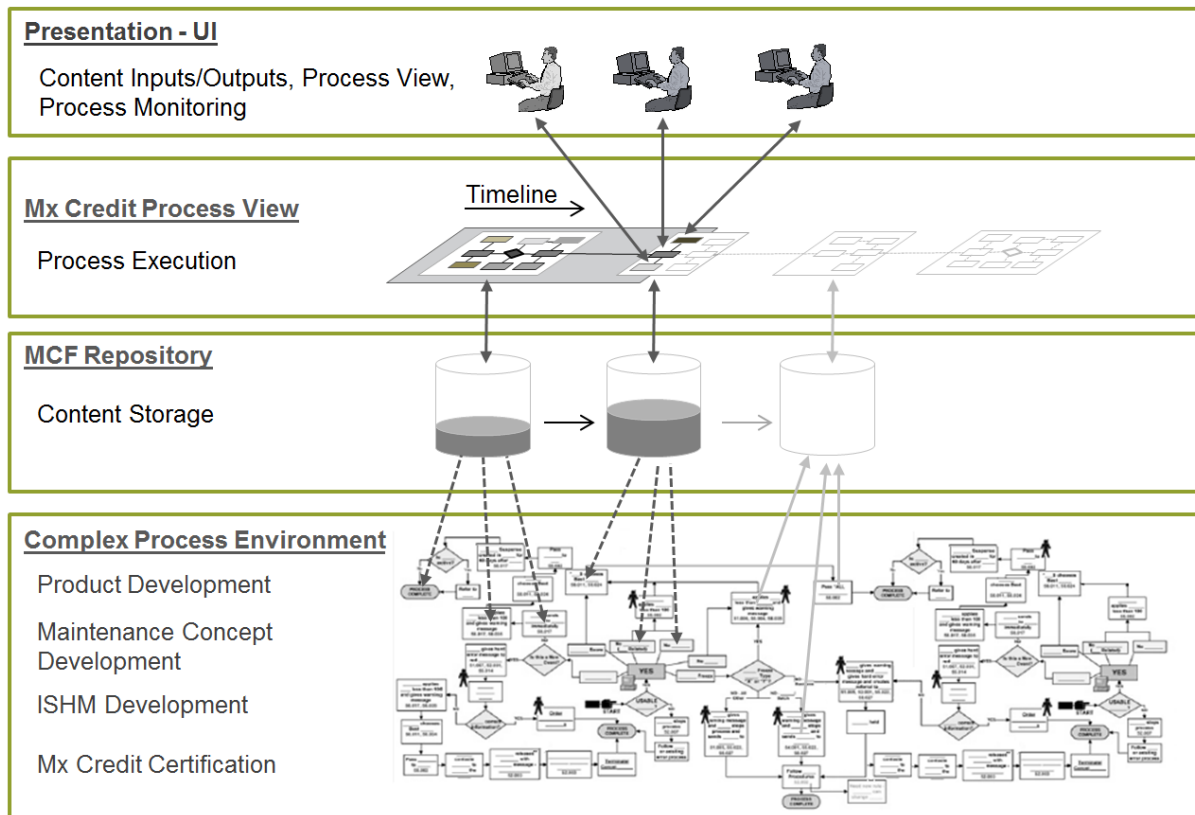


Figure 4.1: Process Environment of MCF.

4.2 Description of Mx Credit Framework

The objective of the Mx Credit Framework is to coordinate each stakeholder's activity and to develop the evidence base such that certification of both, the ISHM applications and the maintenance process dependent on it can be achieved. There are two main outputs of the MCF, the first is to provide the V&V criteria needed to provide the evidence base of the Mx Credit Certification and the second is to provide the requirements to the ISHM design office so they can design and certify ISHM applications with the functionality to meet the requirements for the Mx Credits and correspondent capabilities.

The MCF produces the evidence to certify the ISHM applications independently of the aircraft component's design authority requirement. ISHM applications and Mx Credits are certified based on preventative or corrective maintenance actions. This framework includes a definition of the stakeholder roles in the process and the sources of information for completing the data entries and analysis employed.

Important aspects of the MCF are:

- Modelling of the Mx Credit Process
- Access to the description of the modelled Mx Credit Process
- Execution of the Mx Credit Process at runtime, respectively rule-based process execution
- Adding of content, annotations, data/documents links and results at runtime for each process step
- Traceability of executed process at each point of time
- Reporting of results for specific process steps or grouped activities
- Access to complete Mx Credit Process history, including complete generated results per process step
- Navigation of modelled, executed or logged process

One of the most important objectives of the MCF is to coordinate the different Mx Credit Certification related activities in such a way, that two or more independent activities can be coordinated along the Mx Credit Process. There is no intervention of the MCF with the existent complex process environment intended. The access to the distributed data sources shall be made possible only in a “read” mode.

4.3 Description of Mx Credit Process

The defined Mx Credit process flow is generic. It is neither limited to a specific Mx Credit nor to a specific ISHM application, but it is used for the complete introduction of an ISHM concept into legacy or new developed aircraft.

The Mx Credit process describes all required activities for a developed ISHM System capabilities to meet the requirements of a proposed maintenance regime such as no preflight inspections or the removal or extension of fixed Mx intervals.

In the current Mx Credit Framework the process flow is divided into four swimlanes to support the analysis and the production of evidence to support four different objectives.

1. Swimlane: A/C Development Process to support the integration of ISHM applications
2. Swimlane: Mx Concept Development to develop maintenance concepts with ISHM
3. Swimlane: ISHM Development to support analysis to qualify the ISHM system to meet the requirements demanded by the Mx Credit

4. Swimlane: Mx Credit Certification process to support the analysis required to gain a Mx Credit

The main elements of the determined swimlanes are as follows:

1. Aircraft Development
 - (a) A/C System/Components Selection and System FMEA
 - (b) A/C Preliminary Design Phase including ISHM
 - (c) A/C Detail Design, Implementation and Integration Phases including ISHM
 - (d) A/C Type Certification including ISHM
 - (e) Airworthiness Directives and Post Modification
2. Maintenance Concept Development
 - (a) RCM with Mx Task Development
 - (b) Mx Task Development with ISHM
 - (c) Implementation of the Mx Concept
 - (d) Certification of the Mx Concept
 - (e) Continued Airworthiness Certification for the maintenance types to be adopted
3. ISHM Development
 - (a) ISHM Preliminary Design Phase
 - (b) ISHM Detail Design, Implementation and Integration Phases
 - (c) ISHM Type Certification including ISHM
 - (d) Continued Airworthiness Certification and ISHM Design Modification
4. Mx Credit Certification
 - (a) Preliminary Risk Assessment
 - (b) Mx Credit V&C Process
 - (c) Mx Credit Certification
 - (d) Post Certification and Mx Credit Modifications

The Mx Credit Process is being represented by an extensive data model to provide the different modes of navigation for each ‘view’. This in turn requires some additional data descriptors, like codes and naming conventions, to be defined and entered into the MCF to support the data model.

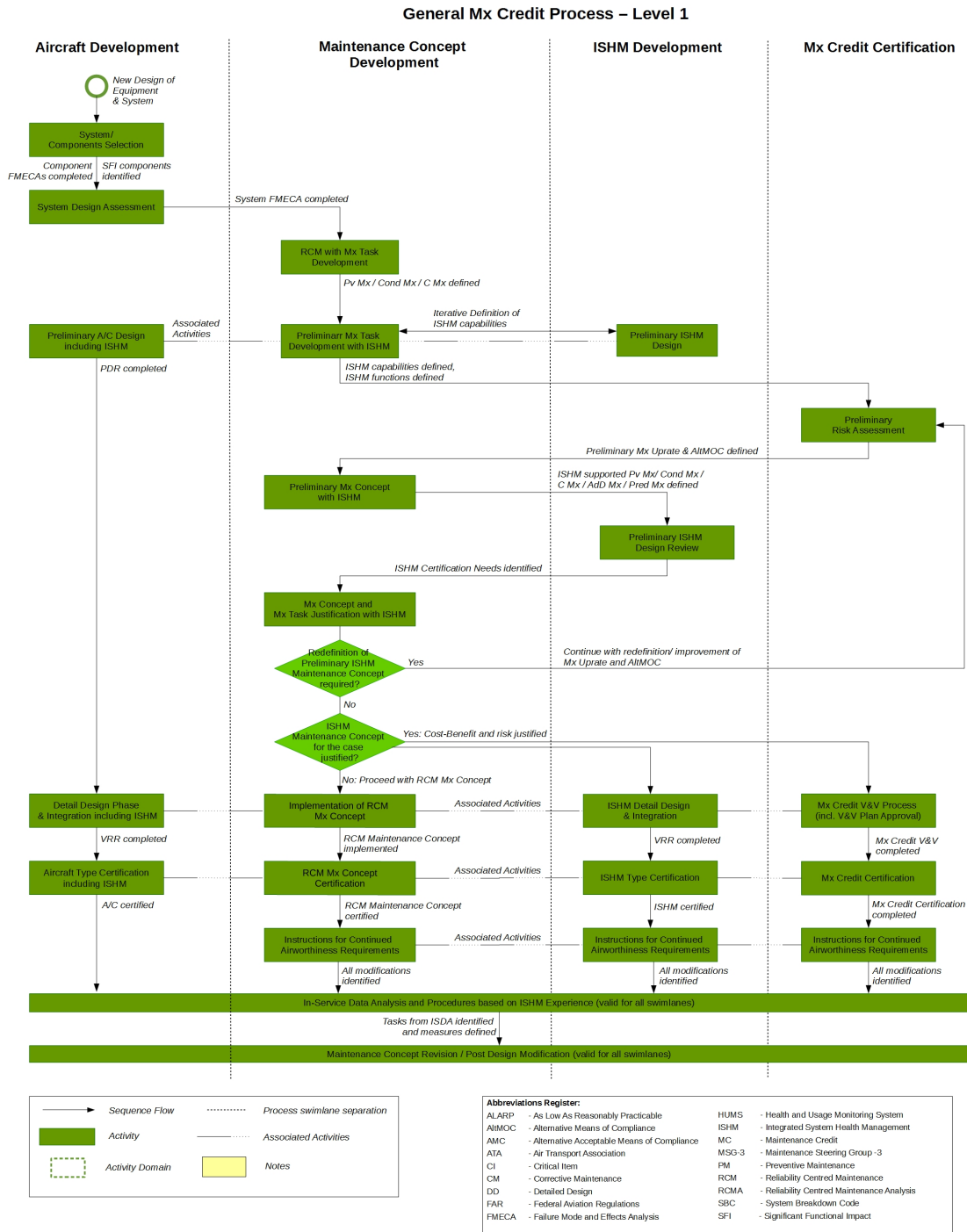


Figure 4.2: General Mx Credit Process [39].

There is also a need to provide a means of validating and authorizing the data entered into the MCF Tool. An introduced standard terminology shall be available to support the data entries with standard terms, available in drop down boxes, which support the filtering of the data base to provide summaries and comparisons along the process.

4.4 Involved Stakeholders

The involved stakeholders in the new defined Mx Credit Framework play a crucial role for the proper execution of the process steps. To identify stakeholder groups a review of stakeholder requirements was carried out.

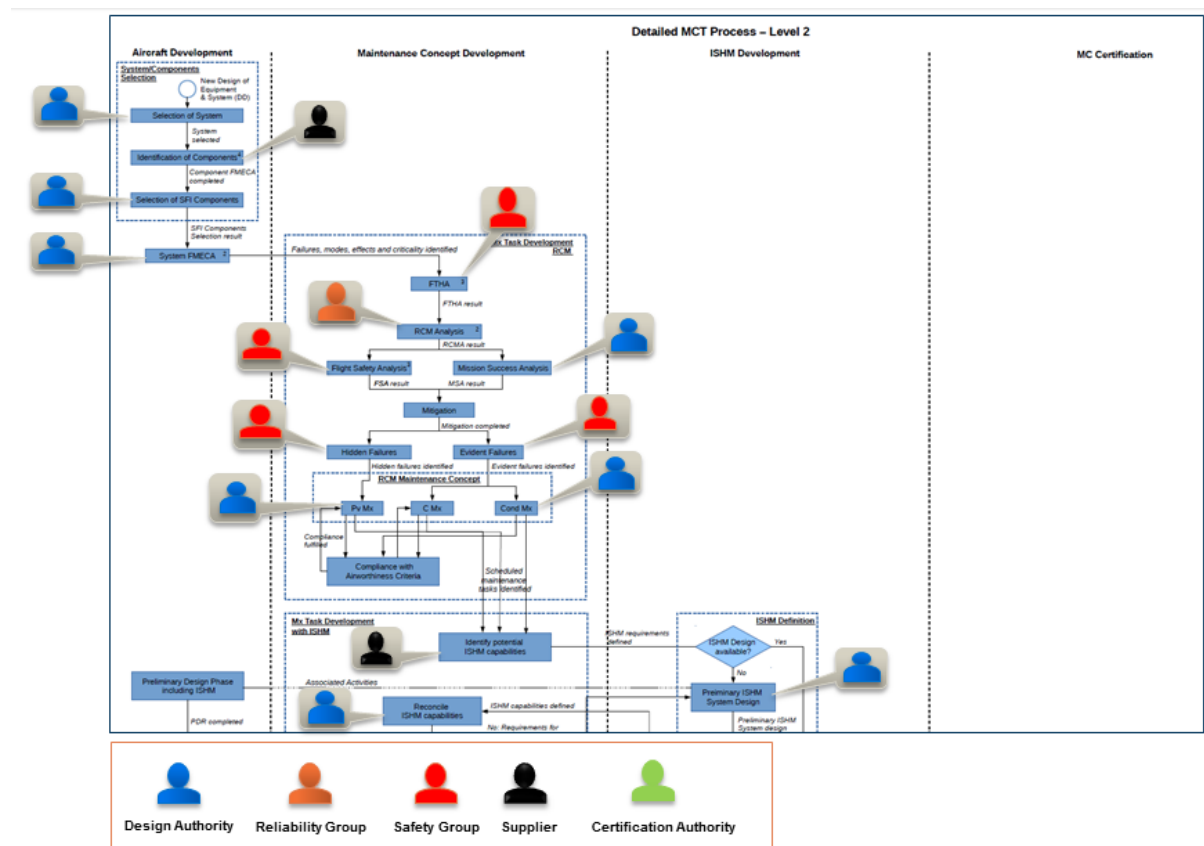


Figure 4.3: User group association with process steps [39].

The review invoked the concept of different ‘views’ for the different stakeholders in the process such that each stakeholder was only presented with the subset of process steps that is relevant to them in a format that the stakeholder recognizes. There are the following number of stakeholders in the process:

- A/C Design Dept
- RMT Dept (Reliability, Maintainability Testability)
- Safety Dept
- ISHM System Design Dept
- System Integration Dept
- Supplier (SHM System Manufacturer)
- Certification Authority
- ISDA Dept
- Tech Pubs Dept
- PMO (Program Mgmt Organization)

The objective of the MCF is to coordinate each stakeholder's activity and develop the evidence base such that certification of both the ISHM applications and the maintenance process dependent on it can be achieved.

4.5 Technical Concept for Mx Credit Application

The goal is to improve the process flow with respect to the complex process environment. There should be also an efficient way to associate content with it and apply user role management. In order to achieve it, the MCF Tool should provide an efficient way to model the process flow and content should be included with respect to the user role.

In this section, we present an overview of the proposed architecture. This overview includes the main problem solution areas, creating the process modeler for the process flow navigation and second Mx Process Engine for providing a mechanism to deal with the content with respect to process the flow.

4.5.1 Proposed Architecture

In this section, we present an overview of MCF Tool's proposed architecture. Figure 4.4 shows initial architecture. It contains three main components. First the Client, in our case it is the User interface. MCF Tool can be accessed from the standard web browser. So the users of the MCF Tool will use the web browsers as a user interface to use the system.

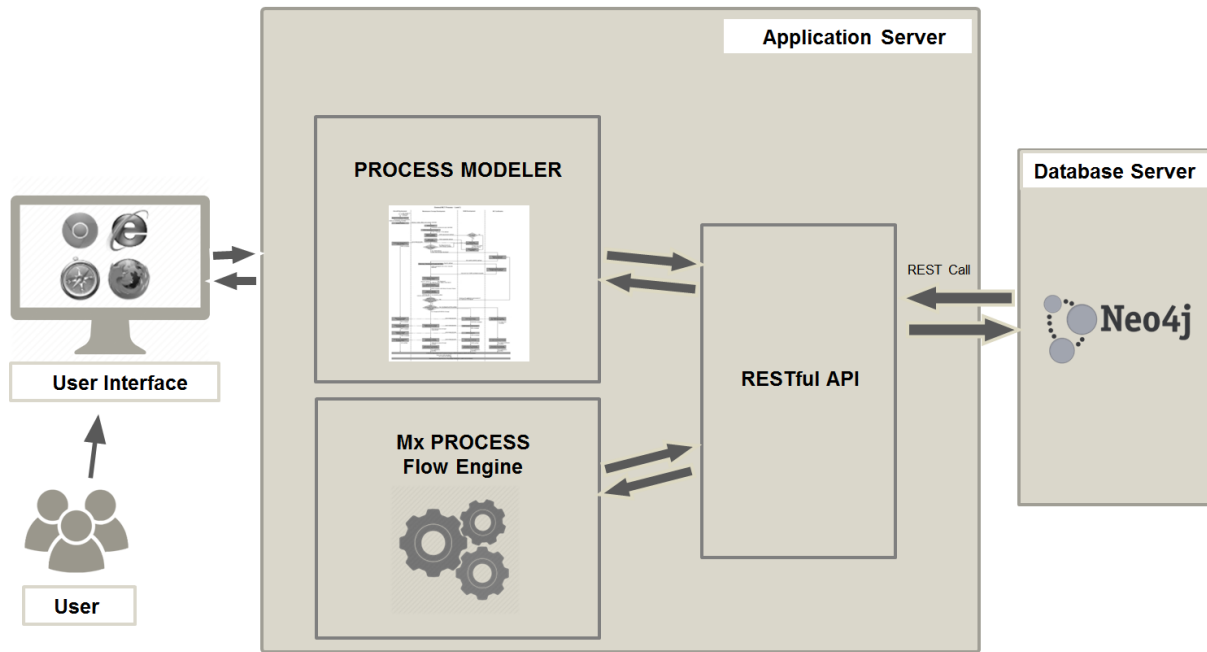


Figure 4.4: Process Flow with Model Based UI

Database server supplies organization's production data structure in a structured form. In our proposed architecture, we choose Neo4j database server to store and provide the process flow and related data with it. Neo4js is a graph database described in Section 3.3.5.

And the application server holds the functionality required by the MCF Tool. The three most important components for the application are a Process Modeler, Mx Process Flow Engine and API. The Mx Process Flow Engine is described in detail in section 4.5.2. And the Process Modeler is described in section 4.5.3. Both of these two components make the request for data through RESTful API. This API provides a better way to store and retrieve data from the Neo4js database server. Because of the REST functionality, it gives us the opportunity to extend the application also to other platforms in the future.

4.5.2 Management of Process Flow

Based on the fundamental architecture shown in section 4.5.1, Process Flow Engine is responsible for managing the flow of the process steps based on the rules applied when creating the steps template. The primary key to load the steps is based on component id, as each component has its own process steps attached with it. Figure 4.5 shows the responsible objects and their communications.

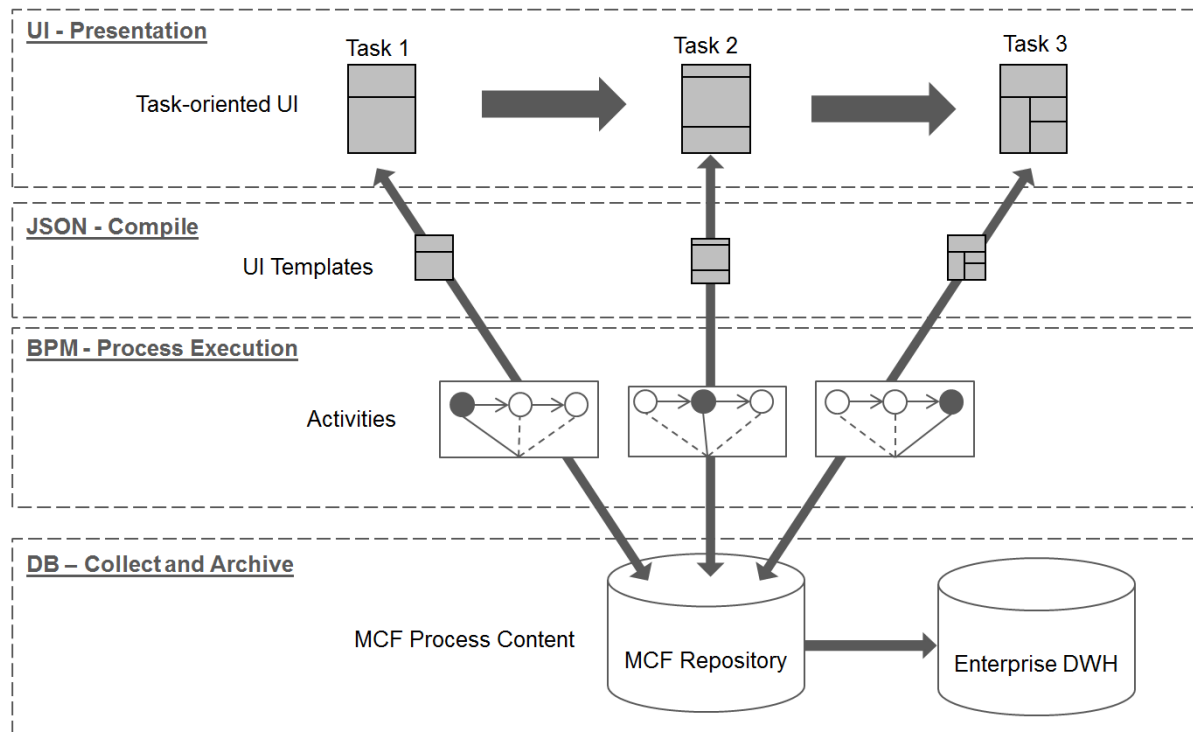


Figure 4.5: Process Flow with Model Based UI.

So when a user selects a component from the component dashboard View, ProcessEngineController gets the id of the selected component. It passes the id to the processStepService. processStepService makes the communication with the database API and collects the respective process steps. After getting the process steps list it returns the currently active process step to the ProcessFlowEngine.

Each process step has its own JSON schema for the data field. ProcessFlowEngine passes the JSON schema to the modelBasedUI controller to generate the forms for the process step. And after that, it is the responsibility of the Model-Based UI to generate the respective user interface.

4.5.3 Approach for Authoring and Publishing of Mx Credit Process content

To describe the Mx Credit process content, we present the structure of the content of MCF Tool in this section. It additionally, outlines the relationship between the contents and its access control for different groups of users of the tool. Figure 4.6 shows the content structure of the MCF Tool. It has to be noted, that a *Component* could have been different

Sub Components. And also *Sub Components* have the *Items* which is a non splitting unit. For every *Component* there are *System* and *Sub System layer* over it.

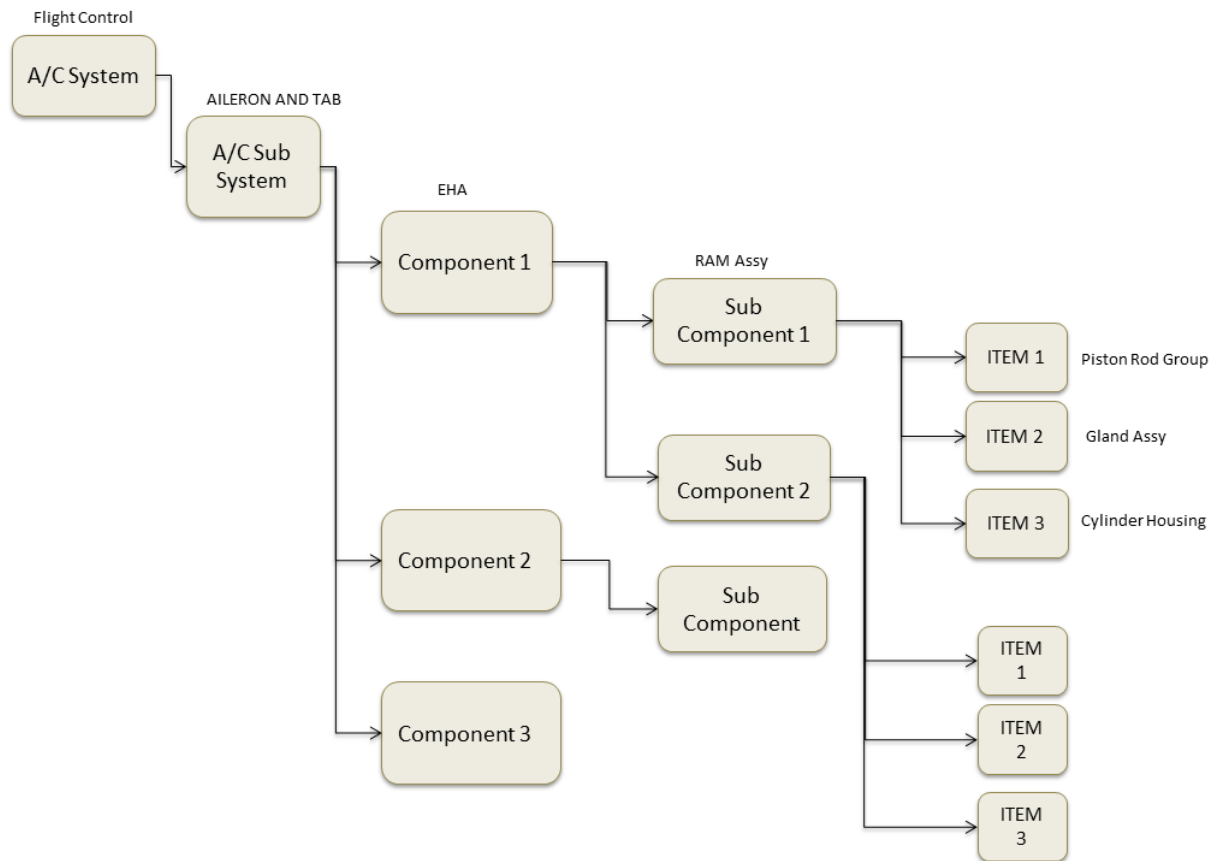


Figure 4.6: Relationship of the Components, Sub Components and Items of the MCF Tool [39].

Mx Credit Process runs as a base from the Component level. So each Component has a set of process steps. In every process step, the user gives the data based on the Item of the selected Component.

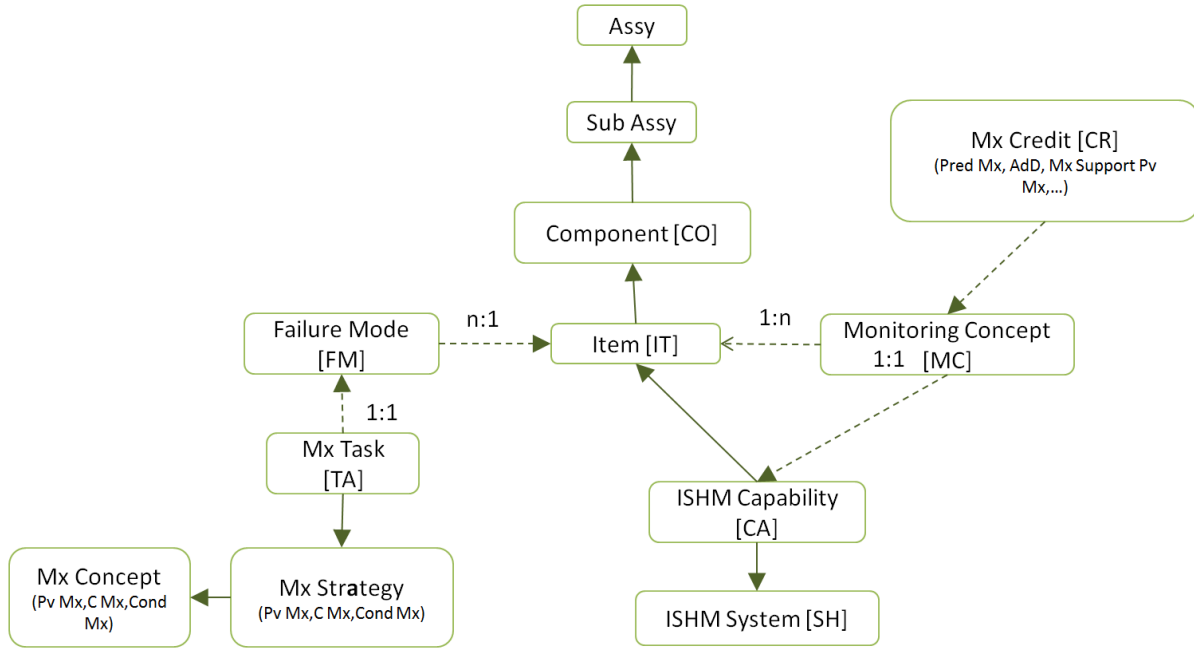


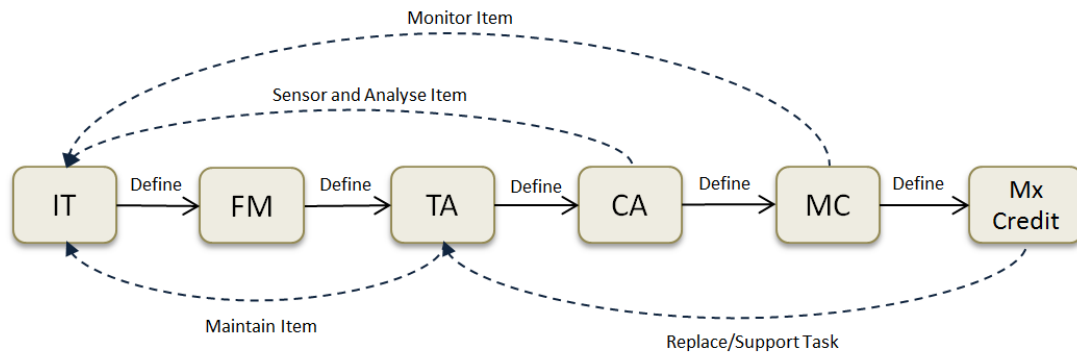
Figure 4.7: Relationship of contents to Item level [39].

Figure 4.7 shows the relationship between the Items to its different Failure Modes and Monitoring concepts. So Item is the granular part of the component where all the further contents are associated with it.

Running each process step gives a different level of decision values to Item. From Figure 4.8 we can observe that Failure mode, which is unique for the tool, is defined for the Item. So once a failure mode is defined in one process step it moves to next process step where Mx Task is defined. Moreover, it goes for ISHM capabilities, Monitoring concept, and Mx Credit.

Users have various access roles to the process steps. In addition of applying important data for items, they also add documents and links to each process step. After providing all the contents for the process step user marks the step to a locked state. So for the subsequent step the associated user can pull the information and decide about the next necessary content to be added.

Managing the content for each process step is one of the most important features of the MCF Tool. Additionally, there is a big task to cope with the change of content. Content structure is not fixed, and it might change during the time as a necessity to introduce more content for the process step. To solve this problem we have introduced a Model-Based User Interface described in section 3.1.4.



| |
|--|
| - Component --> CO |
| - Monitoring Concept --> MC |
| - Item (Part of Component) --> IT |
| -Mx Credit Type --> CR (Mx Credit , Pred Mx,...) |
| - Mx Concept --> Pred Mx, AdD Mx, Pv Mx CR (Mx Credit , Pred Mx,...) |
| - Failure Mode --> FM |
| - Mx Task --> TA |
| - ISHM System --> SH |
| - ISHM Capabilities --> CA |

Figure 4.8: Relationship of contents to Item level [39].

Chapter 5

Design and Implementation

In this chapter, the conceptual design for Mx Credit Framework that was described in Chapter 4, will be reviewed to initiate design and implementation. This chapter outlines the process of designing the MCF Tool and describes the major concepts and libraries influencing it. Before going into details of design and implementation, we would like to describe the whole procedure we have followed to develop the MCF Tool.

The goal of the Maintenance Credit Framework was to facilitate the process modeling, executing and accessing the process, adding content, annotations, data/documents links and results at runtime for each process step. Navigation of the modeled process was important. And another vital point was, not to intervene with or to modify the existent complex process and the distributed data sources.

Initially, we had the existing process model in a paper-based format. To understand in depth each process and how our MCF Tool should act, we followed a mockup driven development [10]. Web application framework can be very problematic without appropriate framework, architecture and application model. A good implementation model helps developers to communicate with the client, speed up the development and make the code highly reusable [11].

Figure 5.2 shows the general approach we followed to implement the MCF tool. Initially, we model the process flow. Out of this, we gathered the domain model and Tasks. Afterwards, we have found the basic user requirements. In order to visualize how the actual MCF Tool web application looks like, we have created all the mock ups based on tasks, that were defined in the first step. We have made about sixty mock-ups to illustrate each of the defined tasks and understand in-depth the domain model.

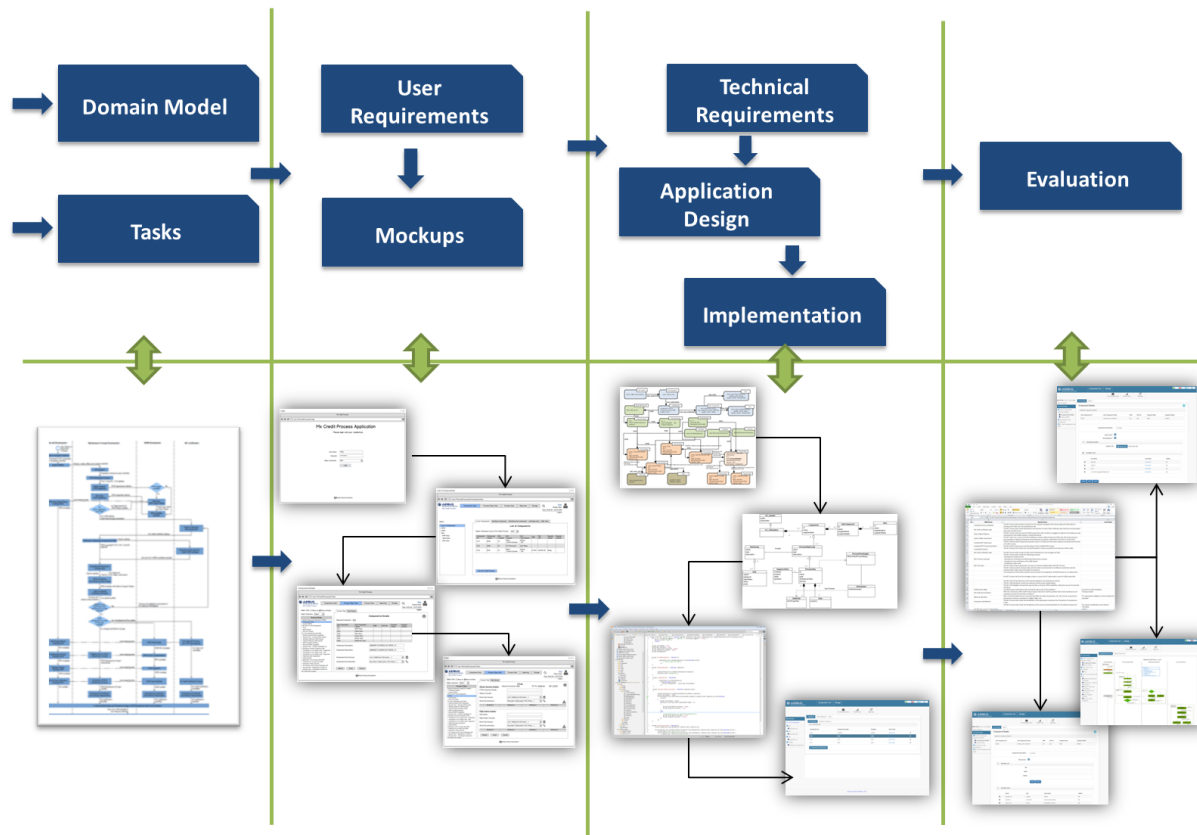


Figure 5.1: Design and Implementation approach of MCF Tool

After having a good understanding of the domain model, we have created the data model for the MCF Tool. In our case, we chose to use graph data based. So we have defined the graph data model as nodes with relationships. Based on the domain model we have made our class diagram to use that as a baseline of development. We have made MCF Tool as a single page web application based upon the concepts of Model-View-Controller(MVC). However, it is closer to the MVVM(Model-View-ViewModel) architecture. AngularJS framework was used to develop the SPA and Bootstrap was used for developing the responsive user interface.

Verification and Validation of MCF tool were one of the main important factors while developing the application. After completing each phase of development milestone, we have examined the developed features with the exemplary use case data. The representation key stakeholders were involved in this procedure. And after the developing initial phase, we have gone through a series of testing as a part of the evaluation for the MCF tool. This evaluation phase is discussed in detail in chapter 6.

5.1 Mockups for Mx Credit Framework

We have collected the mock-ups to have the precise information of the Mx Credit Framework. The Mock-ups show the basic functionality with visual details and also help to identify the problems earlier. It can be considered as a high-profile visual design draft and also adjacent to the final version of the application. We are going to describe some major mock-ups for a better understanding of the application. These screenshots of the mock-ups are only examples of the specified User Interfaces at the beginning of the design phase, which have evolved continuously during the implementation phase of the MCF tool. Figure 5.2 depicts the login page of the application where the user provides their credentials and based on the access rights of the user application provides the corresponding user interface.

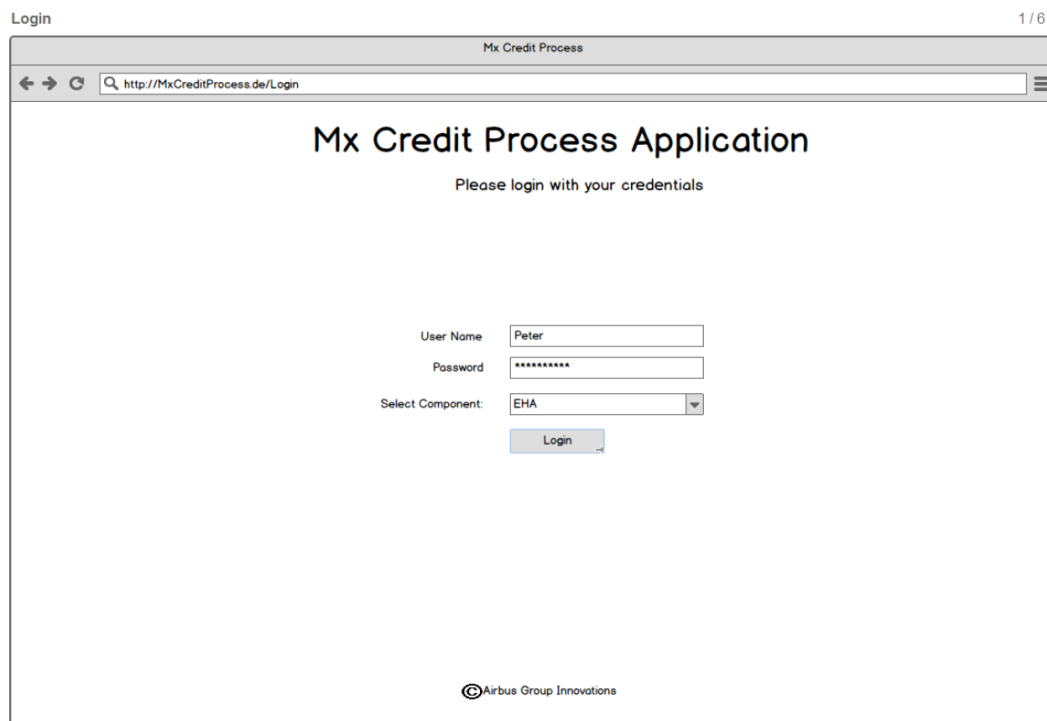


Figure 5.2: Proposed architecture for Mx Credit Application

It is also possible from the dashboard to select a component and run the Mx Credit process. Following mock-up in Figure 5.3 shows Dashboard mock-up.

The dashboard provides the important information associated with the application. It shows the main navigation panel with the list of components. It also shows the used information in the upper corner of the user interface. So from this mock-up, the user can navigate through Components view, Process Steps View, Process View, Reporting and the Manage View, if the user has the credential to manage the users and the groups. Also

from this left panel tree, it can be informed how many components the user is associated with.

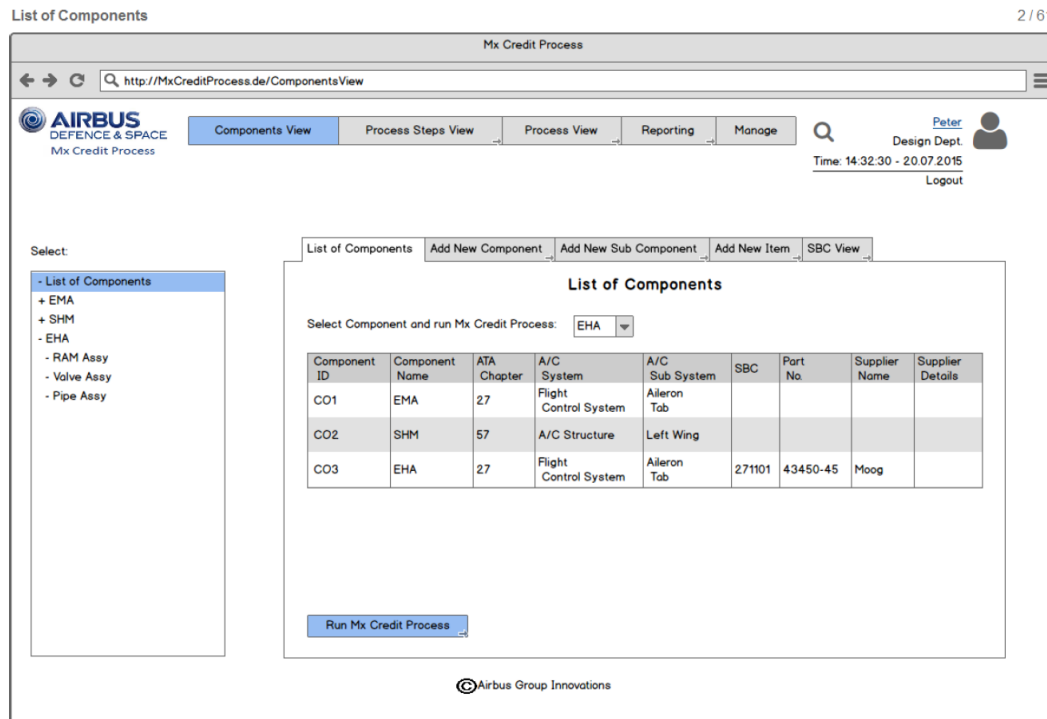


Figure 5.3: Dashboard Mock-up for Mx Credit Application

If the user selects the 'Run Mx Credit Process', application should show the functionality shown in Figure 5.4

This mock-up describes the functionality of the process step. In the left part of the user interface, there is a way to show the list of process steps that are associated with the component. Also, there is an option to select the process steps based on the association of the user from the "Select Task" checklist. So the tree, that shows the list of steps should highlight the current active step. And the right part of the user interface shows the content for the process step. For example, this "Component Details" process step has some input that should be provided by the associated user. Also there should be a way to upload files and links that are associated with the step. And also, this step should provide the component details as how many sub components are associated with this component and their details in a table view.

Component Details 22 / 61

Mx Credit Process

← → ↻ http://MxCreditProcess.de/Tasks

AIRBUS
DEFENCE & SPACE
Mx Credit Process

Components View **Process Steps View** Process View Reporting Manage

Select Task ☒ Show all ☐ Show my Tasks

Select Swimlane: Select

Process Steps

- System/Components Selection & FMECA
- Component Details**
- System FMECA
- RCM with Mx Task Development
- FTHA
- RCM Analysis
- RCM Mx Concept
- Mx Task Development with ISHM
- Identify potential ISHM Capabilities
- Decision Point - Add New ISHM System
- Add New Preliminary ISHM System
- Update Preliminary ISHM System
- ISHM Capability Definition
- Reconcile ISHM capabilities
- Decision Point - All ISHM Capabilities for Contribution to Risk & Preliminary Risk
- Contribution to A/C Safety Case - Prognostics
- Contribution to A/C Safety Case - Add
- Contribution to A/C Safety Case - Supportive
- Preliminary Risk Assessment
- Define AltMOC
- Justification of Mx Concept with ISHM
- Preliminary Mx Concept with ISHM
- Evaluation of Risk and Cost
- Mx Concept and Mx Task Justification with
- Decision Point - Redefinition of ISHM Mx
- Decision Point - Cost-Benefit and Risk for Mx Credit V&V Process

Current Task **Task History**

Components Details

Selected Component: EHA

| Sub Component ID | Sub Component Name | SBC | Part No | Supplier Name | Supplier Details |
|------------------|--------------------|-----|---------|---------------|------------------|
| SC1 | RAM Assy | | | | |
| SC2 | Valve Assy | | | | |
| SC3 | Pipe Assy | | | | |
| SC4 | Pump Assy | | | | |
| SC5 | Motor Assy | | | | |
| SC6 | Reservoir Assy | | | | |

Component Description: INBOARD FLAPERON ACTUATOR, LH

Component Data Sources: Link I (Additional Information, ...)

Component Documentation: Document I (Description: PoC, Phone, ...)

Submit Save Cancel

Airbus Group Innovations

Figure 5.4: Process Steps Mock-up for Mx Credit Application

When the user provides the information and press on submit button, it goes to the next process step based on the process model template. So next Figure 5.5 shows the second process step "FTHA".

FTHA 24 / 61

Mx Credit Process

http://MxCreditProcess.de/Tasks

AIRBUS DEFENCE & SPACE Mx Credit Process

Components View Process Steps View Process View Reporting Manage

Select Task ☐ Show all ☒ Show my Tasks

Select Swimlane: Select

Process Steps

- System/Components Selection & PHECA
- Component Details
- System PHECA
- RCM with Mx Task Development
- FTHA**
- RCM Analysis
- RCM Mx Concept
- Mx Task Development with ISHM
- Identify potential ISHM Capabilities
- Decision Point - Add New ISHM System
- Add New Preliminary ISHM System
- Update Preliminary ISHM System
- ISHM Capability Definition
- Reconcile ISHM capabilities
- Decision Point - All ISHM Capabilities for Contribution to Risk & Preliminary Risk
- Contribution to A/C Safety Case - Prognostics
- Contribution to A/C Safety Case - Add
- Contribution to A/C Safety Case - Supportive
- Preliminary Risk Assessment
- Define AIMOC
- Justification of Mx Concept with ISHM
- Preliminary Mx Concept with ISHM
- Evaluation of Risk and Cost
- Mx Concept and Mx Task Justification with
- Decision Point - Redefinition of ISHM Mx
- Decision Point - Cost-Benefit and Risk for Mx Credit V&V Process

Current Task Task History

FTHA

Mission Success Analysis Selected Component: EHA Part No: 43450-45 SBC: 2711011

FTHA Component Details

Mission Criticality:

Enter Data Sources: Link I (Additional Information, ...)

Attach Documentation: Document I (Description: PoC, Phone, ...)

| Attribute 1 | Attribute 2 | Attribute 3 | Attribute 4 |
|-------------|-------------|-------------|-------------|
| | | | |

Flight Safety Analysis

FSA Details

Flight Safety Criticality:

Enter Data Sources: Link I (Additional Information, ...)

Attach Documentation: Document I (Description: PoC, Phone, ...)

| Attribute 1 | Attribute 2 | Attribute 3 | Attribute 4 |
|-------------|-------------|-------------|-------------|
| | | | |

Submit Save Cancel

Airbus Group Innovations

Figure 5.5: Process Steps Mock-up for Mx Credit Application

This mock-up shows the necessary input fields and process navigation tree on the left. Also users should be able to navigate between process steps by clicking on the left part of the process navigation tree. Also it is possible to know the status of the process steps by clicking on the process view. Figure 5.6 shows the graphical process view. Graphical process view provides the information of the current status of the process steps for a component.

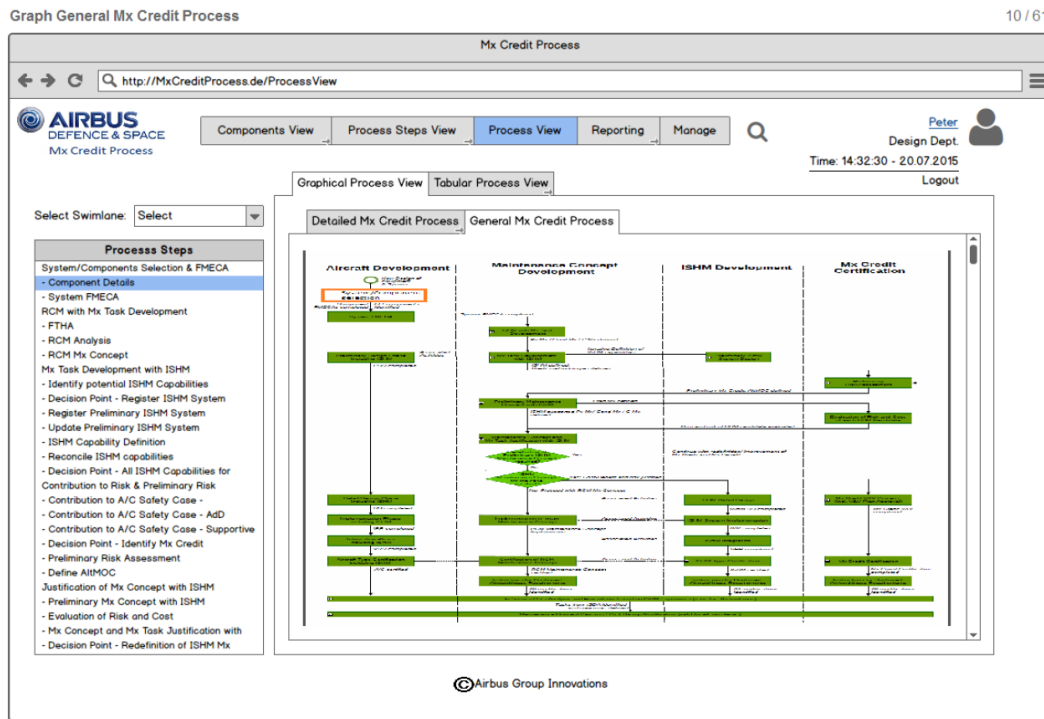


Figure 5.6: Process view Mock-up for Mx Credit Application

5.2 Mx Credit Application Architecture

This chapter explains the overall Mx Credit Application architecture with a deployment diagram shown in Figure 5.7. If we take a look at the proposed architecture in section 4.5.1, we can see, that the MCF Tool contains three main components: User Interface, Application server and Database Server. Deployment diagram provides a more detailed view by showing the internal components located under each main component.

Below, a list of important components of MCF Tool is shown:

- **Process Modeler:** This module creates the template of the process flow with rules.
- **Model-Based UI:** This is one of the main components of the Mx Credit Framework. Based on the JSON schema it generates the User interface for the process steps. It collects the JSON schema from the process steps and generates the process graphical user interface.
- **Process Flow Engine:** Process Flow Engine creates the instance of process flow created by the Process Modeler component and assign it to a certain component. This component also controls the transition of process flow based on rules.

- Neo4js API: As we have used graph database to store the Mx Credit Framework data. It provides the API to communicate with the database using the REST call.

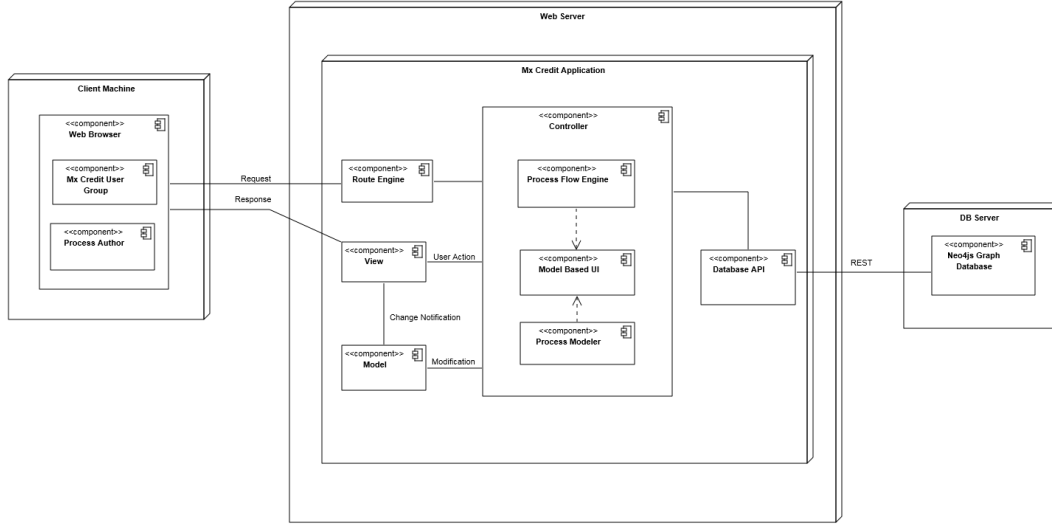


Figure 5.7: Deployment diagram of Mx Credit Application

To give the overview of the implemented system, section 5.2.1 describes the graph data model of the MCF Tool. Section 5.3 describes the process modeler and section 5.4 provides the implementation details of Model-Based user interface.

5.2.1 Graph Data model for MCF Tool

We have developed a graph model to navigate the process steps as shown in fig 5.8. Graph data modeling is the process to describe an arbitrary domain as a connected graph of nodes and relationships. Every node represents entities, and relationship connects nodes to nodes [37].

Both nodes and relationships are containers of properties, which are effectively a name/value pair. To implement the graph data model, we have used Neo4j database. We can find more details of Neo4j in section 3.3.5. In section 4.5.3 we discussed the hierarchy of System, Sub System, Component, Sub Component, Item. In our graph model, each object acts as a node. Furthermore, we have made the relationships between the nodes. For instance, A/C SubSystem node obtains a HAS_COMPONENT relationship with Component node.

To introduce the process steps, we have added a parent node *Mx Credit Process*. Component node includes a HAS relationship with Mx Credit Process node. Furthermore, Mx Credit Process node consists of Process Group Node. And Process Group node

contains Process Step nodes. Each Process contains its properties and relationship to other process nodes. The type property of Process Step node defines the category of process step. There are two types of categories, "userTask" and "exclusiveGateWay". Process Step having "exclusiveGateWay" property is responsible for managing rule based process flow.

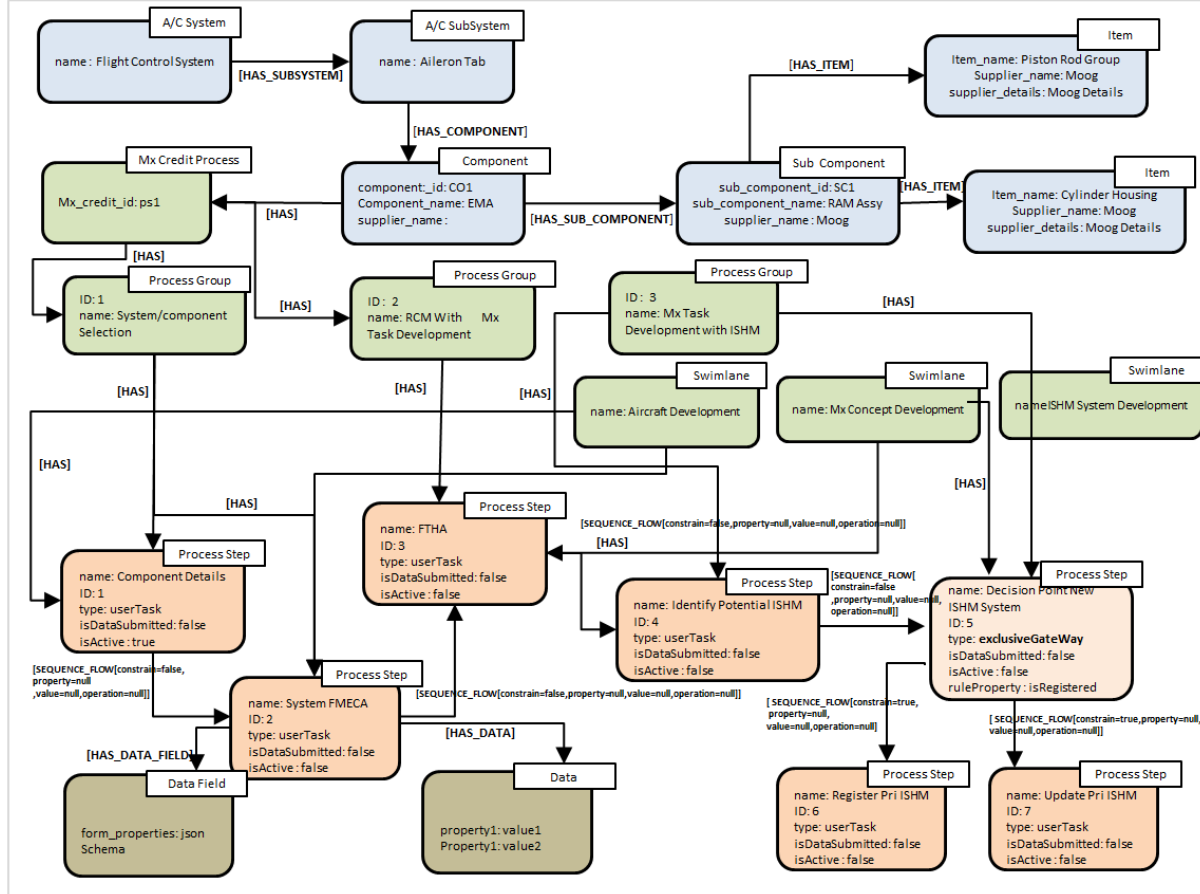


Figure 5.8: graph For Mx Credit Application

Also Process Step node is associated with two nodes "Data Field" and "Data". Data Field node stores the JSON schema in its **form_properties** property and user submitted values for the process step stores in "Data node".

5.2.2 RESTful API

MCF Tool uses RESTful api for all its data communication stored on the neo4j database. Over HTTP, different clients can work on the same data. Basic description of REST can be found in section 3.3.6. As mentioned there, the data sent to the client is up to the

server, and JSON is used to be the standard representation of data. Description of JSON can be found in section 3.3.4. As mentioned above the protocol HTTP is used to handle the client-server communication. Methods GET, PUT, POST and delete match with the operations create, read, update and delete. The general behavior of the API concerning collections, elements and each supported HTTP method can be seen in table 5.1

| Method | Description |
|--------|--|
| GET | Used to retrieve resources and their status or description or more other information |
| POST | Used to create resources, or performing custom actions |
| PUT | Used to update or insert resources |
| DELETE | Used to delete resources |

Table 5.1: HTTP Verbs available in MCF Tool API

All API are defined to communicate with Neo4j api. We have used neo4j-js ¹ library to use the Neo4j API's. Many elements of MCF Tool have been mapped to collections. To organize the business capabilities, single responsibility and high cohesion, we determine the proper number of services. Shown in 5.9 these services give the gate way to manage content of Mx credit process.

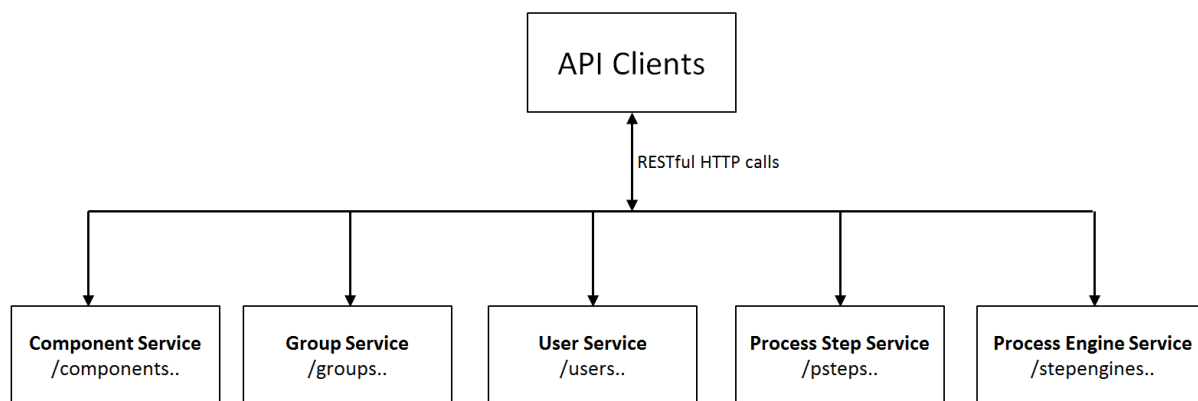


Figure 5.9: MCF Tool REST API services

Every service in the MCF Tool is responsible for creating cypher queries for the Neo4j database. In order to add a content to database, a service has to create a cypher query and to call corresponding API to pass this query. After receiving the request with a query, the API connects with Neo4j database with Node.js client library *neo4j-js*. When connecting

¹<https://github.com/bretcope/neo4j-js>

to the database server, the authentication values have to be given. After connecting to the database server, neo4j-js library passes the query. And neo4js database executes the query. In that part there is also some error handling mechanism used. So if any error occurs during execution of the query to the database, it returns error details. And if there is no error, then the results are returned in JSON format.

```
1   var neo4j = require('neo4j-js');
2   var neo4JUrl = 'http://localhost:7474/db/data/';
3
4   router.post('/', function(req, res) {
5
6     neo4j.connect(neo4JUrl, function(err, graph) {
7
8       var query = req.body.query;
9
10      graph.query(query, null, function(error, results) {
11        if (error) {
12          error = error;
13        } else {
14          retValue = results;
15        }
16        res.json({
17          responseData : retValue,
18          error : error });
19      });
20    });
21  });
22  }
```

Figure 5.10: Example of neo4j-js connection to Neo4j database

Figure 5.10 shows connection details code. In this case, neo4jUrl contains, the database server URL and neo4js-js library use this connection string. A sequence diagram in Figure 5.11 is introduced to explain the data flow more clearly. As from the diagram we can observe that application user sends a HTTP POST call. In this example the user wants to get all the components that are available in MCF Tool. The request goes to *Routes* and it redirects to corresponding *ComponentView*. And then interacts with the *Component Controller*. *Component Controller* calls *Service Factory*, in this case *ComponentService*. *ComponentService* creates the Neo4j cypher to retrieve the component list from the database. And it calls the neo4j-js library API with the HTTP call and submits the cypher.

The *RESTful API* connects with the Neo4j database server and passes the query to

execute. Database executes the query and returns the query result in a response. RESTful API receives the response and creates a JSON format component list as a return value and passes it to Service Factory. Controller receives the list from Service Factory and binds it with \$scope and updates view.

5.3 Implementation of Process Modeller

This section provides an overview of the implementation of the Mx Credit Application and gives a detail inside the implementation of the process steps modeller. To implement the described architecture in section 5.2, many technology stacks have been included. We have incorporated various different libraries on the individual layers.

In order to get the object model of the system, we have created a class diagram. Figure 5.12 shows the class diagram for the MCF Tool. Class diagrams play a key role in the analysis and design of information systems especially in development contexts that use modeling oriented methodologies [35].

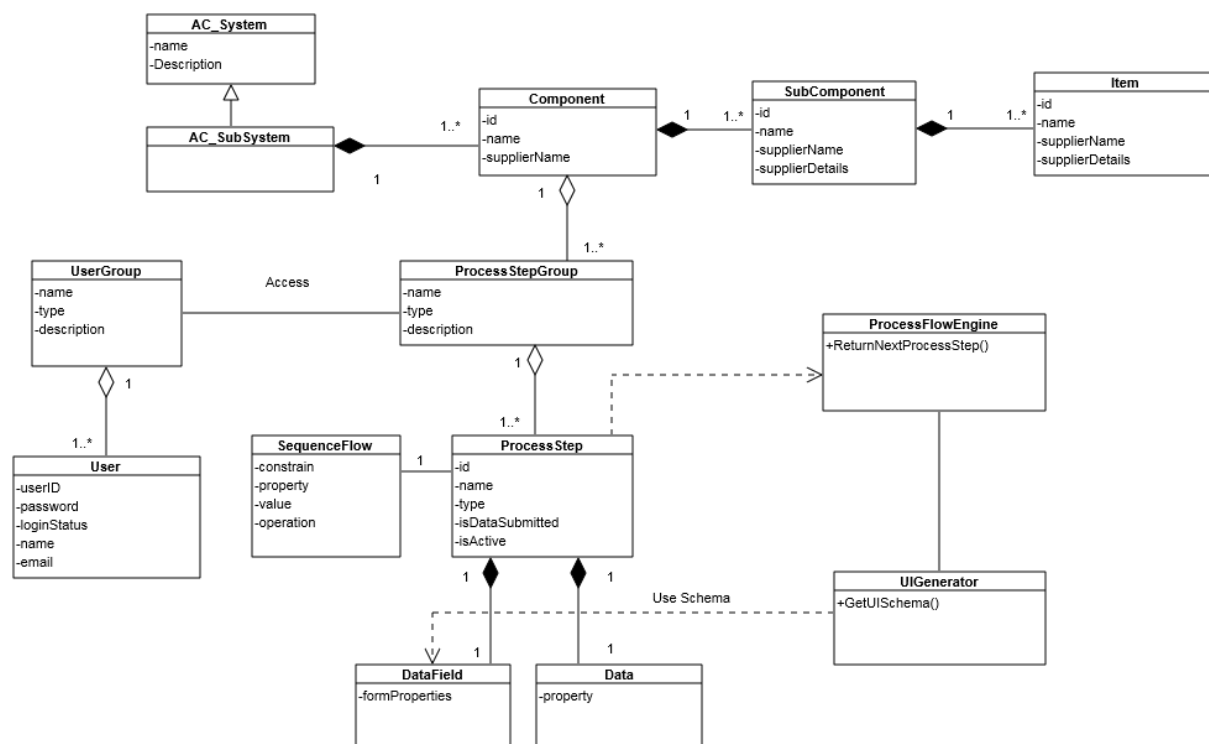


Figure 5.12: Class diagram of Mx Credit Application

5.3.1 Single Page Application Implementation

We have implemented the MCF Tool as single page application(SPA). A single-page application is a website that retrieves all needed components in one single page load. The main purpose is to get a user experience that gives an impression of using a native application rather than a website. Furthermore, a Single Page application can be defined as " The single-page web interface is composed of individual components, which can be updated/replaced independently, so that the entire page does not need to be reloaded on each user action " [7]. From this definition, we can find out most important attributes of SPA:

- Web interface: Used on the web, focuses on user interfaces.
- Singular components: It divides into smaller components.
- Reloading: Full page is never reloaded, new contents are loaded into some sections.
- Asynchronous communication: It communicates asynchronously with the server to load only specific parts and it allows server load only specific parts of the page

In order to implement SPA, client side has to contain more logic. Use of JavaScript libraries like AngularJS provides the support for a logic-driven front end. Figure 5.13 shows at a glance the architecture of SPA implementation of MCF Tool.

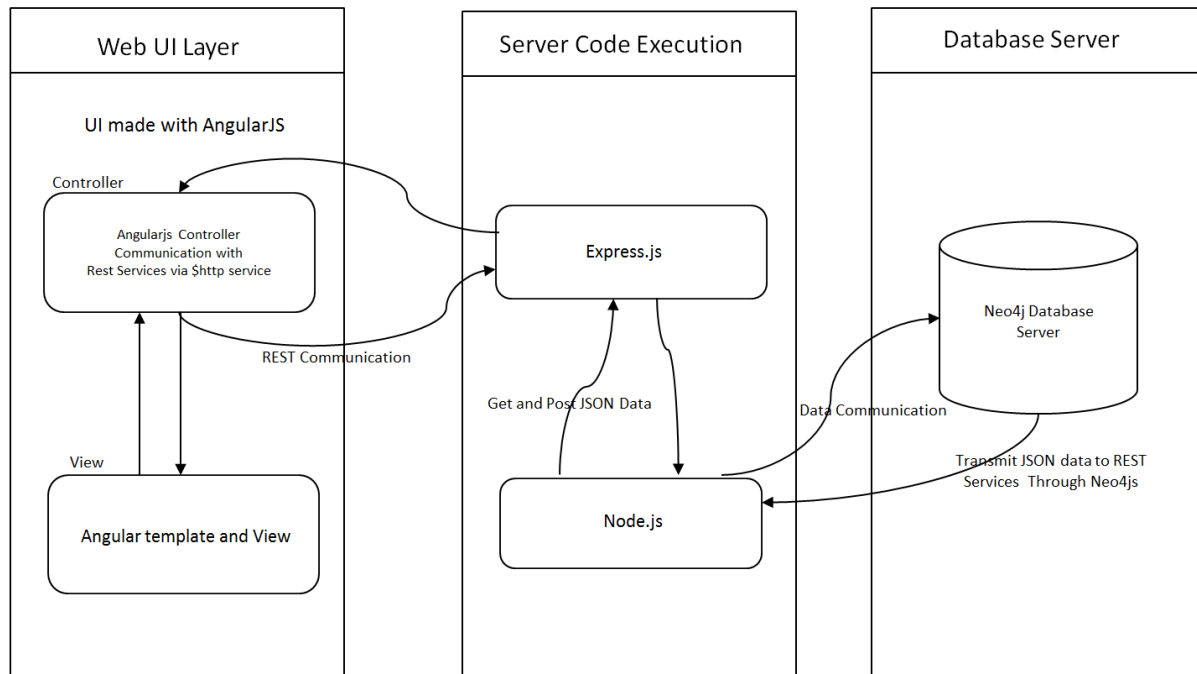


Figure 5.13: Single page architecture of MCF Tool

Process modeler creates the process steps flow template. As described in the class diagram in Figure 5.12, each component will have process groups. Each process step group will have process steps. The transition between process steps is made by certain rules.

To implement the structure based upon the class diagram, we followed the Model-View-Controller(MVC) pattern [14]. We developed the MCF Tool web application to the context of AngularJS application. While building the web application we needed

- Model that serves the state of the application.
- View to Display data.
- Controller that controls the relation between Models and Views.

From Figure 5.14, we can understand how these three components work in AngularJS MVC Architecture.

Model

In AngularJS, a model is a plain JavaScript object. A model can consist of all primitive data type such as integer, string and boolean. No getter or setter method is in plain JavaScript method.

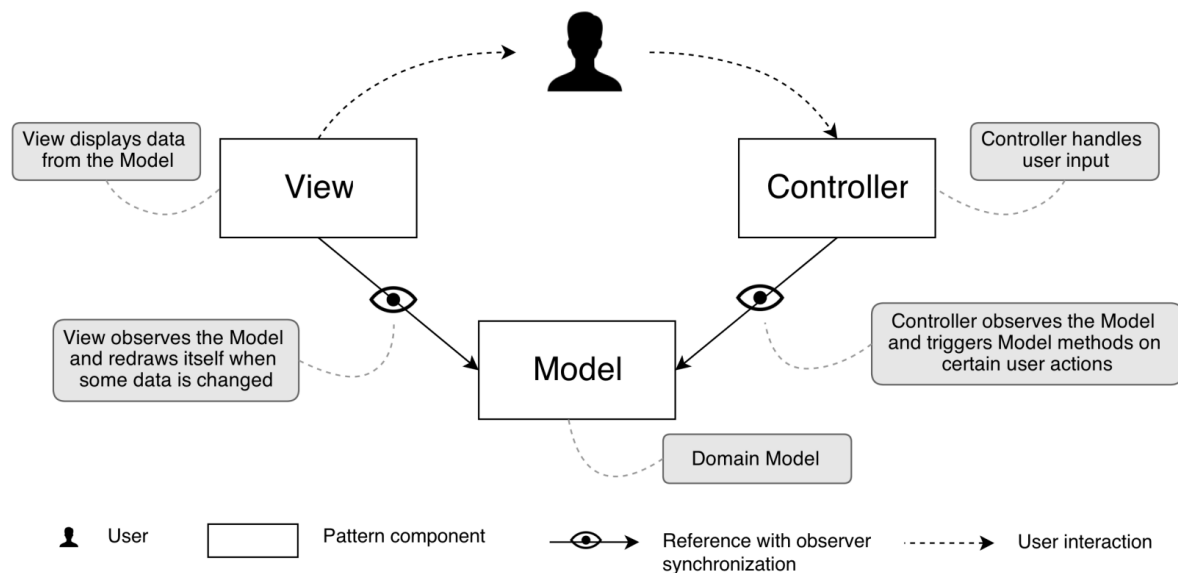


Figure 5.14: The Model View Controller Software Pattern as described by Syromiatnikov and Weyns [15].

View

The View is the DOM element which is used to display data. AngularJS supports two-way data binding. So, any changes in model data will be automatically updating the view.

Controller

All the application logic located in the controller as a concept of MVC pattern. In AngularJS, the controller is formed by JavaScript classes. It proves control over view and model in order to fetch the data per request and display it in the view.

So coming back to the creation of process steps and apply the rule-based navigation, we have implemented the AngularJS MVC pattern based web application. When creating a process template, these steps are followed :

1. Create the process step.
2. Set the process step type. Type can be UserTask or GateWay.
3. Set the sequence flow properties. If the gateway is of type Process Steps, the constrain is true, set the property, value and the operation.
4. Set the Data Filed for the process Step.

This data field generation is done by the Model-Based UI component. We will discuss this component in detail in next section 5.4. Once the template is being generated, it is attached with the component. So each component will have an instance of the template. So the flow of the process steps will be executed as defined in the template.

5.4 Model Based UI

In order to handle the complex process environment and its contents, we have used a Model-Based UI development procedure. As discussed in section 3.1.4, the main idea of model-based approaches for User Interface(UI) development is to deal with the complexity of interactive applications. Model-based UI gives a way to meet the UI changes efficiently. In our MCF Tool, we have introduced this model-based UI approach. As derived from the Model-Based User Interface Development (MBUID) methodology we first find out the main tasks and then transform it to an abstract user interface. And then we have applied JSON schema generate the concrete user interface. And after that getting the concrete UI we have made some rendering mechanism to obtain the Final User interface. Figure 5.15 gives the overview of the procedure.

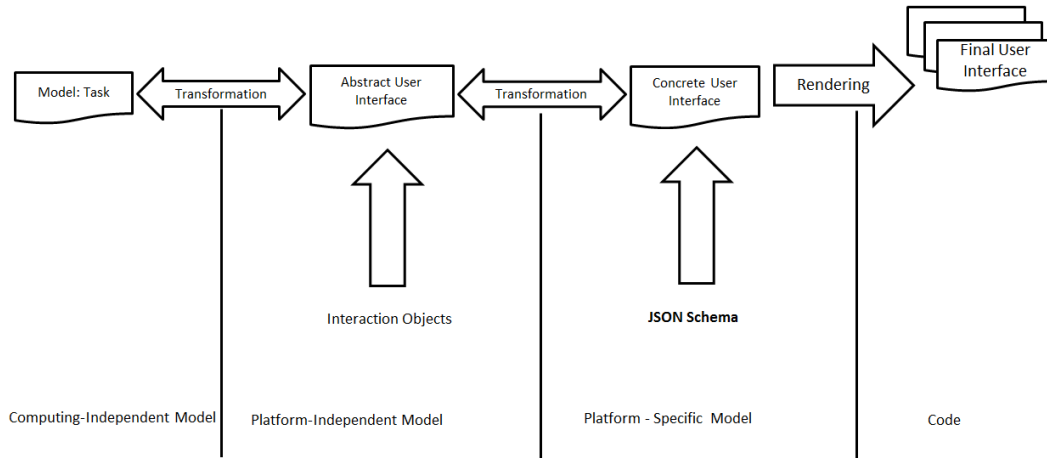


Figure 5.15: Model based Approach for UI development based on JSON [5]

In MCF Tool, we have used JSON schema for setting the outline of the process UI and stored it in the Neo4j graph database. We also created a custom directive of AngularJS to render code from the stored JSON schema. There are two main focus points in MCF Tool for model-based UI implementation. First, the author creates JSON schema and stores it according to the process steps. And second, the users who run the process steps, receive the user interface as generated HTML code from the JSON schema. In Section 5.4.1 and 5.4.2, we described details of this two main implementation area.

5.4.1 Process Schema Generation

According to the MCF Tool, the admin user in our case is the author responsible for creating the JSON schema for the process steps. We can see from the schema listed in 5.1 for Component Details process step. Author has to create each web element as a JSON array for each process step. So for creating a textbox the value in the key `field_type` is "text field". Also, there are other options like disabling the field, setting the title. Initial values for the field are available.

Listing 5.1: JSON Schema for Component Details Process Step

```

1  {
2    "1" : {
3      "form_id" : "1",
4      "form_name" : "Component Details Form",

```

```
5      "form_fields" : {
6      "1" : {
7      "field_id" : 1,
8      "field_title" : "Component Name",
9      "field_type" : "textfield",
10     "field_value" : " ",
11     "field_required" : true,
12     "field_disabled" : false
13   },
14   "2" : {
15     "field_id" : 2,
16     "field_title" : "Component Details",
17     "field_type" : "textfield",
18     "field_value" : " ",
19     "field_required" : true,
20     "field_disabled" : false
21   },
22   "3" : {
23     "field_id" : 3,
24     "field_title" : "Mitigation required",
25     "field_type" : "radio",
26     "field_value" : "2",
27     "field_required" : false,
28     "field_disabled" : false,
29     "field_options" : [
30     {
31       "option_id" : 1,
32       "option_title" : "Yes",
33       "option_value" : 1
34     },
35     {
36       "option_id" : 2,
37       "option_title" : "No",
38       "option_value" : 2
39     }
40   ],
41   },
42   "4": {
43     "field_id": 4,
44     "field_title": "Line Break",
45     "field_type": "linebreak",
46     "field_value": " ",
47     "field_required": true,
48     "field_disabled": false
49   }
50 }
```

```

51     }
52 }

```

After creating the schema, author has to store this according to the specific process step. As discussed in section 5.2.1, we have used Neo4j to store the schema. We have introduced a template-based process step modeling. So each process step should have a JSON schema defined and when the user runs Mx Credit Process, every process step takes the template schema and stores it to the "DataField" node.

As shown in Figure 5.16, Author selects the Manage UI section and gives the JSON schema for Component Details process step. From the implementation perspective, when author saves the schema, *MbuiCtrl* calls "Updateschema" function of *MbuiService* and gives the schema and step number as an input.

MbuiService receives the parameters and creates a query from that. It also calls the ProcessSchema API using HTTP POST and passes the query to Neo4j database through RESTful API. As this schema is saved as a temple for the process step, it also provides options to modify it. However, one important issue is when Mx Credit Process is running for one Component, the complete template schema is loaded to this particular Component. And the data that is inserted by the user is saved for this specific component. So for this case, if author modifies the schema and adds a new "text field" the new text field will be available to the next created Component of MCF Tool. In order to have a good data-management procedure, this version concept of the schema is introduced.

```

{
  "1": {
    "form_id": "1",
    "form_name": "Component Details Form",
    "form_fields": {
      "1": {
        "field_id": 1,
        "field_title": "Component Name",
        "field_type": "textfield",
        "field_value": "",
        "field_required": true,
        "field_disabled": false
      },
      "2": {
        "field_id": 2,
        "field_title": "Component Details",
        "field_type": "textfield",
        "field_value": "",
        "field_required": true,
        "field_disabled": false
      },
      "3": {
        "field_id": 3,
        "field_title": "Mitigation required",
        "field_type": "radio",
        "field_value": "2",
        "field_required": false,
        "field_disabled": false,
        "field_options": [
          { "option_id": 1

```

Figure 5.16: Including JSON schema for Component Details process step

5.4.2 User Interface Generation from JSON Schema

When users of MCF Tool run the Mx Credit Process, they are redirected to their associated process steps. Process steps controller loads all necessary details of the process step. For this instance, *ComponentDetailsCtrl* loads the available data and shows it on ComponentDetails Step View. For loading the JSON schema associated with this process step it calls *GetCurrentProcessStepDetails* function of *ProcessEngineService* and receives the JSON schema. It binds this JSON schema to `$scope.form`.

This "form" property converts JSON schema to JSON object list. As we can find out in 5.2, **form-directive** is added in the HTML of process steps. It receives the field object from all the `form_fields`. That is the list of JSON objects of the schema.

Listing 5.2: form-directive in Process steps HTML

```

1 <div class="form-group">
2   <div class="col-lg-12">
3     <form-directive form="form"></form-directive>
4   </div>
5 </div>
```

This **form-directive**, loads `form.html` by using `templateUrl: './views/directive-templates/form/form.html'` of the directive. Inside the `form.html`, there is another **field-directive** as we can see from Listing 5.3

Listing 5.3: field-directive in form.html

```

1 <form name="myForm" id="myForm">
2   <div ng-repeat="field in form.form_fields">
3     <field-directive field="field">
4
5     </field-directive>
6   </div>
7 </form>
```

Using the "ng-repeat" of AngularJS all the field objects are passed to the field-directive, and it generates the HTML code for every field Object. The JSON, schema listed in 5.1 for Component Details process step looks like Figure 5.17. From the JSON schema we can find out there were two text-boxes, one radio button and one line break component. As it passed to the **form-directive** of the Component Details process step View, four form fields have been generated.



Component Name:

Component Details:

Mitigation required: ☐ Yes ☒ No

Figure 5.17: Including JSON schema for Component Details process step

5.5 Process Flow Engine

One of the key aspects of MCF Tool is the proper navigation of process steps flow. As shown in Figure 5.18, we have used a communication diagram to describe the process flow engine. Users of the MCF Tool, in this case a client, select the Components from the List of Component in Component Dashboard View. Then View communicates with ProcessFlowEngine with the Component id. ProcessFlowEngine calls ProcessStepService to get the process step list. ProcessStepService calls the componentApi to retrieve the list of process steps that are available for the respective component.

As ProcessFlowEngine receives the steps, it communicates with the ModelBasedUI object to get the JSON schema for the process step. ModelBasedUI generates the UI and calls for UI update to show the process step UI.

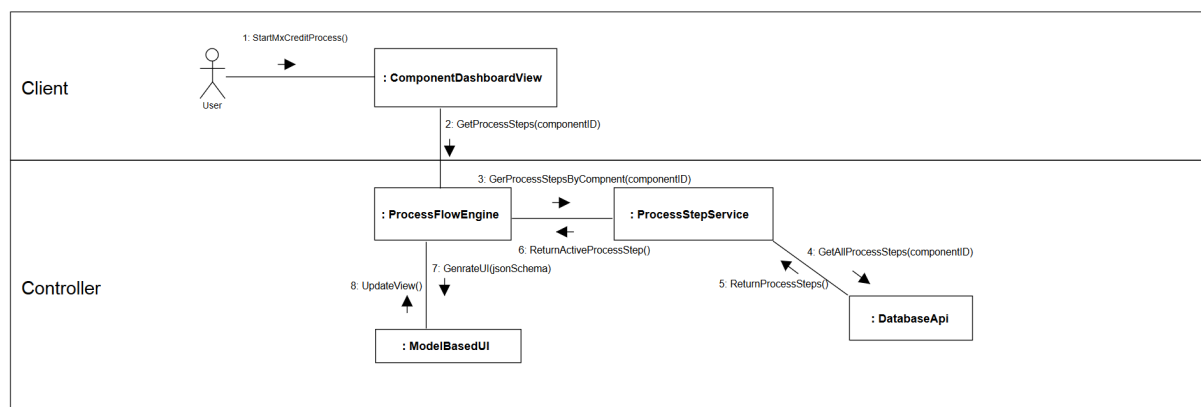


Figure 5.18: Communication diagram for Process Flow Engine

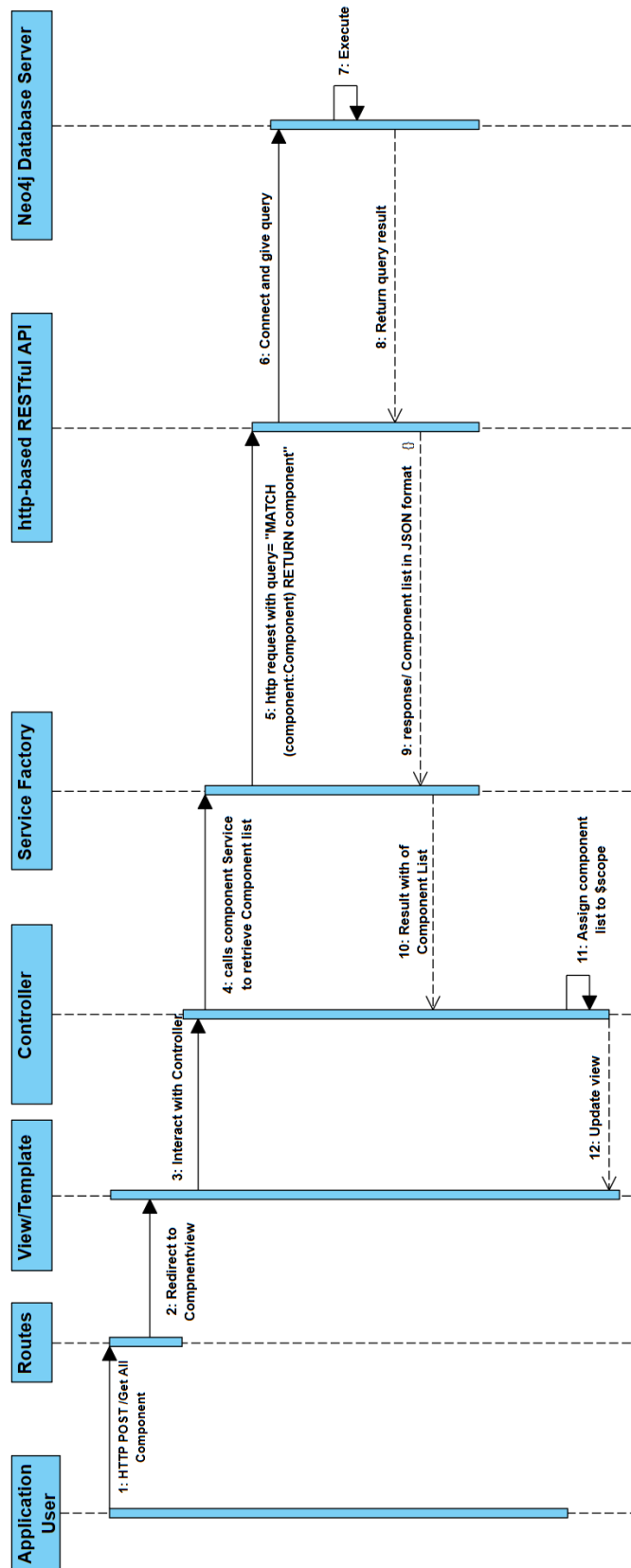


Figure 5.11: REST API Request/Response Data Flow

Chapter 6

Evaluation of the MCF Tool

The main goal of the evaluation is to verify and validate in a first step the main tool features. The MCF tool development will continue in a near future. The tool implemented until now can be seen as a baseline for further developments.

There are four main elements to be evaluated for now in the MCF tool:

- The GUI design
- The Graph Data Model
- The Mx Credit Process execution
- The authoring process for the model-based creation of the user interface

As the Mx Credit Process model is not yet fully revised, the evaluation of this part has been excluded. For the evaluation of the MCF tool a few test persons, as representative users, have been selected to use the tool and give feedback based on a questions-and-answers exercise. This short user acceptance tests should lead to the identification of possible missing or faulty features.

Besides this, a proof-of-concept regarding the defined Mx Credit Process approach can be carried out. This includes finding out the quality of the user guidance along the process, the ability of the users to include and extent data sets with the provided features, and to follow the process status, as well as to report it. The tool evaluation criteria is taken from the collected requirements described in section 2.2. The following criteria, respectively MCF features, have been selected for evaluation:

- Condition-based user guidance along the executed Mx Credit Process
- Collection and easy access to all relevant data during the execution of the Mx Credit Process

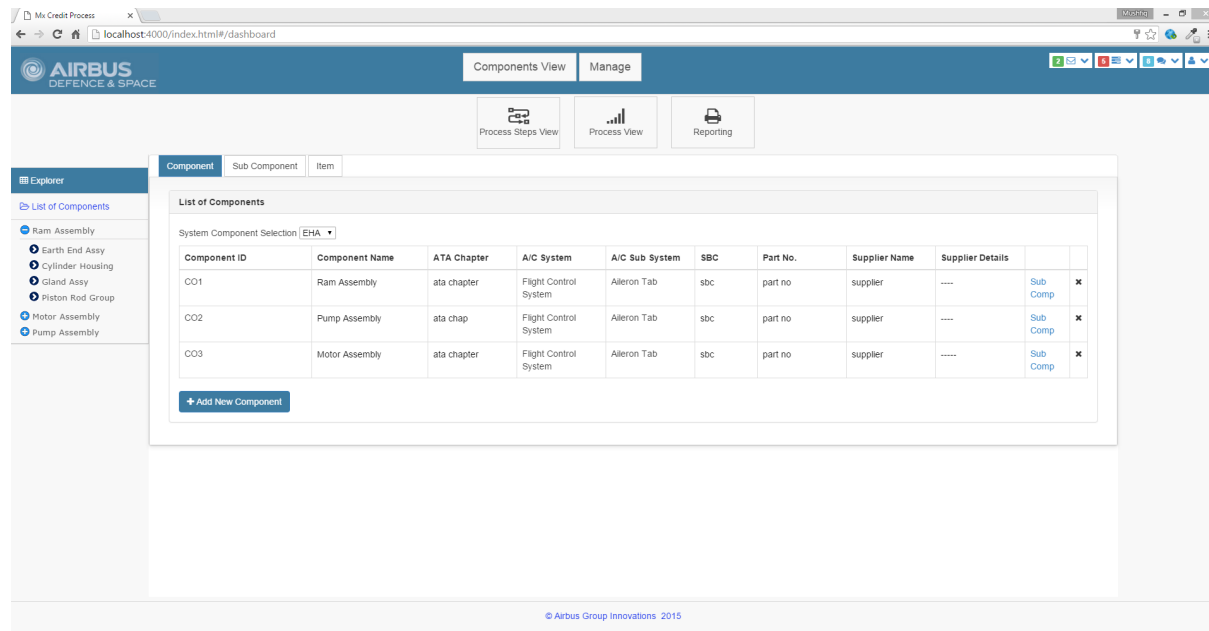


Figure 6.1: MCF web application user interface.

- Adding of content, annotations, data/documents links and results at runtime for each process step
- Traceability and Tracking of all relevant information
- Accessibility to the description of the modelled Mx Credit Process
- Accessibility to complete Mx Credit Process history, including complete generated results per process step.
- Mx Credit Process status view at any time through a graphical representation (graphical process view)

Based on a representative use case and these selected criteria user acceptance tests for the MCF tool have been carried out.

6.1 Definition of a representative Use Case

The evaluation realized for the MCF tool is based on one exemplary use case. The use case describes the selection of an aircraft component as a candidate to be monitored by advanced diagnostics and prognostics, so-called CBM candidate. The component runs through all the steps of the described Mx Credit Process. Here, the selected component is being represented by an Electro-Hydrostatic Actuator (EHA).

Select Task ☐ All ☐ My Tasks
 Select Swimlane
 Aircraft Development
 Process Steps

Current Step History

Process Steps View Process View Reporting

Component Details

Selected Component Ram Assembly

| Sub Component ID | Sub Component Name | Items |
|------------------|--------------------|--------------------|
| SC1 | Cylinder Housing | 1 Static Seals (4) |
| | | 2 Static Seals (3) |
| SC2 | Gland Assy | 1 Static Seals (2) |
| | | 2 Static Seals (1) |

Data Source : ☐

Documentation : ☐

Submit Save Cancel

Figure 6.2: Mx Credit Process Description

As the modelled Mx Credit Process is extensive, only the first ten steps of the Mx Credit Process have been selected for this evaluation (see figure 6.2). As not all the required EHA data could be gathered from the key stakeholders, only a few representative data sets of the EHA component have been used for evaluation. A short storyboard led by the defined Mx Credit Process has been created. The storyboard describes the use case with all the necessary actions to be carried out by the test users along the selected Mx Credit Process section.

6.2 Short EHA Description

The Electro-Hydrostatic Actuator (EHA) is an electrically driven, self-contained hydraulic actuator developed for aileron actuation of the Airbus A321. It was developed and flight tested within the European research and development program "EPICA" (1992-1996). The EHA is an actuator which contains an internal hydraulic supply including a pump and a reservoir. An electrical motor drives the constant displacement pump providing the hydraulic flow to the ram. Direction of ram travel is given by the direction of pump rotation. This means a positive speed command results in positive rotational direction of drive and pump. A positive pump rotation direction supplies pressure to the extend port of the ram. A negative pump rotation supplies pressure to the retract port of the ram. The pump is supplied by an internal hydraulic reservoir. Different sensors are available to monitor the system status of the EHA.

6.3 Considered Failure Modes and Maintenance Concepts for EHA

Based on two selected failure modes of the EHA, two RCM maintenance concepts have been defined. Based on the defined maintenance concepts, including the identified contribution to risk and criticality levels, two monitoring concepts have been exemplary implemented to demonstrate the use of the MCF tool. The selected failure modes, defined RCM maintenance concepts and the monitoring concept are described in table B.1, B.2, B.3 and B.4.

6.3.1 Visual Inspection

During the flight-line or pre-flight check a visual inspection of the control surface connection shall be carried out. Additionally, a spot check with respect to leakage shall be done. The EHA features an optical filling indicator. Therefore the fluid level has to be checked during every pre-flight inspection.

6.3.2 Leakage Check

A leakage check shall be carried out to check static and dynamic seals. The leakage check shall be performed during a maintenance interval every 500 FH. Initial Condition:

- EHA Controller operative
- Electrical power available
- Bypass Valve OPEN

6.4 User Acceptance Tests

The user acceptance tests followed two objectives, on the one hand to verify the features provided by the MCF tool and, on the other hand, to validate the quality of the same by getting feedback from the test users. The EHA use case data was mainly provided by Excel sheets. No connection to databases has been implemented, although the MCF tool offers such functionality. The users were asked to use the MCF tool and to make their correspondent entries according to the defined storyboard. According to the EHA use case scenario the main user actions along the described Mx Credit Process steps were:

1. Provide all details of the EHA component, selected as CBM candidate.

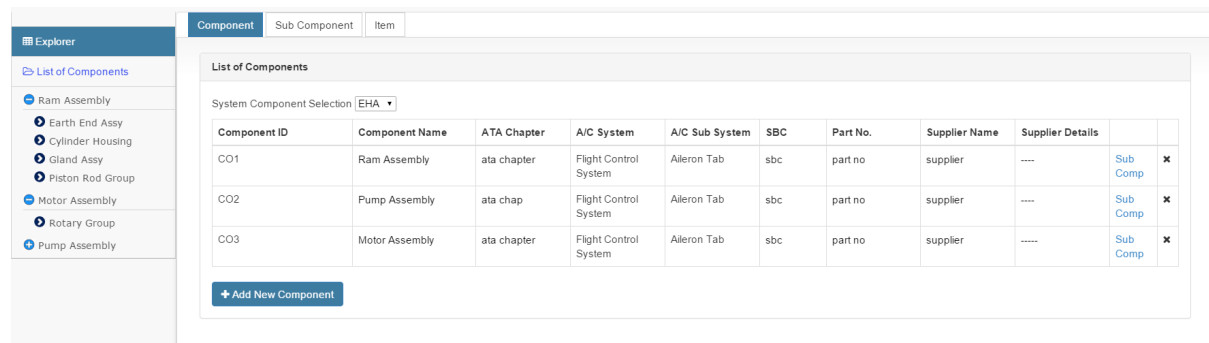
2. Provide all necessary FMEA and FMECA data as part of the System Design Assessment
 - (a) Fill correspondent FMEA/FMECA text fields and tables according the templates provided in the MCF tool.
 - (b) Make reference to all available and relevant documentation and existent data sources related to the performed analyses
3. Provide the results of the developed RCM maintenance concept
4. Provide all relevant information about the Preliminary Mx Task Development with ISHM
5. Provide all relevant information about Preliminary ISHM System Design

6.5 Evaluation results

The outcome of the tests concluded into some modifications of the graph database model and a few adaptations of functionalities on the GUI side. Based on the selected criteria for the MCF tool evaluation the following conclusions described in the sections 6.5.1- 6.5.8 have been established.

6.5.1 Component View

The starting point for the Web UI evaluation was the “Components View”. Based on the EHA use case specific content has been entered to build the hierarchical components data model.



The screenshot displays the 'Component View' interface. On the left is a sidebar with an 'Explorer' menu containing a tree of components: Ram Assembly, Earth End Assy, Cylinder Housing, Gland Assy, Piston Rod Group, Motor Assembly (selected), Rotary Group, and Pump Assembly. The main area has tabs for 'Component', 'Sub Component', and 'Item'. Below the tabs is a 'List of Components' section with a 'System Component Selection' dropdown set to 'EHA'. A table lists three components: CO1 (Ram Assembly), CO2 (Pump Assembly), and CO3 (Motor Assembly). Each row includes columns for Component ID, Component Name, ATA Chapter, A/C System, A/C Sub System, SBC, Part No., Supplier Name, Supplier Details, and a 'Sub Comp' link with a delete icon. An 'Add New Component' button is at the bottom.

| Component ID | Component Name | ATA Chapter | A/C System | A/C Sub System | SBC | Part No. | Supplier Name | Supplier Details | |
|--------------|----------------|-------------|-----------------------|----------------|-----|----------|---------------|------------------|------------|
| CO1 | Ram Assembly | ata chapter | Flight Control System | Aileron Tab | sbc | part no | supplier | ---- | Sub Comp ✕ |
| CO2 | Pump Assembly | ata chap | Flight Control System | Aileron Tab | sbc | part no | supplier | ---- | Sub Comp ✕ |
| CO3 | Motor Assembly | ata chapter | Flight Control System | Aileron Tab | sbc | part no | supplier | ---- | Sub Comp ✕ |

Figure 6.3: Component View with list of components with details

Slightly modifications on the UI layout have been done. Besides this, all available functionalities were fully operational.

6.5.2 Manage Group & User View

The Manage User groups and users have been tested. Three users have been registered accordingly and assigned to available groups, representing the different departments. Only a few changes have been done to refresh the UI so that the link between users and assigned user groups could be instantly visible.

The screenshot displays the 'Groups View' interface. On the left is a sidebar with three department links: 'RCM Dept', 'Safety Dept', and 'A/C Design Dept'. The main content area is titled 'Group Details' and includes a 'Create new Group' button in the top right corner. The 'Group Details' section shows the following information:

- Name:** A/C Design Dept
- Type:** A/C Design Dept
- Description:** A/C Design Dept

Below this is the 'Members' section, which contains a table with the following data:

| Display Name | First Name | Last Name | Email | |
|--------------|------------|-----------|---------------------------|---|
| Mushfiq | Mush | Rahman | mushfiq.rahman@airbus.com | ✕ |
| Milton | Milton | amador | milton@airbus.com | ✕ |

Underneath the members table is the 'Associated Process Steps Per Department' section, which includes a 'Component Details' button with a close icon (✕). At the bottom is the 'Available Process Steps' section, which lists three items, each with a plus icon (+) for addition:

- RCM Maintenance Concept
- System FMECA
- System FMEA

Figure 6.4: Groups View

Figure 6.4 shows the association between users in "Members" section and "Associated Process Steps Per Department" displays the number of process steps are associated with this step. The rest of the process steps for the component are shown in "Available Process Steps" area.

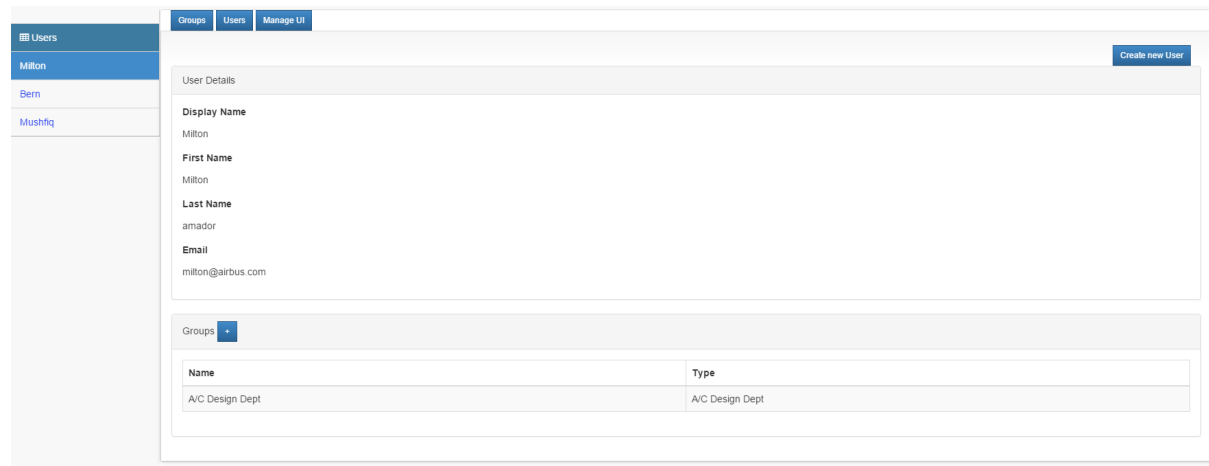


Figure 6.5: Users View

Figure 6.5 displays user association with groups. In this case, an user was associated with A/C Design Department. The layout and the complete functionality of the “Groups View”, as well as the “Users View”, were properly implemented.

6.5.3 My Tasks View

The functionality to select personalized process steps with the options “Select All” or select “My Tasks” had a bug related to a wrong selection of “My tasks”. When selecting the “My Tasks” option the hierarchy was shown correctly, but clicking on the tree brought sometimes an error. Afterwards, this error has been fixed accordingly.

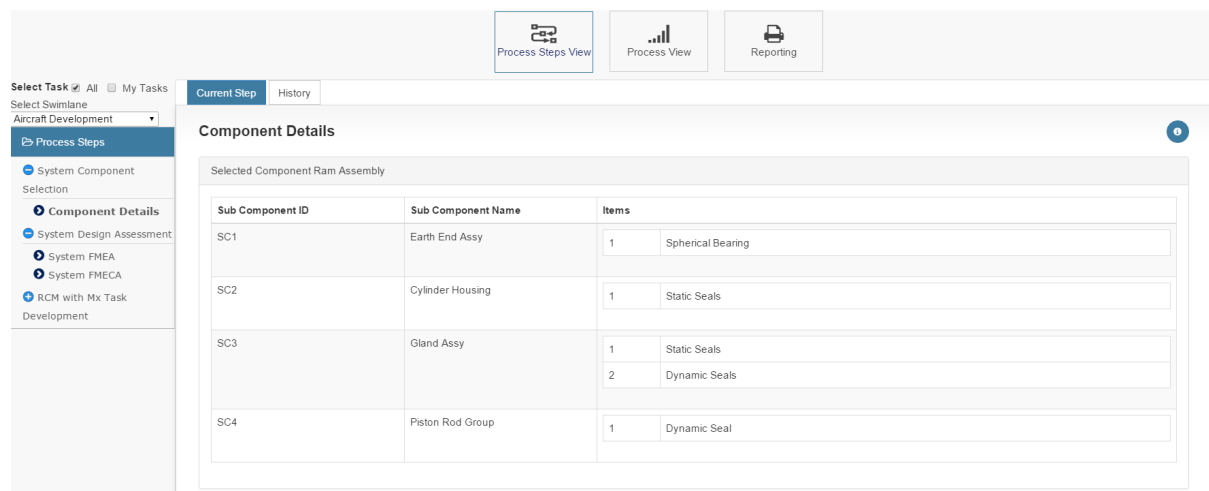


Figure 6.6: All Tasks

After selecting "My Tasks" the tasks, that are assigned to the user, are shown. In Figure 6.7 shows the list. Some UI modifications are necessary to show the process groups and process steps accordingly.

| Sub Component ID | Sub Component Name | Items |
|------------------|--------------------|-----------------------------------|
| SC1 | Piston Rod Group | 1 Dynamic Seal |
| SC2 | Earth End Assy | 1 Spherical Bearing |
| SC3 | Cylinder Housing | 1 Seals 2 Static Seals |
| SC4 | Gland Assy | 1 Static Seals 2 Dynamic Seals |

Figure 6.7: Assigned Task to User

All buttons and functionalities not implemented yet, have been disabled to avoid confusion on the test users.

6.5.4 Process Step Detail View

Each process step contains three main areas. The first area provides the functionality to add specific step related data, and the second area is a Model-Based UI approach to get more information of the process steps from the user. The third area is to add a way to give links and documents that are related to the step. In this section, we will validate the third area, links, and document's insertion.

Add New Link

Add new Link functionality operated correctly. Foldable function and add link function have been successfully implemented.

The screenshot shows the 'Add New Link' form and the 'Available Links' table. The 'Add New Link' form has fields for URL, Name, and Details, with 'Save' and 'Cancel' buttons. The 'Available Links' table lists two links: 'http://www.airbus.com/aircraftfamilies/passengeraircraft/a380family/' and 'www.google.com'.

Add New Link

URL:

Name:

Details:

Available Links

| | Name | URL | Description | Delete |
|-------------------------------------|--|------------|---------------------------------------|-------------------------------------|
| <input checked="" type="checkbox"/> | http://www.airbus.com/aircraftfamilies/passengeraircraft/a380family/ | Airbus 380 | Airbus 380 , Boost your profitability | <input checked="" type="checkbox"/> |
| <input checked="" type="checkbox"/> | www.google.com | Google | search engine | <input checked="" type="checkbox"/> |

Documentation :

Figure 6.8: Upload and View Links to process step

Add New Document

Add New Document functionality operated correctly. Fold-able function and add new Document function have been successfully implemented.

The screenshot shows the 'Add New Document' form and the 'Available Documents' table. The 'Add New Document' form has an 'Upload a file' button and a 'Choose a file' button. The 'Available Documents' table lists two documents: 'General Mx Credit Process_171215.pdf' and 'Detailed Mx Credit Process_171215.pdf'.

Add New Document

Upload a file (*.pdf, .doc, .txt)

Available Documents

| | File Name | Download | Delete |
|-------------------------------------|---------------------------------------|--------------------------|-------------------------------------|
| <input checked="" type="checkbox"/> | General Mx Credit Process_171215.pdf | Download | <input checked="" type="checkbox"/> |
| <input checked="" type="checkbox"/> | Detailed Mx Credit Process_171215.pdf | Download | <input checked="" type="checkbox"/> |

Figure 6.9: Upload and View Documents to process step

6.5.5 Process Execution View

To illustrate the process execution in MCF Tool, we have used four process steps. To show the process step navigation more precise, we will describe it for two process steps.

First process step is *Component Details* and second one is *System FMEA*. In Figure 6.10 section [1] shows current process step in the Process Steps tree view menu. When Submit button is selected, focused on section [2] it goes to next process step *System FMEA*.

Select Task: All My Tasks
Select Swimlane: Aircraft Development
Process Steps
System Component Selection
Component Details
System Design Assessment
RCM with Mx Task Development

Current Step History

Component Details

Selected Component Ram Assembly

| Sub Component ID | Sub Component Name | Items |
|------------------|--------------------|-----------------------------------|
| SC1 | Earth End Assy | 1 Spherical Bearing |
| SC2 | Cylinder Housing | 1 Static Seals |
| SC3 | Gland Assy | 1 Static Seals 2 Dynamic Seals |
| SC4 | Piston Rod Group | 1 Dynamic Seal |

Data Source :
Documentation :

Submit Save Cancel

Figure 6.10: Process Step *Component Details*

As shown in the Figure *System FMEA* is active now and transition happened after the user selected submit button in *Component Details* step.

Select Task: All My Tasks
Select Swimlane: Aircraft Development
Process Steps
System Component Selection
System Design Assessment
System FMEA
System FMECA
RCM with Mx Task Development

Current Step History

System FMEA

System Bottom Up FMEA

Select Sub Component

| Item ID | Failure Mode | Defect Description | Defect Effects | Component Reliability |
|----------------|--------------|--------------------|----------------|-----------------------|
| + Add New Item | | | | |

Selected Component : Ram Assembly

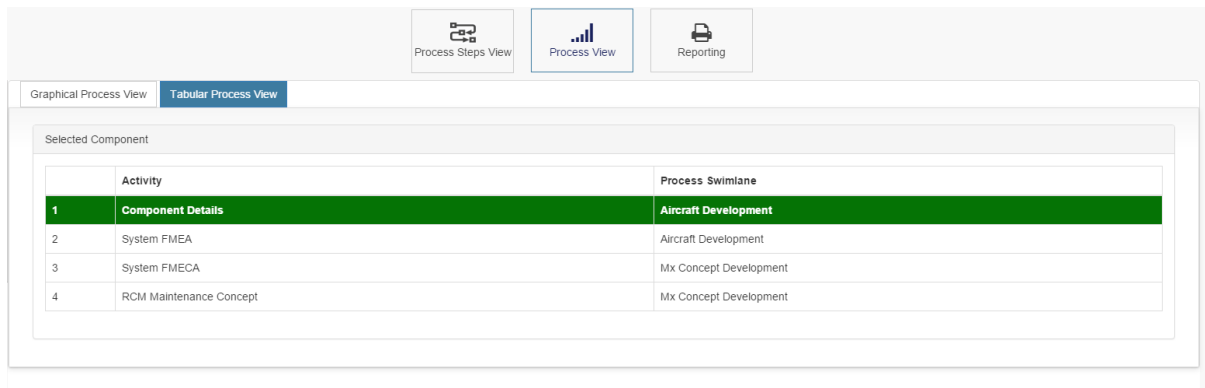
Data Source :
Documentation :

Submit Save Cancel

Figure 6.11: Process Step *System FMEA*

6.5.6 Tabular Process View

To evaluate the Tabular process view we will use the process steps mentioned in section 6.5.5. When save button is pressed in *Component Details* process goes to saved state and in the tabular process view shows its status as green. In Figure 6.12, shows status of the process step. All the other process steps are shown in white color as no data is submitted.

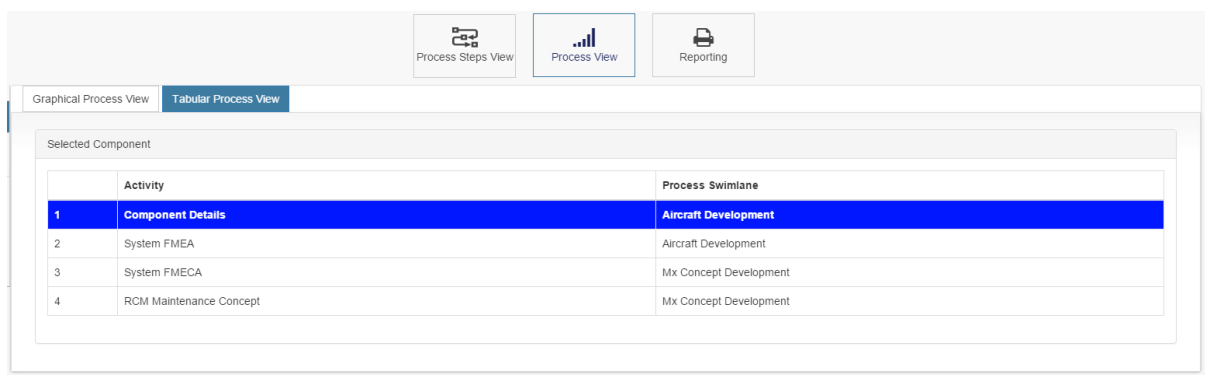


The screenshot shows the 'Tabular Process View' interface. At the top, there are three buttons: 'Process Steps View', 'Process View' (which is highlighted with a blue border), and 'Reporting'. Below these buttons, there are two tabs: 'Graphical Process View' and 'Tabular Process View' (which is selected). The main content area is titled 'Selected Component' and contains a table with two columns: 'Activity' and 'Process Swimlane'.

| | Activity | Process Swimlane |
|---|-------------------------|------------------------|
| 1 | Component Details | Aircraft Development |
| 2 | System FMEA | Aircraft Development |
| 3 | System FMECA | Mx Concept Development |
| 4 | RCM Maintenance Concept | Mx Concept Development |

Figure 6.12: Process Step *Component Details*

After clicking on the submit button, shown in Figure 6.11, the *Component Details* step goes to submitted state. In principle, no modification or addition of data is possible at that point. So the tabular process view is updated and shows *Component Details* process steps in Blue color.



The screenshot shows the 'Tabular Process View' interface after the 'Component Details' step has been submitted. The interface is identical to Figure 6.12, but the first row of the table, 'Component Details' under 'Aircraft Development', is highlighted in blue.

| | Activity | Process Swimlane |
|---|-------------------------|------------------------|
| 1 | Component Details | Aircraft Development |
| 2 | System FMEA | Aircraft Development |
| 3 | System FMECA | Mx Concept Development |
| 4 | RCM Maintenance Concept | Mx Concept Development |

Figure 6.13: Process Step *Component Details*

As process step transition is evaluated in section 6.5.5, shows current active process step is *System FMEA*. When some data is inserted in this step and Save button is pressed it goes to saved state. In Figure 6.14 section [2] shows *System FMEA* is in green state.

| | Activity | Process Swimlane |
|---|-------------------------|------------------------|
| 1 | Component Details | Aircraft Development |
| 2 | System FMEA | Aircraft Development |
| 3 | System FMECA | Mx Concept Development |
| 4 | RCM Maintenance Concept | Mx Concept Development |

Figure 6.14: Process Step *System FMEA*

section [1] shows the status of *Component Details* and section [3] shows rest of the process steps status.

6.5.7 Model Based UI Generation

Model based UI generation is tested with given json schema. Presented in Figure 6.15, gives the JSON schema for *Component Details* step.

```

{
  "1": {
    "form_id": "1",
    "form_name": "My Test Form",
    "form_fields": {
      "1": {
        "field_id": 1,
        "field_title": "Description",
        "field_type": "textfield",
        "field_value": "no details",
        "field_required": true,
        "field_disabled": false
      }
    }
  },
  "2": {
    "field_id": 2,
    "field_title": "Line Break",
    "field_type": "linebreak",
    "field_value": "Doe",
    "field_required": true,
    "field_disabled": false
  }
}

```

Figure 6.15: Process Step *Manage UI with JSON Schema*

The schema contains one text box and one line break. Once the schema is saved, it is used as a template for the process step. So in the *Component Details* step shown in Figure 6.16, shows the text-box and line break web control.

Component Details

Selected Component Ram Assembly

| Sub Component ID | Sub Component Name | Items |
|------------------|--------------------|-----------------------------------|
| SC1 | Piston Rod Group | 1 Dynamic Seal |
| SC2 | Earth End Assy | 1 Spherical Bearing |
| SC3 | Cylinder Housing | 1 Seals 2 Static Seals |
| SC4 | Gland Assy | 1 Static Seals 2 Dynamic Seals |

1

Description: Component details

Figure 6.16: Process Step *Componet Details* process step with generated UI from JSON Schema

Section 1 points to the generated web controls. In this case one text-box and line-break. So web control generation based on Schema defined in manage section works perfectly.

6.5.8 Process Steps View

To evaluate the process step view we have used use case tables mentioned in Section 6.3. To validate the process steps we have collected the use case data first, then load it accommodatingly for each step. Here we describe four process steps with detail status after inserting use case data.

Component Details View

In this process step we have used Table B.1 data to verify *Component Details* process step. Use case contains data for System "RAM Assembly" with its Sub Systems and Items. Figure 6.17 shows the status of the steps, after filling data.

The screenshot displays the 'Component Details' process step. The sidebar on the left includes a 'Select Task' dropdown set to 'Aircraft Development', a 'Process Steps' section with 'System Component Selection' and 'Component Details' (the latter being active), and a 'System Design Assessment' section with 'RCM with Mx Task Development'. The main area is titled 'Component Details' and shows 'Selected Component Ram Assembly'. It contains a table with two sub-components: 'SC1' (Gland Assy) and 'SC2' (Cylinder Housing). Each sub-component has a list of items, specifically 'Static Seals'. At the bottom, there are 'Data Source' and 'Documentation' links, and 'Submit', 'Save', and 'Cancel' buttons.

| Sub Component ID | Sub Component Name | Items | | | | |
|------------------|--------------------|---|---|------------------|---|------------------|
| SC1 | Gland Assy | <table border="1"> <tr> <td>1</td> <td>Static Seals (1)</td> </tr> <tr> <td>2</td> <td>Static Seals (2)</td> </tr> </table> | 1 | Static Seals (1) | 2 | Static Seals (2) |
| 1 | Static Seals (1) | | | | | |
| 2 | Static Seals (2) | | | | | |
| SC2 | Cylinder Housing | <table border="1"> <tr> <td>1</td> <td>Static Seals (3)</td> </tr> <tr> <td>2</td> <td>Static Seals (4)</td> </tr> </table> | 1 | Static Seals (3) | 2 | Static Seals (4) |
| 1 | Static Seals (3) | | | | | |
| 2 | Static Seals (4) | | | | | |

Figure 6.17: Process Step *Componet Details* Process Step evaluation using Use case tableB.1

All the Sub Systems and Items are shown accordingly. So we can make decision that this step reflects the data perfectly.

System FMEA View

Table B.2 used to validate *System FMEA View*. From the use case, we can find out that, items of a Sub Component will have some additional data like Defect Description, Defect Effect. The goal of this step is to connect this data to the item and create an unique Failure Mode(FM) id for each of them. So that, for further calculation in next process steps of the application FM and Item pair can be used.

Select Task [All] [My Tasks]
 Select Swimlane
 Aircraft Development
 Process Steps
 System Component Selection
 System Design
 Assessment
 System FMEA
 System FMECA
 RCM with Mx Task Development

System FMEA

System Bottom Up FMEA

Select Sub Component: Gland Assy

| Item ID | Failure Mode | Defect Description | Defect Effects | Component Reliability | |
|------------------|--------------|---|---|--------------------------|---|
| Static Seals (1) | FM1 | Leaking due to ageing condition, exposure to high temperature, abrasion or faulty seal installation | Damaged seal, minor internal leakage No functional effect | 2 (0.5 ¹⁰ -6) | ✗ |
| Static Seals (2) | FM2 | Leaking due to ageing condition, exposure to high temperature, abrasion or faulty seal installation | Damaged seal, major internal leakage No functional effect | 0.3 (3 ¹⁰ -6) | ✗ |

+ Add New Item

Selected Component : Ram Assembly

Data Source : [icon]
 Documentation : [icon]

Submit Save Cancel

Figure 6.18: Process Step *System FMEA* Process Step evaluation using Use case table B.2

Figure 6.18 shows unique FM1 is created for "Status Seals(1)" Item. Also, other values are shown appropriately.

System FMECA View

Use case data shown in Table B.3, is used to validate *System FMECA* process step. The data is depicted for the selected items. There should be other related data like "Criticality class" or "Mitigation Required" data. Furthermore, for each association of these data, a unique criticality ID should be generated. So a pair of Criticality IDs and Item IDs could be used in further use cases of the MCF Tool.

The screenshot shows the 'System FMECA' tool interface. On the left, a sidebar lists tasks: 'System Component Selection', 'System Design Assessment', 'System FMEA', 'System FMECA' (selected), and 'RCM with Mx Task Development'. The main area is titled 'System FMECA' and has tabs for 'Current Step' and 'History'. Below the title, it says 'System Bottom Up FMEA'. A dropdown menu 'Select Sub Component' is set to 'Gland Assy'. A table displays the following data:

| Item ID | Criticality ID | Criticality Class | Mitigation Required | |
|------------------|----------------|-----------------------|---|---|
| Static Seals (1) | CR1 | 4 Minor (Spoiler) | No Mitigation required => Corrective Mx | ✗ |
| Static Seals (1) | CR2 | 3 Major (Aileron) | No Mitigation required => Corrective Mx | ✗ |
| Static Seals (1) | CR3 | 3 Major (Spoiler) | No Mitigation required => Corrective Mx | ✗ |
| Static Seals (2) | CR4 | 2 Hazardous (Aileron) | Maintenance | ✗ |

Below the table is a '+ Add New Item' button. At the bottom, it says 'Selected Component : Ram Assembly', 'Data Source : [icon]', and 'Documentation : [icon]'. There are 'Submit', 'Save', and 'Cancel' buttons.

Figure 6.19: Process Step *System FMECA* Process Step evaluation using Use case tableB.3

After filling data from the use case, Figure 6.19 shows details for Sub-System "Gland Assembly". It shows the generated Criticality ID's with respect to Items. 6.20 displays the associated data for Sub-System "Cylinder Housing".

The screenshot shows the 'System FMECA' tool interface with the 'Select Sub Component' dropdown set to 'Cylinder Housing'. The table displays the following data:

| Item ID | Criticality ID | Criticality Class | Mitigation Required | |
|------------------|----------------|-----------------------|---|---|
| Static Seals (4) | CR4 | 2 Hazardous (Aileron) | | ✗ |
| Static Seals (4) | CR5 | 3 Major (Spoiler) | No Mitigation required => Corrective Mx | ✗ |
| Static Seals (3) | CR6 | 3 Major (Aileron) | Maintenance | ✗ |
| Static Seals (3) | CR7 | 4 Minor (Spoiler) | No Mitigation required => Corrective Mx | ✗ |

Below the table is a '+ Add New Item' button. At the bottom, it says 'Selected Component : Ram Assembly', 'Data Source : [icon]', and 'Documentation : [icon]'. There are 'Submit', 'Save', and 'Cancel' buttons.

Figure 6.20: Process Step *System FMECA* Process Step evaluation using Use case tableB.3

For this step we can find out that all the functionalities are working well.

RCM Maintenance Concept

Use case data shown in Table B.4 is used to validate *RCM Maintenance Concept* process step. For this step it is important to relate the Criticality Class that is described in section 6.5.8, to the Item. So when Sub Component is selected, all the Items with related data should be loaded including the Criticality ID from *System FMECA* process step. So when adding data for this *RCM Maintenance Concept* step, criticality class should be automatically loaded.

Select Task ☐ All ☐ My Tasks

Select Swimlane
Aircraft Development

Process Steps

- System Component Selection
- System Design Assessment
- RCM with Mx Task Development
- RCM Maintenance Concept**

RCM Maintenance Concept

System Bottom Up FMEA

Select Sub Component: Gland Assy

Select Item: Static Seals (2)

| Criticality Class | Hidden/Evident Failure | End Effects | Mitigation Method | Mitigation Evaluation | FSA & MSA Impact | Mx Concept |
|-------------------|------------------------|---|--|-------------------------------------|--|---------------------------------|
| CR4 | Hidden Failure | Undetectable defect having no immediate functional effect | Internal Leak Check with intrusive inspection (test bench) | Cond Mx Policy(500hrs inspection) ? | Complaint impact of costs are acceptable | Cond Mx (5000Hrs inspection*) ✖ |

+ Add New Item

Selected Component : Ram Assembly

Data Source :

Documentation :

Figure 6.21: Process Step *System FMECA Process Step* evaluation using Use case table B.4

After submitting data from the use case, for Sub-Component, "Gland Assy" data is being illustrated like in Figure 6.21. Figure 6.22 shows data for the Sub Component "Cylinder Housing" and Item "Static Seals (4)".

The screenshot displays the 'RCM Maintenance Concept' tool interface. On the left, a sidebar shows the 'Process Steps' menu with options like 'System Component Selection', 'System Design Assessment', 'RCM with Mx Task Development', and 'RCM Maintenance Concept'. The main area is titled 'RCM Maintenance Concept' and contains a 'System Bottom Up FMEA' section. This section includes dropdowns for 'Select Sub Component' (Cylinder Housin) and 'Select Item' (Static Seals (4)). Below these is a table with the following data:

| Criticality Class | Hidden/Evident Failure | End Effects | Mitigation Method | Mitigation Evaluation | FSA & MSA Impact | Mx Concept |
|-------------------|------------------------|---|---------------------------------|---------------------------------|------------------|---|
| CR6 | Evident Failure | Detectable defect results in loss of function | Pre-flight inspection (In-Situ) | Visual inspection before flight | Compliant | Aileron: On Condition Mx (Visual inspection before flight **) ✖ |

Below the table is an 'Add New Item' button. At the bottom, there are fields for 'Selected Component: Ram Assembly', 'Data Source', and 'Documentation', followed by 'Submit', 'Save', and 'Cancel' buttons.

Figure 6.22: Process Step *System FMECA* Process Step evaluation using Use case table B.4

Some UI modification were done for viewing the large text. And also, we found a bug as Criticality class has some problems while loading with the selected items. Although adding new values are working fine, there are some future works to mitigate the data loading errors.

6.6 Analysis of MCF Tool Evaluation

While evaluating the MCF Tool we find out more options to improve the Tool. We have also experienced bugs mentioned in section 6.5.3 and 6.5.8, while filling out the data from the use cases. We find out that the process flow navigation is in the simplest form now available, and no logic based navigation was implemented. It would be one of the most important features for the next phase.

1. Process steps view should show the versions of saved status. So if process step data is saved two times the process view status table should show the two versions and also should show the data saved time.
2. After submitting a process step, the state should be “locked state” and no more modifications or addition of data should be available.
3. The graphical view in the “process view” section should show the updated status of the process steps using the status color. And it should show the real time status of the process status of the selected component.

4. For the overview of the metadata, that is added to the process steps as a headline, there should be an optimized way to show them in the next version.
5. adding documents in the process steps should also allow the file type excel and power point.

In the end, we can state that the developed MCF Tool satisfies the uses cases described in this chapter. In addition, we found out the limitations of this version of the MCF Tool, and it helped us to organize the future work for the next phase of development.

Chapter 7

Conclusion and Future Work

The main objective of the thesis was the design and implementation of a server client application to support all stakeholders of the Mx Credit Framework (MCF). This was done by following the Design Science Research approach. The implemented Web UI showed at a glance the advantages of storing, processing, linking and retrieving the collected content along the process navigation using graph data model and model-based UIs.

Initially, the environment was explored. A set of requirement list was collected for modeling the Mx Credit process. Based on the importance and priority, the requirements were grouped. After collecting prioritized requirement set, a state of the art study was conducted to find out the right solution approach.

A couple of well-known BPM tools were analyzed in detail. Among With the BPM tools like Activiti, Bonita, jBPM, a Model-Based UI approach was also briefly explored. In contrast, with the requirements, a comparison list was produced. It was identified that, none of the existing BPM tools can adapt the important requirements of Mx Credit process. Although for some tools it is possible to design the process model with rules, the UI generation and data maintain options are not so adequate. On the other hand, Model-Based UI approach provided a useful solution to accommodate most of the requirements of the MCF Tool. Therefore, Model-Based UI approach was followed to implement the MCF Tool.

After the establishment of the solution approach, development of the MCT Tool was started. MCF Tool is a web-based single-page application. AngularJS was used for user interface and Node.js used for server side execution. A graph based database Neo4j was used for storing the MCF Tool data.

After the development of the MCF Tool, a validation and verification phase was conducted as a part of the evaluation. A use case scenario and correspondent content were provided to find out the status of the MCF web application. It was observed that, developed MCF Tool

performed well for most of the requirements. But there are some important requirements like advance rule implementation for the process model, graphical visualization of the process execution are not available. These unsolved requirements could be the starting point of the next phase of the MCF Tool development.

Besides the first goal to provide a collaborative, process-oriented environment for all stakeholders involved in the design, integration and certification of ISHM applications and correspondent Mx credits, the conducted work also provided the basis for all further implementation activities related to the defined MCF approach.

The implemented Web UI is a first promising step towards a guided content navigation along the complex aircraft development process. It allows keeping track of and tracing all relevant activities to achieve Mx Credits for ISHM applications.

ISHM definitely has the potential to improve safety. In order to fulfill the economical needs of commercial operators, the challenge is to be able to determine the economic advantage and to overcome regulatory restrictions [28].

While ISHM can certainly contribute to safety improvements, its acceptance will depend to a large degree on the cooperation of regulatory authorities and, as well as, on the ability to calculate a positive cost-benefit [28].

It should be expected that more importance will be placed on specifying the effect of safety improvements on systems operations in the future. Moreover, as ISHM systems mature, the benefit of ISHM will not only be in contributing safety-related knowledge, but will also be in taking action that will improve the state of safety of a system through autonomous action [28].

Future work will keep continuously focusing on the adaptation of the MCF to the ongoing changes in the ISHM community, while extending and validating the MCF approach and refining accordingly the MCF tool.

Special emphasis will still be put on ISHM and Mx Credits benefits in terms of ISHM development costs, reduction of ownership costs, safety, quality and reliability to evaluate application advantages and get the approval by the relevant certification authorities.

Bibliography

- [1] Vaishnavi, V., and Kuechler. W. Design research in information systems, 2004.
- [2] Cameleon Reference Framework : http://www.w3.org/community/uad/wiki/Cameleon_Reference_Framework
- [3] Takeda, H., Veerkamp, P., Tomiyama, T., and Yoshikawa. H. Modeling design processes. AI Mag. 11, 4 (1990), 37-48.
- [4] D. Jannach, M. Zanker, M. Ge, M. Groening. Recommender Systems in Computer Science and Information System – a Landscape of Research. – Germany and Austria.
- [5] UsiXML : <http://www.w3.org/2005/Incubator/model-based-ui/wiki/UsiXML>
- [6] Rik Van Bruggen,: Learning Neo4j. Packt Publishing, 2014.
- [7] Ali Mesbah, Arie van Deursen. Migrating Multi-page Web Applications to Single-pageAJAX Interfaces, 2nd revision. March 2007.
- [8] Activiti Components : <http://activiti.org/components.html>
- [9] Activiti Explorer : <http://www.activiti.org/userguide/#activitiExplorer>
- [10] Edward Benson. Mockup Driven Web Development, May 2013.
- [11] Jia Zhang and Jen-Yao Chung. Mockup-driven Fast-prototyping Methodology for Web Application Development, 2003.
- [12] Bonita BPM Portal Administration : <http://documentation.bonitasoft.com/product-bos-sp/bonita-bpm-portal-administration>
- [13] Bonitasoft : <http://www.bonitasoft.com/>
- [14] Glenn E Krasner, Stephen T Pope. A description of the model-view-controller user interface paradigm in the smalltalk-80 system, 1988.
- [15] Artem Syromiatnikov, Danny Weyns. A Journey Through the Land of Model-View-Design Patterns, April 2014.

- [16] Ken Peffers, Tuure Tuunanen, Marcus A. Rothenberger, Samir Chatterjee. A Design Science Research Methodology for Information Systems Research, 2014.
- [17] Simon, H. A, The Sciences of the Artificial, 1996.
- [18] Reveley M, Reveley, M. Systems Analysis of NASA Aviation Safety Program, 2010.
- [19] Type certificate: https://en.wikipedia.org/wiki/Type_certificate
- [20] Introduction to Model-Based User Interfaces: <http://www.w3.org/TR/mbui-intro/>
- [21] National Aeronautics Research and Development Plan, Biennial Update, National Science and Technology Council, 2010.
- [22] Airworthiness procedures manual, Directorate General of Civil Aviation, India 2013.
- [23] Premaratne Samaranayake, Senevi Kiridena. Aircraft maintenance planning and scheduling: an integrated framework, 2012.
- [24] Nowlan, S. “Decision diagram approach to ‘On Condition’ philosophies”, Aircraft Engineering, 1972.
- [25] Mr. Shannon. P, Ackert. Basics of Aircraft Maintenance Programs for Financiers, 2010.
- [26] Military Airworthiness, European Defence Agency.
- [27] Quality Control Management, Approved Maintenance Organizations, 2012.
- [28] Kai Goebel, Safety and IVHM, NASA Ames Research Center, 2012.
- [29] EUR-Lex, Access to European Union Law <http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32003R2042>
- [30] Premaratne Samaranayake, Senevi Kiridena. Aircraft maintenance planning and scheduling: an integrated framework, 2012.
- [31] Mary S. Reveley, Jeffrey L. Briggs. Commercial Aircraft Integrated Vehicle Health Management Study, 2012.
- [32] Aeronautical Design Standard Handbook, Condition Based Maintenance System for US Army Aircraft, 2013.
- [33] Alan R. Hevner, Salvatore T. March, Jinsoo Park. Design Science in Information Systems Research, 2004.
- [34] Nilesh Jain, Priyanka Mangal, Deepak Mehta. AngularJS: A Modern MVC Framework in JavaScript, 2014.

- [35] Paulo Bastos. Finite satisfiability verification in uml class diagrams – a comparative study, 2007.
- [36] Gregory W. Vogl, Brian A. Weiss, M. Alkan Donmez. Standards Related to Prognostics and Health Management (PHM) for Manufacturing, 2014.
- [37] Prashish Rajbhandari, Rabi Chandra Shah, Sonali Agarwal. Graph Database Model for Querying, Searching and Updating, 2012.
- [38] Roy Thomas Fielding. Architectural styles and the design of network-based software architectures, 2000.
- [39] Airbus Group, In-house Documentation, 2015.
- [40] Reliability-Centered Maintenance(RCM) for Command,Control,Communications,Computer,Intelligence and Reconnaissance(C4ISR) facilities, 2003.
- [41] Pedro J.Molina. A Review to Model-Based User Interface Development Technology, 2004.

Appendices

Appendix A

List of Abbreviations

The following table describes the significance of various abbreviations used throughout the thesis.

| | Abbreviation | Meaning |
|-----|--------------|--------------------------------------|
| 1. | RCM | Reliability Centered Maintenance |
| 2. | MSG 3 | Maintenance Steering Group 3 |
| 3. | ISHM | Integrated System Health Management |
| 4. | FMECA | Failure Mode and Effects Analysis |
| 5. | Mx Credits | Maintenance Credits |
| 6. | MCF | Mx Credit Framework |
| 7. | Mx | Maintenance |
| 8. | Mx Credits | Maintenance Credits |
| 9. | FMEA | Failure Modes Effects Analysis |
| 10. | FAA | Federal Aviation Administration |
| 11. | NTSB | National Transportation Safety Board |
| 12. | NDT | Non Destructive Testing |
| 13. | CBM | Condition-Based Maintenance |

Table A.1: Abbreviations list.

Appendix B

EHA Use Case Details

| System Component Selection | | |
|----------------------------|------------------|------------------|
| System | Sub System | Item |
| Ram Assembly | Gland Assy | Static Seals (1) |
| | | Static Seals (2) |
| | Cylinder Housing | Static Seals (3) |
| | | Static Seals (4) |

Table B.1: Use Case Data for System Component Selection process step

| System FMEA | | | | | |
|--------------|------------------|------------------|---|--|-----------------------|
| System | Sub System | Item | Defect Description | Defect Effect | Component Reliability |
| Ram Assembly | Gland Assy | Static Seals (1) | Leaking due to ageing condition, exposure to high temperature, abrasion or faulty seal installation | Damaged seal, minor internal leakage. No functional effect | 2 (0,5*10-6) |
| | | Static Seals (2) | Leaking due to ageing condition, exposure to high temperature, abrasion or faulty seal installation | Damaged seal, major internal leakage. No functional effect | 0.3 (3*10-6) |
| | Cylinder Housing | Static Seals (3) | Leaking due to ageing condition, exposure to high temperature, abrasion or faulty seal installation | Damaged seal, minor external leakage. No functional effect | 4 (0,25*10-6) |
| | | Static Seals (4) | Leaking due to ageing condition, exposure to high temperature, abrasion or faulty seal installation | Damaged seal, major external leakage. Loss of function | 0.5 (2*10-6) |

Table B.2: Use Case Data for System Component Selection process step

| System FMECA | | | | |
|--------------|------------------|------------------|-----------------------|--|
| System | Sub System | Item | Criticality Class | Mitigation required |
| Ram Assembly | Gland Assy | Static Seals (1) | 4 Minor (Spoiler) | No Mitigation required =>Corrective Mx |
| | | | 3 Major (Aileron) | No Mitigation required =>Corrective Mx |
| | | | 3 Major (Spoiler) | No Mitigation required =>Corrective Mx |
| | | | | |
| | | Static Seals (2) | 2 Hazardous (Aileron) | Maintenance |
| | Cylinder Housing | Static Seals (3) | 4 Minor (Spoiler) | No Mitigation required =>Corrective Mx |
| | | | 3 Major (Aileron) | Maintenance |
| | | | 3 Major (Spoiler) | No Mitigation required =>Corrective Mx |
| | | Static Seals (4) | 2 Hazardous (Aileron) | |

Table B.3: Use Case Data for System Component Selection process step

| RCM Maintenance Concept | | | | | | |
|-------------------------|------------------------|---|---------------------------------|---|---------------------------------------|---|
| Item | Hidden/Evident Failure | End Effects | Mitigation Method | Mitigation Evaluation | FSA / MSA Impact | Mx Concept |
| Static Seals (2) | Hidden Failure | Undetectable defect having no immediate functional effect | Lifting Policy for seal | Prev Mx, 2000FH Life? | The impact on costs is not acceptable | Not applicable |
| | | | | Internal Check with intrusive inspection (test bench) | Cond Mx Policy (500hrs inspection)? | Cond Mx (500Hrs inspection*) |
| | | | Pre-flight inspection (In-Situ) | Visual inspection before flight? | | Not applicable |
| Static Seals (3) | Evident Failure | Detectable defect having no immediate functional effect | Pre-flight inspection (In-Situ) | Visual inspection before flight | Compliant | Aileron: On Condition Mx (Visual inspection every 10 sorties**) |
| Static Seals (4) | Evident Failure | Detectable defect results in loss of function | Pre-flight inspection (In-Situ) | Visual inspection before flight | Compliant | Aileron: On Condition Mx (Visual inspection before flight**) |

Table B.4: Use Case Data for RCM Maintenance Concept process step