Operational Software Maturity: An Aerospace Industry Analysis

Raúl González Muñoz, Essam Shehab, Martin Weinitzke, Chris Fowler, Paul Baguley

***Abstract*—**Software applications have become crucial to the aerospace industry, providing a wide range of functionalities and capabilities used during the design, manufacturing and support of aircraft. However, as this criticality increases, so too does the risk for business operations when facing a software failure. Hence, there is a need for new methodologies to be developed to support aerospace companies in effectively managing their software portfolios, avoiding the hazards of business disruption and additional costs. This paper aims to provide a definition of operational software maturity, and how this can be used to assess software operational behaviour, as well as a view on the different aspects that drive software maturity within the aerospace industry. The key research question addressed is, how can operational software maturity monitoring assist the aerospace industry in effectively managing large software portfolios? This question has been addressed by conducting an in depth review of current literature, by working closely with aerospace professionals and by running an industry case study within a major aircraft manufacturer. The results are a software maturity model composed of a set of drivers and a prototype tool used for the testing and validation of the research findings. By utilising these methodologies to assess the operational maturity of software applications in aerospace, benefits in maintenance activities and operations disruption avoidance have been observed, supporting business cases for system improvement.

***Keywords*—**Aerospace, capability maturity model, software maturity, software lifecycle.

I. Introduction

WITHIN the aerospace industry, IT departments manage large portfolios of software applications that support the full lifecycle of aerospace products, providing capability to large numbers of users within the organisations. These capabilities have become crucial to the business and are involved in core processes across the enterprise. As a result, innovative methodologies to improve the management of software portfolios are of interest to the industry.

In the last several decades, rapid evolutions in technology have led to fast-changing application portfolios, and as a result the concept of "maturity" has acquired relevance. Maturity is a widely used concept to analyse the achievement and progression of a set of indicators or attributes in a particular domain or discipline [1].

In order to evaluate the capabilities of an organisation using the concept of maturity, a number of models have been developed over time. These models are instruments to measure the maturity and allow an organisation or industry to have its practices, processes or methods evaluated to determine its current level of achievement or capability and apply these models over time to drive improvement.

According to Becker [2], there are 51 maturity models documented and each one of them has been applied to different domains or disciplines.

In the area of software, one of the most well-known is the Capability Maturity Model (CMM) [3] developed by the Software Engineering Institute. However, this model focuses solely on software maturity during its development stage, not encompassing software behaviour while in usage.

Regarding maturity models for software in operation, Renken [4] developed an IS/ICT Management Capability Maturity Framework with the aim of assessing the capabilities of a company for IT management. Nonetheless, this model has a focus in the capabilities of IT departments, rather than the software applications themselves. Hence, to the author's knowledge, there seems to be a lack of a clear methodology to evaluate software application maturity during usage, this reason being one of the main drivers to conduct the following research.

II. Research Methodology

A clear, accurate and well planned methodology is key to reach the aim and objectives of this research. Between the different options, the author decided to follow a methodology comprised by three main phases divided into seven stages, as it was considered to be the most appropriate regarding the characteristics of the project.

The initial phase focused on the existing literature regarding maturity in software, maturity models, software requirements and software quality. In order to enhance the vision of academia, a group of industry experts was also contacted regularly to engage in a constructive discussion. The participants are shown in Table I.

The second phase involved the creation of a framework encompassing the main areas and drivers that influence the maturity of a software application once in operation. This work was conducted based on the research performed in the previous phase.

The third and final phase encompassed the application of the operational maturity framework within a case study inside

Raúl González Muñoz and Paul Baguley are with the Department of Manufacturing, School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield, Bedfordshire, MK43 0AL, United Kingdom.

Essam Shehab is with the Department of Manufacturing, School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield, Bedfordshire, MK43 0AL, United Kingdom (corresponding author, phone: +44 79 50554 084; e-mail: e.shehab@cranfield.ac.uk).

Martin Weinitzke is with the Airbus Operations, Hamburg, 21129, Germany.

Chris Fowler is with the Airbus Operations, Filton-Bristol, BS34 7PA, United Kingdom.

an aircraft manufacturer. For that end, a review of a set of existing indicators was conducted, as well as a workshop with industry experts, whose participants are illustrated in Table II, with the aim of adapting the model to the key areas of the targeted business. The result was an adapted operational software maturity model, tested through a prototype tool and validated by industry experts.

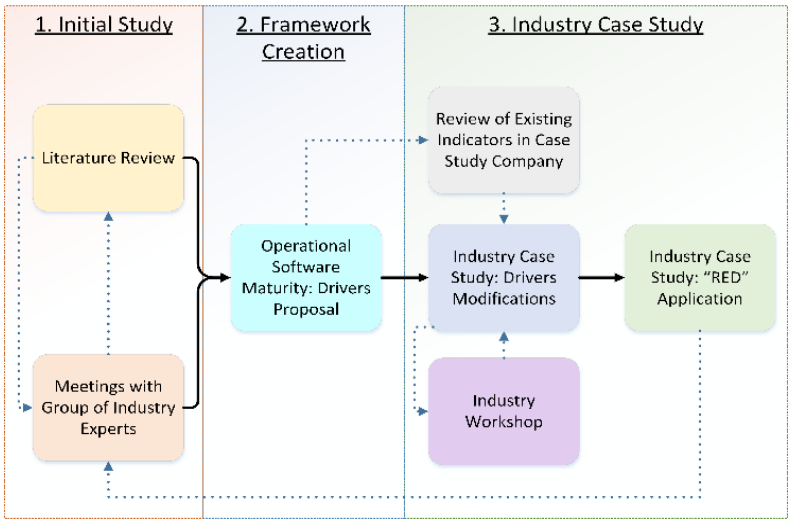

Fig. 1 Research Methodology

A.Industry Participants

For this research, a set of industry experts have been regularly contacted. All the validation from the industry experts regarding the model and the tool as well as feedback regarding a range of aspects was gathered, processed and implemented.

The first group involved in the research was a small core team of industry experts composed of four participants from the aerospace industry and services related, as shown in Table I. This group was actively involved during the whole research, being especially helpful during the case study performed to test the research results with real data, as the responsible of the selected application was among them. Furthermore, the group provided valuable assistance when creating the first generic proposal of software maturity drivers.

TABLE I
INDUSTRY TEAM MEMBERS

Participants	Experience	Role
Participant 1	20 years	Service Package Manager
Participant 2	20 years	SPM Delegate (Contractor)
Participant 3	6 years	Service Manager
Participant 4	29 years	Design Process Architect

The second group consulted, illustrated in Table II, was of a much larger size, intentionally increasing the number of different views on the topic to achieve the best possible set of data to work with. This group was involved through a workshop that was arranged specifically to assess the maturity drivers developed and improve or modify them, with the aim of conducting afterwards an industrial case study within a major aircraft manufacturer.

TABLE II
WORKSHOP PARTICIPANTS

Participants	Experience	Role
Participant 1	30 years	Robustness Manager
Participant 2	15 years	Application Manager
Participant 3	10 years	Total Cost Team (contractor)
Participant 4	10 years	Total Cost Team (contractor)
Participant 5	10 years	Application support
Participant 6	10 years	Service Line Manager
Participant 7	30 years	Head of Application services
Participant 8	20 years	Service Package Manager
Participant 9	20 years	SPM Delegate (Contractor)
Participant 10	25 years	Head of Application Services
Participant 11	30 years	Head of Department
Participant 12	20 years	Service Package Manager
Participant 13	20 years	Service Package Manager
Participant 14	5 years	Product-Owner/SPM Delegate
Participant 15	1 year	Assistance (contractor)
Participant 16	5 years	Business Manager
Participant 17	30 years	Service Package Manager
Participant 18	30 years	Process Lead
Participant 19	30 years	Transition Manager
Participant 20	10 years	Project Manager
Participant 21	5 years	Project Manager Assistant (contractor)
Participant 22	15 years	License Manager
Participant 23	20 years	Reporting
Participant 24	25 years	Finance Manager
Participant 25	20 years	Delegate (contractor)
Participant 26	20 years	Robustness services
Participant 27	20 years	Technical lead (contractor)
Participant 28	20 years	Developer (contractor)
Participant 29	20 years	Delegate (contractor)
Participant 30	25 years	Bundle Manager
Participant 31	15 years	Application Manager
Participant 32	10 years	Service Package Manager

During the workshop a range of topics were covered:

- Introduction of the model in terms of drivers and indicators
- Discussion to reach agreement in the use of several indicators for all the departments
- Agreement on the input data requirements depending on the department
- Remarks that the model needs to align with the security drivers of the company and follow their changes and updates
- Concerns regarding the quality of the input data for some applications

III. Software Quality and Maturity

In order to achieve maturity in an application and to get a software quality product, a better understanding of the user expectations and product attributes need to be analysed and explored in detail to assess what areas of software product quality are important to achieve high maturity. The main idea of most quality models is to break down the complex concept of quality into quality factors that may be broken down again in order to get a hierarchy of quality characteristics.

Quality is a concept which can be used as a needed property to get the required capabilities in a particular domain and, therefore, to achieve the definition of maturity for this research.

The Software Engineering Institute has taken the process management premise of Watts Humphrey: "the quality of a system or product is highly influenced by the quality of the process used to develop and maintain it," [5] and defined Capability Maturity Models that embody this premise.

According to the Software Engineering Institute [1], a maturity model provides:

- "The benefit of a community's experience and knowledge."
- "A common language and shared vision."
- "A way to define what improvement and maturity mean for an organisation."
- "A framework for prioritizing actions."
- "A roadmap for increased maturity."

One of the most widely used Capability Maturity Model is the Capability Maturity Model Integration (CMMI) [3] which has a prescriptive approach to software process improvement. This model has several maturity or capability levels on the way from chaotic processes to highly standardised and optimised processes. Therefore, this model provides a guide of improvement in the processes according to quality assurance standards. It has been understood that good processes produce quality in the software since there is a clear relationship between process and product quality which needs to be established.

Currently in the industry, one of the most used standards is the ISO/IEC 25010 [6], which is illustrated in Fig. 2.

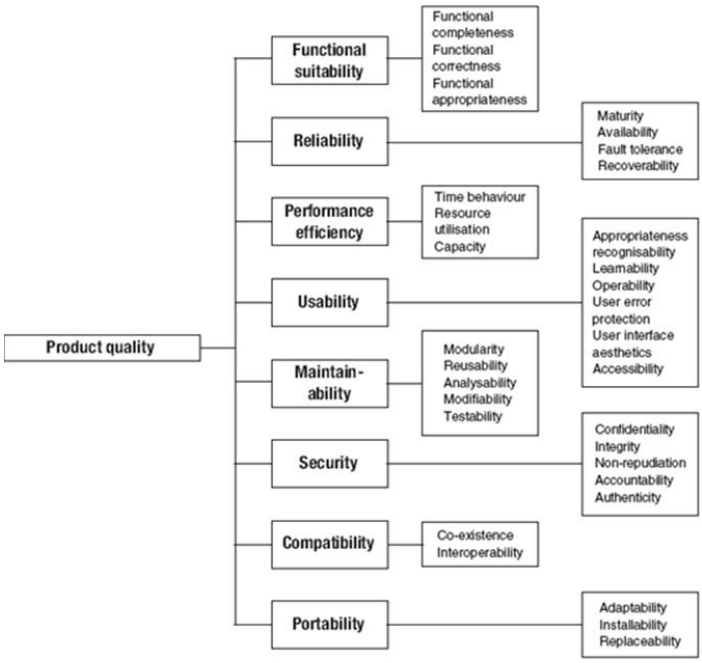

Fig. 2 Product quality model of ISO/IEC 25010 [6]

The model illustrates the hierarchical structure that divides quality into characteristics, which can consist of sub-characteristics and, in turn, of sub-sub-characteristics. This standard provides a guide to ensure quality in the software product but it emphasises that not all characteristics are relevant in every software. It provides no help regarding how to customise the quality model. This standard is an evolution from the previous ISO/IEC 9126 [7], which itself was developed closely based in the model developed by McCall and Matsumoto in 1980 [8], as shown in Fig. 3. This kind of

approach to quality models, which became the basis for several international standards, has been used as one of the main references in the research to develop a suitable operational maturity model for software.

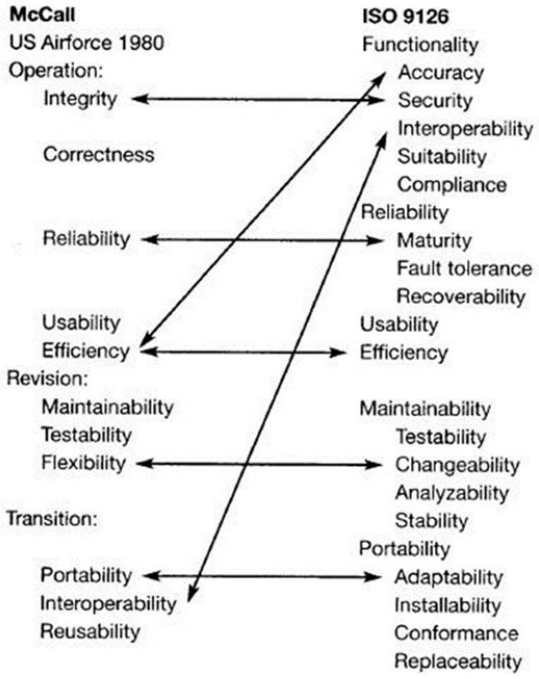

Fig. 3 Quality factor classification [9]

IV. Results

A. Operational Software Maturity: Drivers Proposal

According to de Bruin [10], there are two main ways to design a maturity model, from a top-down or bottom-up approach. Starting from the top, the model would look at the level and progress at maturity divided by characteristic; those models assume the evolution of maturity and are rigid in the paths they can take.

Starting from the bottom, the different characteristics or assessments are then gathered into main areas of maturity to provide a more general view.

The approach selected during the research was the latter one, starting from the bottom, as the changing nature of software through time, depending on the environment in which it operates, makes it very difficult to predict a "rigid path" for a maturity evolution.

During the project, "Maturity" was defined as:

- The capabilities of a software application to perform what is required while in usage, evaluating under which degree of success the applications are meeting the defined requirements of operation.

This definition was later validated through expert elicitation by the industry members depicted in Table I.

As a result of the literature review conducted and the active consultation with industry experts, an Operational Software Maturity Capability Framework was developed, as shown in Fig. 4.

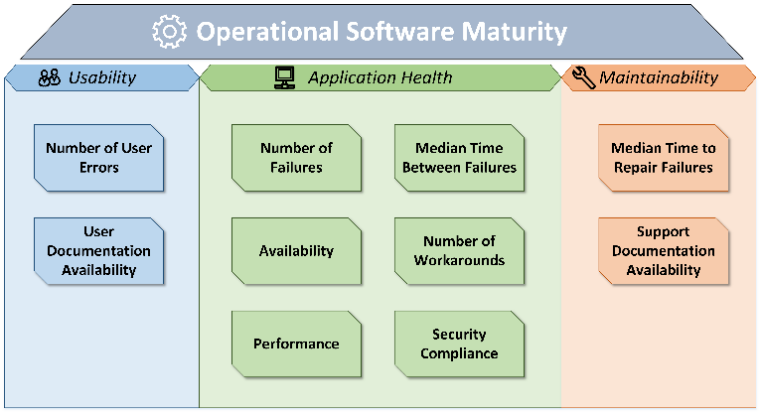

Fig. 4 Operational Maturity Drivers

The framework is divided into three main drivers and several indicators, which are the following:

- **Usability**: to which degree the application is fit for purpose and easy to operate by the user.
  - Number of User Errors: it serves as an indicator of how difficult it is for the user to make proper use of the application. Fewer errors would equal higher maturity.
  - User Documentation Availability: the availability of quality user guides can serve as an indicator of the usability for an application. Lack of them would decrease maturity.
- **Application Health**: to which degree the application can be operated within the service level agreements of the organisation.
  - Number of Failures: the amount of issues of an application for a given time can serve as an indicator of how well it is working in the operative environment. A lower number of issues would mean higher maturity.
  - Availability: compare the level of availability of the application to the requirements of the business. If the availability meets the requirements, the application would have higher maturity.
  - Performance: Performance of the application compared to organisation requirements. If the application meets the requirements, the application would have higher maturity.
  - Median Time Between Failures: time between failures is a

common indicator to assess the robustness of a system. In this case, the use of median has been preferred to the mean, in order to avoid disturbances on the data set due to outliers. A higher time here would translate on a higher maturity.

- Number of Workarounds: compare the evolution of workarounds through time for an application. A higher number of those would mean a lower maturity, as this will cause deficiencies in the long-term.
- Security Compliance: if the application complies with all security requirements of the company. If not, this would equal a lower maturity.

- **Maintainability**: how much effort it takes to keep the application operational.
  - Median time to Repair Failures: is a common indicator used to assess the reparability of assets. A lower time would mean a higher maturity.
  - Support Documentation Availability: the presence of technical documentation depicting maintenance guidelines would translate into a higher maturity.

This framework constitutes a baseline for any company to develop their own metrics to assess Operational Software Maturity of their applications. Furthermore, these drivers and indicators can be modified as well to suit the particularities of the environment in which each organisation operates.

B.Industry Case Study: Drivers Modifications

With the aim of improving the developed framework as well as to show its utility, a case study was conducted in close collaboration with a major aircraft manufacturer.

Initially, the company had been using an approach to the concept of performance/maturity in order to estimate how much effort was required to keep the different applications operational. This approach resulted in a matrix with a set of 10 indicators:

- Maturity of Technology
- Performance Constraints
- Number of interfaces
- Multi-sites application
- Complexity on the infrastructure
- Impact on business functions
- Administration complexity
- Number of users
- Number of lines of code
- Documentation completion

Such a matrix was being filled by the users themselves under their expert opinion. Each indicator could have three different scores, 1, 3 or 5, depending on its perception of low, medium or high, respectively (Ordinal scale). As can be noted, the set of indicators was a mixture of concepts between "Maturity" and "Complexity". Hence, complexity indicators were discarded and a new framework was created focused in operational maturity, based on the previous research (Fig. 4), but also taking into account the particular requirements of the company. Such model main drivers are illustrated in Fig. 4. It can be noted that one of the major changes was the addition of "Security" as an independent driver, rather than just an indicator within "Application Health", due to requirements from the company exposed during the workshop. "Maintainability" and "Usability" were kept as drivers, although some modifications were performed within the indicators used by both drivers.

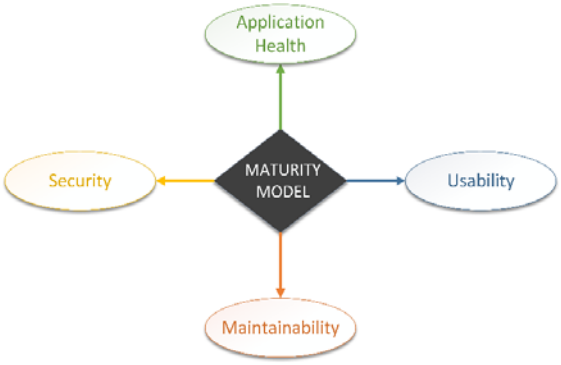

Fig. 5 Adapted Operational Maturity Model

Within each of those four drivers, a number of indicators have been identified, as illustrated in Figs. 6-9. The way of assessing those indicators changed from a subjective point of view to an objective one based on sets of data. Hence, a system based on the Box-and-whisker method was used in several indicators. The Box-and-whisker plot is an exploratory graphic, developed by Tukey in 1977, [11], and it is used to show the distribution of a dataset at a glance. However, the scale used for the evaluation was kept in 1, 3 or 5, effectively keeping the Ordinal scale approach, which is common in many maturity models [10].

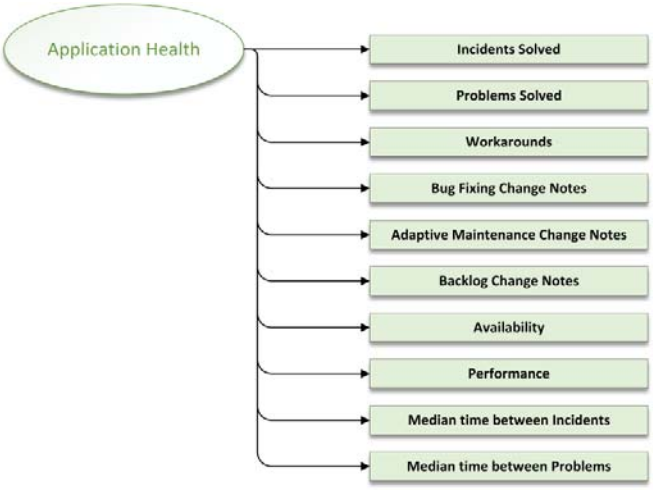

Fig. 6 Application Health Indicators

As a concept, "Application Health" refers to which degree the application can be operated within the service level agreements of the organisation. In this driver, the previous indicator "Performance constraints", from the previous performance/maturity matrix, was developed into two new maturity indicators, "Performance" and "Availability". In addition of those, another eight more indicators are grouped

under this driver, based on the research conducted and shown in Fig. 6. The details of each of the indicators are as follows:

- **Incidents solved:** As a concept, an incident is an issue in which it is known what it is causing it. Regarding maturity, a lower number of incidents mean that the application is more mature in capability terms. In order to assess it, the number of incidents of the application is compared against the total set of data for the portfolio, following a box-and-whisker method with percentiles, a set of scores is given, being 1 if it is over the third quartile, 5 if it is below the first quartile, and 3 if it is between those 2 quartiles. Hence, less incidents results in a higher maturity score.
- **Problems solved:** As a definition, a problem is an issue in which it is not known what it is causing it. Regarding maturity, a lower number of problems mean that the application is more mature in capability terms. In order to assess it, the number of problems of the application is compared against the total set of data for the portfolio, following a box-and-whisker method with percentiles, a set of scores is given, being 1 if it is over the third quartile, 5 if it is below the first quartile, and 3 if it is between those 2 quartiles. Hence, less problems results in a higher maturity score.
- **Workarounds:** A workaround is a temporary fix that implies that a genuine solution to the issue is needed. In order to assess it, the number of workarounds for the application is compared against the total set of data for the portfolio, following a box-and-whisker method with percentiles, a set of scores is given, being 1 if it is over the third quartile, 5 if it is below the first quartile, and 3 if it is between those 2 quartiles. Therefore, the highest number of workarounds the lowest the maturity of the application.
- **Bug Fixing Change Notes:** The number of change notes applied to fix the bugs in an application. In order to assess it, the number of bug fixing change notes of the application is compared against the total set of data for the portfolio, following a box-and-whisker method with percentiles, a set of scores is given, being 1 if it is over the third quartile, 5 if it is below the first quartile, and 3 if it is between those 2 quartiles. Thus, the higher the number of bug fixing change notes the lower the maturity of the application
- **Adaptive Maintenance Change Notes:** The number of change notes implementing small updates linked to the maintenance of the application. In order to assess it, the number of adaptive maintenance change notes of the application is compared against the total set of data for the portfolio, following a box-and-whisker method with percentiles, a set of scores is given, being 1 if it is over the third quartile, 5 if it is below the first quartile, and 3 if it is between those 2 quartiles. Thus, the higher the number of adaptive maintenance change notes the lower the maturity of the application.
- **Backlog Change Notes:** A backlog change note is a change note that is open and has not been applied yet. In order to assess it, the number of backlog change notes of the application is compared against the total set of data for the portfolio, following a box-and-whisker method with percentiles, a set of scores is given, being 1 if it is over the third quartile, 5 if it is below the first quartile, and 3 if it is between those 2 quartiles. Thus, the higher the number of backlog change notes, the lower the maturity of the application.
- **Availability:** By knowing the Service Level Agreement (SLA) of the company, compare the level of availability of the application to the SLA of the business. If the availability is higher than the SLA, the application would be considered more mature.
- **Performance:** Performance of the application compared against the requirement of the company (SLA). If the performance of the application meets the requirements, its maturity will be higher.
- **Median time between Incidents:** Median time between each of the incidents of the application. In order to assess it, the median time between incidents of the application is compared against the total set of data for the portfolio, following a box-and-whisker method with percentiles, a set of scores is given, being 5 if it is over the third quartile, 1 if it is below the first quartile, and 3 if it is between those 2 quartiles. Hence, a higher median time between incidents means a higher maturity.
- **Median time between Problems:** Median time between each of the problems of the application. In order to assess it, the median time between problems of the application is compared against the total set of data for the portfolio, following a box-and-whisker method with percentiles, a set of scores is given, being 5 if it is over the third quartile, 1 if it is below the first quartile, and 3 if it is between those 2 quartiles. Hence, a higher median time between problems means a higher maturity.

Regarding the driver "Usability", it refers to which degree the application is fit for purpose, how easy is for the user to make use of it. In this driver, the indicator "Documentation completion", from the previous performance/maturity matrix, was developed into a new maturity indicator, "User documentation". In addition of that one, two more indicators are grouped under this driver, as shown in Fig. 7.

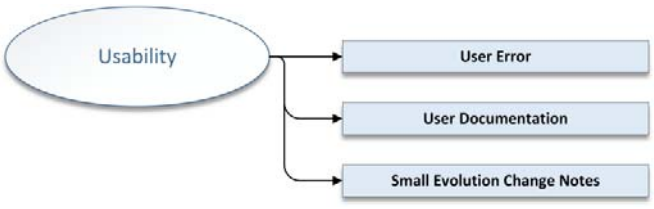

Fig. 7 Usability Indicators

The details of each of the indicators are as follows:

- **User Error:** Regarding number of failures of the application due to a user error. This metric is obtained as a proportion of the user errors taken into account the total number of users of that application. In order to assess it,

the number of user errors of the application is compared against the total set of data for the portfolio, following a box-and-whisker method with percentiles, a set of scores is given, being 1 if it is over the third quartile, 5 if it is below the first quartile, and 3 if it is between those 2 quartiles. Therefore, the higher the number of user errors is, the less mature is the application.

- **User Documentation:** This indicator refers to the availability of a User Guide or not. The availability of a User Guide will mean a higher maturity for the application.
- **Small Evolution Change Notes:** Change notes applied when there is the need to improve minor aspects of the application. In order to assess it, the number of small evolution changes of the application is compared against the total set of data for the portfolio, following a box-and-whisker method with percentiles, a set of scores is given, being 1 if it is over the third quartile, 5 if it is below the first quartile, and 3 if it is between those 2 quartiles. Therefore, a lower number of small evolution change notes will mean a higher maturity.

The concept of "Maintainability" refers to how much effort it takes to keep the application operational. This driver groups six indicators, as shown in Fig. 8, from which several use the box-and-whisker method as a way of assessment.

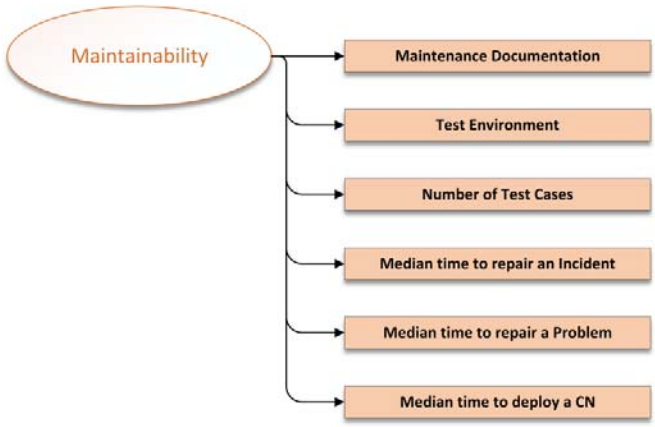

Fig. 8 Maintainability Indicators

The details of each of the indicators are as follows:

- **Maintenance Documentation:** Refers to the availability of documentation that will aid in the maintenance effort for the application. The higher the availability of those documents, the higher the maturity of the application.
- **Test Environment:** Refers to the number of environments an application has been tested in. The more environments available to test application changes, the higher the maturity of that application.
- **Number of Test Cases:** Referring to the availability of the document "Test cases", due to its importance and to the fact that a higher number of such test cases, it is positive for the maintenance of the applications. In order to assess it, the number of test cases of the application is compared against the total set of data for the portfolio, following a box-and-whisker method with percentiles, a set of scores is given, being 5 if it is over the third quartile, 1 if it is below the first quartile, and 3 if it is between those 2 quartiles. Thus, the higher the number of test cases, the more mature is the application.
- **Median time to repair an Incident:** This indicator refers to the amount of time it takes to solve an incident, which is an issue whose cause is known, in a given application. In order to assess it, the median time to repair an incident of the application is compared against the total set of data of the portfolio, following a box-and-whisker method with percentiles, a set of scores is given, being 1 if it is over the third quartile, 5 if it is below the first quartile, and 3 if it is between those 2 quartiles. Therefore, a lower time to repair incidents will result into a higher maturity.
- **Median time to repair a Problem:** This indicator refers to the amount of time it takes to solve a problem, which is an issue whose cause is not known, in a given application. In order to assess it, the median time to repair a problem of the application is compared against the total set of data for the portfolio, following a box-and-whisker method with percentiles, a set of scores is given, being 1 if it is over the third quartile, 5 if it is below the first quartile, and 3 if it is between those 2 quartiles. Therefore, a lower time to repair problems will result into a higher maturity.
- **Median time to deploy a CN:** Median time it takes, since a change note (CN) is raised until it is deployed for a given application. In order to assess it, the median time to deploy a CN of the application is compared against the total set of data of the portfolio, following a box-and-whisker method with percentiles, a set of scores is given, being 1 if it is over the third quartile, 5 if it is below the first quartile, and 3 if it is between those 2 quartiles. Therefore, a lower time to deploy CN will result into a higher maturity.

The driver of "Security", as a concept, assesses to which degree the application is compliant with the security drivers of the organisation. There are six security indicators grouped under this driver, as depicted in Fig. 9.

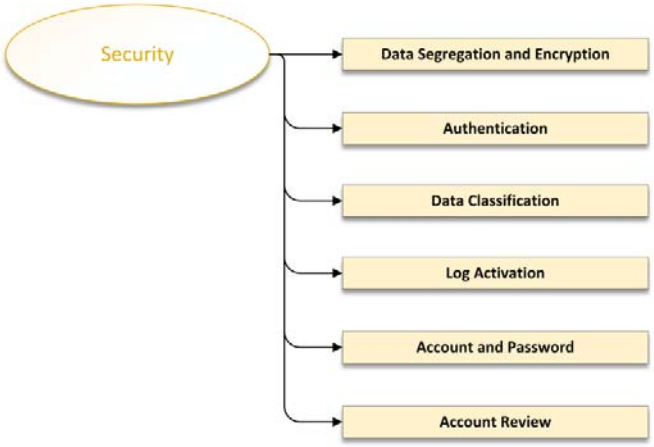

Fig. 9 Security Indicators

Those security drivers were provided by the organization from the case study and they may vary from one company to another. In this driver the box-and-whisker method is not used, as the security indicators have to be assessed usually just once in the life of an application, unless major changes are implemented. An application can have "Full compliance", "Partial compliance" or "No compliance" with the different security indicators. "Full compliance" means high maturity, score of 5, "No compliance" means low maturity, score of 1 and "Partial compliance" means medium maturity, score of 3. The details of each of the indicators are as follows:

- **Data Segregation and Encryption**: Data access segregation has to be implemented using Role based access control and depending on data classification level, encryption has to be implemented also.
- **Authentication:** Implementation of identification based on company official directory using Identity and Access Management solution in order to apply Identity management directives and password policy.
- **Data Classification:** Assessment of the classification level of data managed by the application, so the application is classified at the level of managed data.
- **Log Activation:** All applications have to log a minimal level of events such as authorization activities: logon/logoff/attempt and all administration activities. Depending on the data classification level these logs must be kept three or six months. These logs are different and must be separated from standard applicative logs.
- **Account and Password:** Replacement of all generic accounts into Service or Technical Accounts and disablement of hard coded password or readable (not encrypted) password into all application components.
- **Account Review:** Process required accordingly to application classification aiming at reviewing periodically all declared administrators and end-users.

C.Industry Case Study: Prototype Tool

With the aim of testing the maturity model, a prototype tool was developed, using an Excel spread sheet with embedded formulas and some Visual Basic for Applications (VBA) code. A dashboard was also created to visually analyse the evolution, details and accuracy of the maturity scores. The final goal of the tool was to build a scenario for each application that supports decision making. Its basic structure is illustrated in Fig. 10.

The tool has several tabs/worksheets, which will be navigated by the user to obtain the assessment. The first of the tabs is the "Introduction" tab, and provides the user with an overview of the models and the tool and also has links to the User Guide, in case more information was required. The second tab is the main tab of the tool, "Application 1", and is the Dashboard where all the information regarding the selected application is shown. The last tab is the data entry tab, "AppData", where the data is introduced and stored, coming from a variety of reports and databases. There are as well a number of hidden tabs that perform calculations and store data labels.

The software tool needs to be fed with data in order to have a relevant assessment. It is expected to gather the information needed from different files, filter them with the help of pivot tables, and evaluate them for each metric. An entry per month is expected, to have a valid evolution for constant evaluation.

The different indicators are computed separately, and then gathered into a unified formula. For most of the indicators, if the value of the metric is between two limits, its maturity score is 3; otherwise it is 1 or 5 depending on its impact on maturity (5 is considered as high maturity, whereas 1 is low maturity). Those two limits have been determined using a box-and-whisker plot method [11] that is automatically performed by the tool in a hidden tab for all the applications whose data has been introduced in the tool.

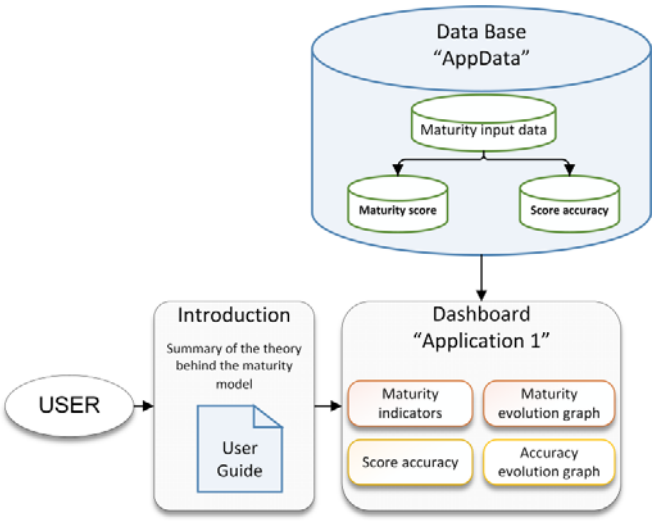

Fig. 10 Maturity prototype tool structure

Once all the different indicators have been computed and weighted, they are all gathered into a single value through (1):

$$\frac{\sum_{i=1}^{n} (w_i x_i)}{\sum_{i=1}^{n} (w_i)} \quad (1)$$

where: w is the weight assigned to each indicator, x is the indicator value, i is the referring month.

To view the results for an application, a Dashboard tab "Application 1" was created. It groups the indicators for the last month of the set of data available, providing the maturity scores for each indicator, together with its assigned weight by the company experts and its data quality. The total maturity score and the weights for each of the drivers are also shown, including the total score accuracy based on the data availability. Furthermore, two graphs showing the evolution through time of both the maturity and the score accuracy are also provided.

D.Industry Case Study: "RED" Application

For the purpose of this paper, a specific application has been chosen to serve as an illustrative example. This application, which will be referred as "RED", accesses Product Data Management (PDM) systems with design data and generates from this data specific data sets (product structure and geometry information), which can be later visualised using software applications for computer-aided design (CAD), computer-aided manufacturing (CAM) and computer-aided engineering (CAE).

Maturity =	3.77	
*Score Accuracy	88.46%	
Drivers	Weights	Value
Usability	15.00%	4.10
Application Health	50.00%	3.7
Maintainability	30.00%	3.5
Security	5.00%	5
Total	100.00%	

Drivers	Indicators	Weight	Value
Usability			4.10
	User Error	45.00%	3
	User Documentation	45.00%	5
	Small Evolution CN	10.00%	5
Application Health			3.7
	Availability	20.00%	5
	Performance	0.00%	5
	Incidents solved	7.50%	5
	Problems solved	12.50%	5
	Workarounds	7.50%	5
	Bug Fixing CN	12.50%	5
	Adaptative Maintenance CN	7.50%	5
	Backlog CN	12.50%	1
	Median time between Incidents	7.50%	1
	Median time between Problems	12.50%	1
Maintainability			3.5
	MTTR incident	12.50%	5
	MTTR problem	25.00%	1
	Maintenance Documentation	12.50%	5
	Test Environment	25.00%	5
	Number of Test cases	12.50%	1
	MTTD a CN	12.50%	5
Security			5
	Data Classification	20.00%	5
	Data Segregation & Encryption	20.00%	5
	Authentication	15.00%	5
	Log Activation	15.00%	5
	Account and Password	15.00%	5
	Account Review	15.00%	5

Fig. 11 "RED" Maturity Indicators

As shown in Fig. 11, the maturity score of "RED" on October is 3.77, which is slightly above medium maturity, with room for improvement. The score accuracy is 88.46%, due to the lack of data for the indicators "Median time between Incidents", "Median time between Problems" and "Median time to deploy a Change Note", as signalled by the three red flags.

As shown in Fig. 12, the maturity of "RED" had a remarkable drop during the months of February and March. This was due to "RED" being a new application deployed by the end of 2013. During the year 2014 "RED" started to take on the functions provided by an older, by then obsolete application. With this entry into service several issues arose, being "Incidents", "Problems" and "Workarounds" the main reason for higher workloads than usual and causing a drop in the maturity. These contingencies were expected, as "RED" had to interact with several applications and databases, each additional interface increasing the risk of malfunctions. However, after several months in operation, and with the required support activities, the improvement is considerable, by October the maturity score had recovered.

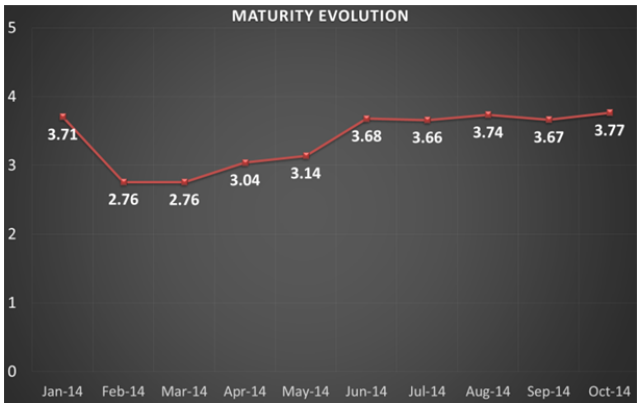

Fig. 12 "RED" Maturity Evolution

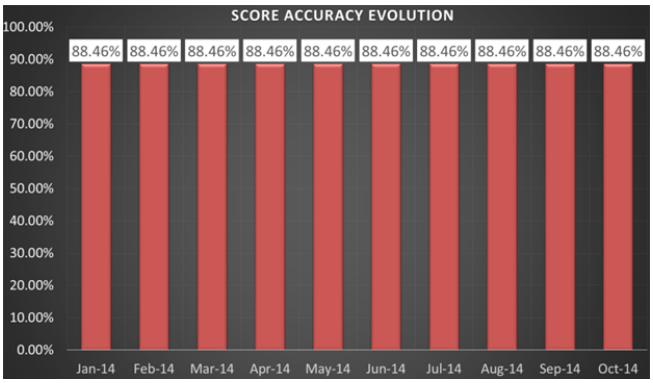

Fig. 13 "RED" Score Accuracy Evolution

The accuracy of the maturity scores provided is shown in Fig. 13. The data availability remained constant from January to October and the previously mentioned indicators of "Median time between Incidents", "Median time between Problems" and "Median time to deploy a Change Note" were at the moment of the study difficult to estimate precisely, due to the novelty of the application and the heterogeneity of the data encompassing that year. Additionally, those indicators were not considered critical for that application by the industry experts at that moment in time.

This case study was conducted in close collaboration with the industry experts depicted in Table I, which included the direct responsible for the correct performance of the "RED" application. The experts agreed the information showed by the prototype tool during the case study constituted a good reflection of the operational behaviour of "RED" during that period of time.

V. Conclusions and Further Work

This paper has developed the concept of operational

maturity in aerospace software applications, creating a reliable framework that can be integrated into an assessment tool. As a result, a successful lifecycle management system has been developed, that can be applied to assess large software application portfolios.

From this research, maturity has been identified as a very dynamic indicator, being a concept that offers an opportunity for improvement in the future. Using it as a baseline, the model could be linked and work together with the concepts of software obsolescence and complexity.

Future possible enhancements of this research could be developed by:

- Applying the model in companies from different sectors, validating in this way the model in a cross industry environment.
- Developing a standard methodology to gather data from software application portfolios, linking it with the model data input requirements.
- Studying the relationship between software obsolescence and operational maturity.

ACKNOWLEDGMENT

This research project is funded by Airbus and Cranfield University. The author would like to gratefully acknowledge the support and assistance of Airbus as well as the contribution of participants from several aerospace organisations during the research.

REFERENCES

[1] Caralli, R., Knight, M. and Montgomery, A., "Maturity Models 101: A Primer for Applying Maturity Models to Smart Grid Security, Resilience, and Interoperability", Software Engineering Institute, 2012.
[2] Becker, J., Knackstedt, R. and Pöppelbuβ, J., "Developing Maturity Models for IT Management – Aprocedure Model and its Application". Westfälische Wilhelms Universität Münster European Research Center, Münster, 2009.
[3] Paulk, M., Curtis, B., Chrissis, M. and Weber, C., "Capability maturity model for software", 1994 Version 1.1. http://www.sei.cmu.edu/pub/documents/ 93.reports/pdf/tR24.93.pdf. Accessed on 20/04/2017.
[4] Renken, J.," Developing an IS/ICT management capability maturity framework." In: Research conference of the South African Institute for Computer Scientists and Information Technologists (SAICSIT). Stellenbosch, 2004, 53–62.
[5] Humphrey, W., "Managing the Software Process", Goodreads, 1989.
[6] ISO/IEC 25010:2011, "Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models", 2011.
[7] ISO/IEC 9126-1, "Software engineering – product quality – Part 1: Quality Model", first ed.: 2001-06-15.
[8] McCall, J. and Matsumoto, M., "Software Quality Measurement Manual", General Electric Company, 1980.
[9] Lauesen, L., "Quality Factors", Software Requirements Styles and Techniques, Addison-Wesle y (ed), London, 2002, pp. 220-230.
[10] de Bruin, T., Rosemann, M., Freeze, R., Kulkarni, U., "Understanding the Main Phases of Developing a Maturity Assessment Model". In: 16th Australasian Conference on Information Systems (ACIS), 2005, Sydney, Australia.
[11] Tukey, J., "Exploratory Data Analysis", Addison-Wesley, 1977.