Evaluation criteria for application life cycle management systems in practice

József Klespitz*, Miklós Bíró**, Levente Kovács*
* Research and Innovation Center of Óbuda University, Physiological Controls Group,
Óbuda University, Budapest, Hungary
** Software Competence Center Hagenberg, Hagenberg, Austria
klespitz.jozsef@phd.uni-obuda.hu, miklos.biro@scch.at, kovacs.levente@nik.uni-obuda.hu

Abstract — **The effective use of (human) resources and a mature product assumes the application of a software development life cycle (SDLC) management system which actively supports the process of development. A well-adjusted SDLC system is a prerequisite in safety-critical developments, such as the development of medical devices. Otherwise, the intensive documentation needed proving the correct and safe operation of the equipment cannot be fulfilled. The aim of this paper is to provide structured and mostly quantified criteria for companies which plan to establish newly an application life cycle management system or improve an existing one. The answering of these question helps making an objective and optimal choice among the different systems. Finally, this paper prepares a case study, where the application of these criteria will be demonstrated in practice.**

I. Introduction

High quality software is required in many different application fields, such as in telecommunication, information technology, or in the financial sector. Yet, these developments are not bounded by as many directives and standards as the safety-critical software development [1, 2, 3]. The concerning directives may specify general requirements, such as IEC 61508 standard [4], which gives recommendation according to functional safety of electronic systems. Furthermore, different fields have their own specific directives from which the most relevant are the ISO 26262 for road vehicles [5], DO-178C for airborne systems [6] and IEC 62304 for medical devices [7]. The requirements in these standards have to be fulfilled, as the product cannot be lunched otherwise. The fulfilments of standards are examined by the entitled organizations.

In case of medical devices the Medical Device Directive (MDD, responsible for European market) and the US Food and Drug Administration (FDA, responsible for market of United States of America) are the responsible organizations. Both of them accept the IEC 62304 standard [7].

Finally, there are standards regulating a specific group of devices. In case of medical devices (the main scope of the current research) several other standards have to be complied as well [8]. These standards include the quality management standards ISO 13485 [9], the risk management standard ISO/IEC 14971 [10] or ISO/IEC 12207 standard [11] for software, and ISO/IEC 15288 standard [12] for systems. Finally, country specific regulations have to be mentioned, such as the standard in case of machines. The restrictions and requirements provided in these regulations means the most relevant difference between safety-critical and non-regulated software development.

The software development process has to be well documented in order to be able to prove the fulfilment of directives. These documents attest for authorities that every safety aspect is examined and handled, so the use of device is safe. Naturally, this workload raises the need to ease this commitment. As a result, *application life cycle management* systems (ALM) were created which effectively support the full development (Fig. 1.). Usually ALM systems are designed to support plan-driven software development, though this is not required by the standards [13, 17, 18]. The different ALM systems and their different versions accentuate different phases of a development. They support effectively more or less testing, requirement management, reviewing, problem solving, etc. The traceability is a vital aspect here. It is hard to create and maintain complete traceability. Although more and more attempts try to create advanced traceability techniques [14], ideal traceability system is still missing [15]. Moreover, ALMs try to support the development process itself, so they usually have workflow management features. The different approaches and the various solutions make difficult the choice among different systems, which raise the need for an optimally chosen requirement system.

The paper is structured as follows: Chapter II presents the environment, where the mentioned aspects can be used. Afterward, a quantitative requirement system is presented for evaluation. Conclusion is listed at the end of the paper together with the planned case study.

II. Environment of Application

The current research work examines ALM systems from the viewpoint of a small or medium enterprise with expertise in medical devices. The requirements can be used to analyze a complete ALM system itself, or to analyze a group of tools which provides together an ALM system. This makes possible to handle customized combination of tools or complete third party systems as well.

Although examination focuses on development of programmable medical devices, the requirements can be used for any safety-critical software development. However, answering some questions may require previous experience in operation of ALM systems, but most of the aspects can be evaluated without experience.

978-1-5090-2380-6/16/$31.00 ©2016 IEEE

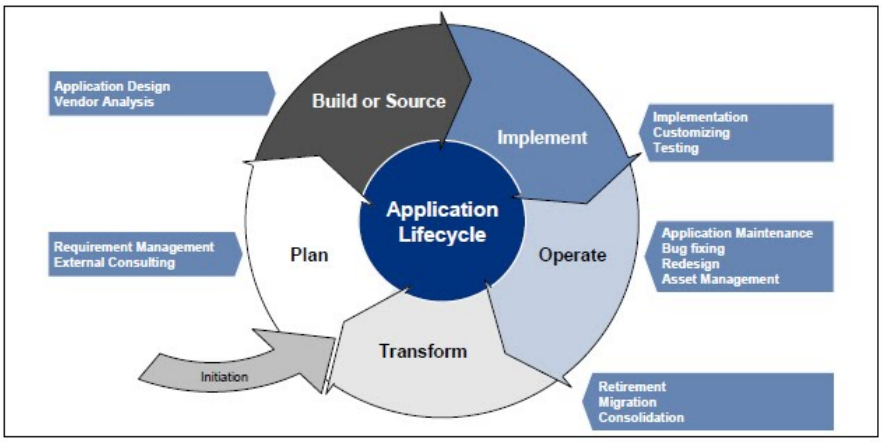

Figure 1. Application Lifecycle Management (ALM) [27]

This paper is the sequel of [16]. However, the viewpoints are collected here together in a manner to be able to compare the different aspects financially and help the management making a systematic and objective choice among the numerous possibilities.

III. Aspects of Comparison

This selection of choices was created by analyzing various ALM system features and by collecting knowledge from experts and relevant people. The different aspects are grouped here according to their benefits. Four main groups are distinguished with possible overlapping situations.

A. Actual costs

The first group is formed from aspects, where an exact value is assigned, so the price is known and defined. Here the license prices can be compared (both for standalone systems and for combined solutions with different vendors) and the price for the individuals has to be calculated ('human' users), just as for the automations ('machine' licenses). The cost-benefit analysis has to be done with individual and floating licenses as well. Furthermore, if company policy makes possible the price of having and maintaining an own server has to be compared with the storage services usually provided by manufacturers.

B. Costs of savings

In the second group price still can be exactly defined individually for companies. Hereby, the human effort can be specified (man-hour or man-month) and the real costs can be calculated from this. The first, straightforward point is the setup of the system: how much effort does it need to install the server, clients, etc. When changing an existing ALM system the introduction of a new system means extra cost. If own data server is used then the maintenance and setup costs have to be defined here in the same manner. Moreover, the cost of migration has to be considered, keeping in mind that restructuring might be necessary to utilize the most of the system. These efforts can be defined easily. On the other hand, there are factors which are still important yet not so exact. The first aspect here is to learn using the new system. The better usability means less effort needed by the user learning its use and operating it. Ideally, the newly set up system has interfaces which is familiar for the developers (e.g. Word-like text editor) and it stores and visualize information in similar windows with similar structure (or in a more reasonable way). Furthermore, it has to be analyzed what are the possibilities to reduce human efforts: using templates or creating smart algorithms makes the everyday work smoother and more effective.

C. Savings by usability and maintenance

The cost and benefit among the aspects in the two previous groups can be calculated. However, there are other features which reduce the workload or make the development more effective. Hereby, the usability related features are collected which are important and expected from an ALM system.

One of the most important aspects is to have an effective communication channel. It must be easily editable, easy to follow and easy to trace. This is especially important to solve issues, but plays important role in other phases of the development as well. The performance of the different ALM system has to be checked with different configurations. The performance could be estimated from the system requirements. However, if there is possibility, the actual performance of the systems has to be tested with realistic amount of data. The editorial interface has to be as suitable as possible.

As an example, the requirements only need a simple text editor (with Word like operation as mentioned before), while the requirements can be treated more practically via spreadsheets (where Excel like operation could provide familiar interface). Again, the ALM system has to solve the development at the highest possible level. As more people collaborate on a certain development it is inevitable to avoid conflicts and parallel editing. Furthermore, everyone has to be up-to-date. Therefore, a system is needed, where the real-time access is granted, where the fields can be parallel edited, while in case of conflicts the software provides effective help. The general usability has to be suitable as well. The user interface has to be well structured, the functionalities have to be easily reachable and the navigation must be simple. The system has to handle user friendly switching between projects and/or repositories, and it should be important to reach the development environment (where coding is done) with minimal effort.

Usually a user does not require all of the information what is stored in an ALM system. The system has to be capable to limit the presented information according to the privileges and interest of the user. This is usually done via

filters, which has to be configurable both for administrators and users as well. This is not only important to access only the necessary information, but it can be used for automations as well. When creating reports it is easier to extract the necessary information from the system and create documents automatically by concatenating the required points (selected via configurable filters). However, the system has to support change management: not only the company management has to know the readiness of development, but the users has to be able to check modification and the have to know the occurrence and reason of changes (requirement change, bug fix, etc.) as well. This assumes the correct tracing and keeping history logs. Moreover, the task management is not only important for the managers to distribute tasks among developers, but it is important for the developers as well, to know the number of their tasks, the probable required time to solve it and the importance of the certain tasks. Assuming reasonable working, this provides the optimal time development.

Along with the strength of the systems their weaknesses has to be found as well. There are certain features which might seem practical, but altogether hinder the development process. For example, in case of using the client of the ALM system via a web browser, it requires the use of timeout to minimize the number of inactive users and to prevent unpermitted access. However, the timeout could make impossible the running of complex automated scripts, might disturb the workflow (by dropping out the user at improper time without saving) and it even bother the users in their everyday work. Such weaknesses have to be collected and considered in the development process, and their cost and benefit has to be evaluated.

D. Indirect benefits

Finally, the fourth group can be created from the potential useful improvements what the new ALM system might provide. The features of the third group are required and accessible at start (or with minimal effort) and they are required for the effective use. The possible improvements require human workload to be implemented (more investment for greater benefits) and their existence is not straightforward. Here the first aspect is handling different version of the product under development and support product line engineering [19]. Among many other factors this requires the possibility to combine and modify certain (more) artefacts at the same time. Not only the simultaneous modification of certain fields is required here, but documentation branching (and version handling of), and workflow management handling flexibly as well. Even it might generate automatically workflows according to the modifications in the different products.

Certain ALM systems support special software languages which might be practical to benefit from. For example MATLAB-Simulink is capable to work with the Microsoft Office product family (Microsoft Word and Microsoft Excel) or with the IBM Rational DOORS. Here, they provide the possibility to link Simulink objects to the requirements instead of linking the generated code, which is a huge advantage: the generated code does not need any additional understanding and operation explanation via a block diagram (Simulink objects).

Similarly, if the ALM system supports certain other software (development environments, version handling systems, etc.) it is wise to think of using supported tools to further enhance the potential and effectiveness. Nowadays many of them are capable making certain automatic checks. The introduction has to be analyzed on such automations: the import of external test results (such as code analyzers) or scheduling external testing processes might mean huge benefit. Furthermore, external checks can be done as well for existing artefacts. Hereby, the most important feature would be the checking and fixing of *traceability and consistency*.

E. Importance of traceability

The complete traceability is required by various standards, but since Automotive SPICE 2015 [20], the complete consistency is required in the automotive industry (and later probably in the other industries) as well. This means that relationship between the related artefacts has to exist and the occurring changes has to update these connections. There are trials to check these connections automatically and to do automatic updates, but there is no generally accepted method or tool [21-25]. In case of analyzing traceability not only the existence of links are necessary, but also to check every link which necessarily exists and connects the correct artefacts as well.

The traceability can be extended until code level being the goal. Theoretically, the life of every line of code has to be known: why the code line was created, why it was modified, etc. In practice it is almost impossible to achieve this, but it can be approximated. As already mentioned in the previous chapters, if a standalone ALM system is used from a single manufacturer even in the case when more tools are used as coding, version handling and other tools, they are not integrated into ALM systems. However, the switching between these tools are time consuming and not ergonomic. To solve this problem the used tools could be connected together and could be reached from a certain interface. This also helps the creation and maintenance of traceability as well. To solve this, different approaches might be followed, but here we wish to highlight the Open Services for Lifecycle Collaboration (OSLC), which is widely accepted by the manufacturers of ALM systems and it is an open source standard [26]. The combination of tools cannot be negotiated and the further enhancement should be considered.

F. Ground for refusal

There is a fifth group of aspects which does not have a financial value, but affect highly the cost of operation of an ALM system. Namely, there are certain rules and policies for each company which cannot be negotiated. These policies might exclude certain systems from analysis.

Hereby, we wish to show some example: the security policy is always extremely important for companies and this might limit the choice. If the company has an own IT supporter and they ensure the security then the database server cannot be rented from the cloud or order external server provided by the manufacturer. For the same reason the choice of client might be limited; the access could be limited to the intranet of the company even if the client could be accessed via internet connection. Furthermore, the chosen tool has to have proper authentication system and limit the access according to the rules of the company.

The ALM system has to support the chosen development model and be able to handle flexibly enough not to hinder the development. Every company has its own custom properties, which have to be supported by the system. The creation of custom properties is common, but to use these properties later (for filtering as an example) is not trivial.

Finally, the system which cannot support the base lining and saving certain states of the development process will be probably excluded from such analysis as these are common needs.

IV. Conclusion and Further Work

To prove the safe and correct operation of a software, it requires many documents. For safety critical development this is enhanced by meeting the standards and directives. Altogether, this documentation burden has to be created with as low human effort as possible. For this reason, to support the development, ALM systems are introduced. This paper overviewed the information how to choose among the possibilities. By following the mentioned features improper ALM systems can be excluded from analysis. Moreover, it will provide information about the expected costs of introduction (or system change). The aim of the paper is to select a suitable ALM system and to try solving problems which will improve existing ALM systems. Tool connection and traceability analysis is in our focus.

In the future we wish to use a chosen ALM system to connect with other development tools (especially with development environment and version handling systems) and expand the traceability until code level.

Acknowledgment

The authors are grateful for the support of Research and Innovation Center of Óbuda University.

The research reported in this paper has been supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Science, Research and Economy, and the Province of Upper Austria in the frame of the COMET center SCCH.

References

[1] Cooke-Davies, Terence J.; Arzymanow, Andrew. "The maturity of project management in different industries: An investigation into variations between project management models", *International Journal of Project Management*, vol. 21:(6), pp. 471-478, 2013.

[2] Niazi, Mahmood; Wilson, David; Zowghi, Didar. "A maturity model for the implementation of software process improvement: an empirical study." *Journal of Systems and Software,* vol. 74:(2), pp. 155-172, 2005.

[3] Humphrey, Watts S. "Characterizing the software process: a maturity framework." *Software, IEEE,* vol. 5:(2), pp. 73-79, 1988.

[4] International Electrotechnical Commission. *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems.* IEC-61508:2010

[5] International Organization for Standardization. *Road vehicles – Functional safety.* ISO 26262:2011

[6] Radio Technical Commission for Aeronautics, European Organization for Civil Aviation Equipment. *Software Considerations in Airborne Systems and Equipment Certification.* RTCA/DO-178C, 2012.

[7] International Electrotechnical Commission. *Medical device software – Software life cycle processes.* IEC 62304:2006.

[8] Miklós Biró, "Functional Safety, Traceability, and Open Services." *Software Engineering from Research and Practice Perspective*, pp. 73-82, Scientific Papers of the Polish Information Processing Society Scientific Council, 2014.

[9] International Organization for Standardization. *Medical devices – Quality management systems.* ISO 13485:2003

[10] International Organization for Standardization. *Medical devices – Application of risk management to medical devices.* ISO 14971:2012

[11] International Organization for Standardization, International Electrotechnical Commission. *Systems and software engineering – Software life cycle processes.* ISO/IEC 12207:2008

[12] International Organization for Standardization International Electrotechnical Commission. *System and Software Engineering – System life cycle processes.* ISO/IEC 15288:2015

[13] Martin, McHugh; Fergal, McCaffery,. "Challenges Experienced by Medical Device Software Organizations while following a Plan-driven SDLC." *European Systems and Software Process Improvement and Innovation Conference EuroSPI*, Dundalk, Ireland, 2013.

[14] Glenn, A. Stout. "Requirements traceability and the effect on the system development lifecycle (SDLC)." *Systems Development Process Research Paper*, pp. 3-17, Spring Cluster, 2001.

[15] Orlena, Gotel; Jane, Cleland-Huang; Jane, Huffman Hayes; Andrea, Zisman; Alexander, Egyed; Paul, Grünbacher; Alex, Dekhtyar; Giulio, Antoniol; Jonathan, Maletic. "The grand challenge of traceability (v1.0)". *Software and Systems Traceability*, pp. 343-409., Springer London, 2012.

[16] Klepsitz, József; Bíró, Miklós; Kovács, Levente. "Aspects of improvement of software development lifecycle management." *16th IEEE International Symposium on Computational Intelligence and Informatics (CINTI)*, pp. 323-327, 2015.

[17] Sandeep, Chanda; Damien, Foggon. "Application Lifecycle Management." *Beginning ASP. NET 4.5 Databases,* Apress, pp. 235-249., 2013.

[18] David, Chapell. "Application lifecycle management as a business process." 2008.

[19] William, Frakes; Carol. Terry. "Software reuse: metrics and models." *ACM Computing Surveys (CSUR)*, vol. 28:(2), pp. 415-435., 1996.

[20] VDA QMC Working Group 13, "*Automotive SPICE® Process Assessment*", Version 3.0, Revision 470, 2015

[21] Padmalata, Nistala; Priyanka, Kumari. "Establishing content traceability for software applications: An approach based on structuring and tracking of configuration elements." *7th Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE)*, pp. 68-71., IEEE, 2013.

[22] Bouillon, Elke; Patrick, Mäder; Ilka, Philippow. "A survey on usage scenarios for requirements traceability in practice." *19th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*, vol. 7830 of LNCS, pp. 158-173, Springer Berlin, 2013.

[23] Patrick, Mäder; Orlena, Gotel;Ilka, Philippow. "Motivation matters in the traceability trenches." *17th IEEE International Requirements Engineering Conference (RE)*, pp. 143-148, IEEE, 2009.

[24] Michael C., Panis. "Successful deployment of requirements traceability in a commercial engineering organization...really." *18th IEEE International Requirements Engineering Conference (RE)*, pp. 303-307, IEEE, 2010.

[25] Jane, Cleland-Huang; Olly, Gotel; Jane, Huffman Hayes; Patrick, Mäder; Andrea, Zisman. "Software traceability: trends and future directions." *Proceedings of the on Future of Software Engineering,* ACM, pp. 55-69, 2014.

[26] Arthur G., Ryman; Le Hors, Arnaud; Speicher, Steve. "OSLC Resource Shape: A language for defining constraints on Linked Data." *Linked Data on the Web" (LDOW)* 996, 2013.

[27] Noise Consulting Group, "Managing the application lifecycle from inception to retirement", (last seen Oct. 2015.) http://noisetcd.com/tech-slns.html?alm