INCOSE Italian Chapter Conference on Systems Engineering (CIISE2014)
Rome, Italy, November 24 – 25, 2014

Product Architecture Management - an approach to Product Life Cycle

Gaetano Cutrona, Andrea Margini, Cesare Fantuzzi

Tetra Pak Packaging Solution / University of Modena and Reggio Emilia

gaetano.cutrona@unimore.it, andrea.margini@unimore.it,
cesare.fantuzzi@unimore.it

Abstract. Today a lot of small, medium and also large companies have a project organizational setup.
This means that the major part of the development activities are gathered thru the conduction of a project that at its completion will deliver a new version of the product.
In this situation managing versions and resources for some of the product components (subsystems) could be a problem or performed in a non efficient way.
This paper shows the approach applied in a company producing machines for the food industry.
The methodology is based on the application of PLC (Product Life Cycle) principles aiming at the rationalization of the decisions made during the planning, analysis and implementation phases.
The goal of the approach is to help designers and product architects to correlate the needs of projects stakeholders (requirements) with other needs related to the product strategy and roadmap in order to improve efficiency in terms of resource management, product variants and other aspects that affect the life cycle of the product.

Introduction

Complex products are integrating very different functionalities that are belonging to different technologies and disciplines and must coexist all together in order to achieve the common goal that is a successful product that brings value to the customer and to the company.
In order to achieve the above goal a structured company follows different processes to acquire the input needed from the customer, translate them into requirements that are satisfied by the implemented solution. This implies an organization structure that usually consists of projects (responsible for the product delivery) and a set of line organizations (responsible for the product development).
The methodology developed will help product (system/subsystem) owner or responsible to efficiently develop a roadmap that will optimize the resource allocation and the releases to the related projects.
As mentioned at the beginning of this paragraph, usually, a common operational model consists of a project entity (a team lead by a project manager) and one or more development lines (teams lead by line managers).

The project is responsible for the acquisition of the requirements, control of the planning and deliveries and on the other hand, the lines are responsible for the design of the solutions that will fulfil the requirements received by the project (Figure 1).

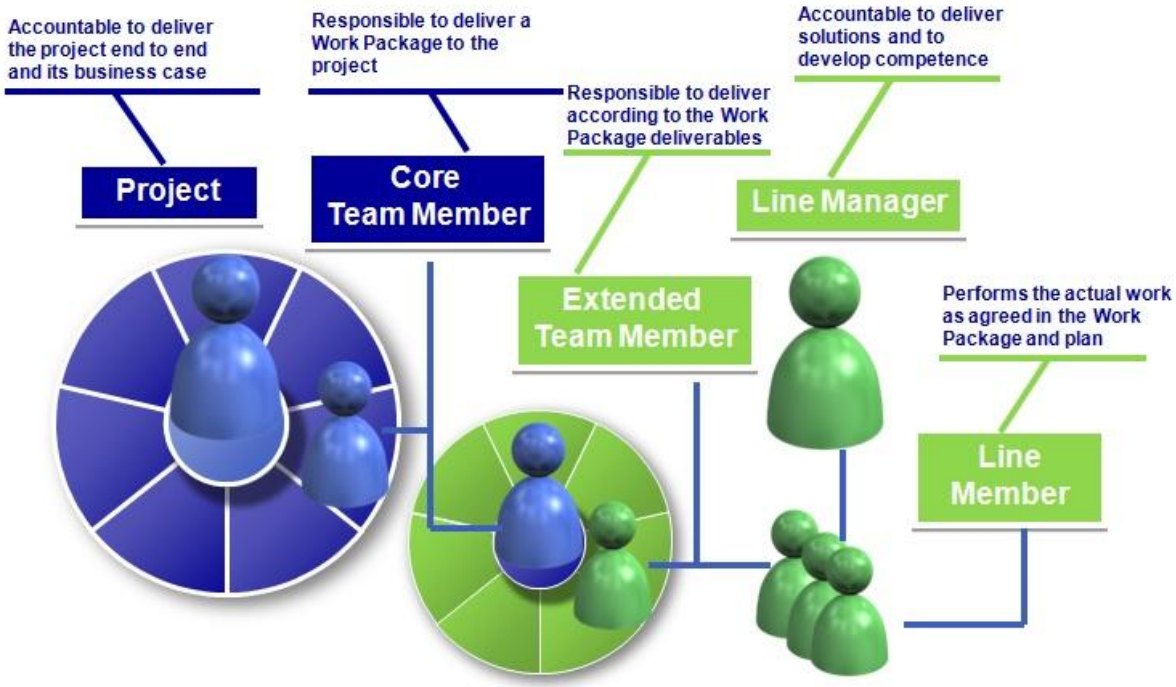


Figure 1 Project / Line Setup

The implication of such organization is that the increasing number of projects is not always increasing consequentially the number of resources available from the lines that have to develop the solutions. So a certain resource has to work in parallel in more than a project. In addition a line have to manage different sets of requirements belonging to different projects that sometimes are conflicting each other or causing reworking due to sequential update of certain functionalities in a product.

The detected resulting situation causes big problems in managing the versioning of a certain architecture of product (or component) increasing thus the effort in developments and configurations taken by the lines that are owning it.

To support lines in the management of a product a lot of methodologies are available in literature [2], [3], [5]. Most of them consist of prioritisation and classification of requirements performed by the company's marketing organizations or development governances. The outcomes support the decision of what features to include in a certain product.

The methodology described in this paper will improve the product architecture management by enabling the line to actively participate in the decisions regarding the solution evolution by directly owning its roadmap.

Next section explains the motivation behind such particular methodology.

Afterwards the benefits of its implementation are shown by practically describing the case study where this methodology was applied.

In the last section the methodology results are discussed and next steps for further improvements are briefly mentioned.

Motivation

An ideal process of product development and management starts with needs of customers. They are analyzed by the company market organization which then creates a set of stakeholder requirements. That list together with other internal stakeholder requirements will be satisfied by a project with the delivery of a solution. As already mentioned large companies, usually, are involved with several concurrent projects and produce different configurations of their products. The worst possible case of the above situation is when a development line delivers one version of its product per project. Assuming that each version may be considered as a stand-alone product, in case of optimized architecture management, every version is the result of the new features addition to the older ones (case n.1). In case of non-optimized architecture management there is the coexistence of more than one concurrent version in the same life cycle phase of the product (case n.2). A simple example (Figure 2) that explains the above situation is related to the product user manuals describing the features added to the different product versions. In case n.1 the line has to manage one user manual evolving together with the product versions along with the product life because the latest version is able to replace (includes) the older ones. Three different user manuals have to be generated and separately managed in case n.2 due to the partial dependencies between the different product versions.

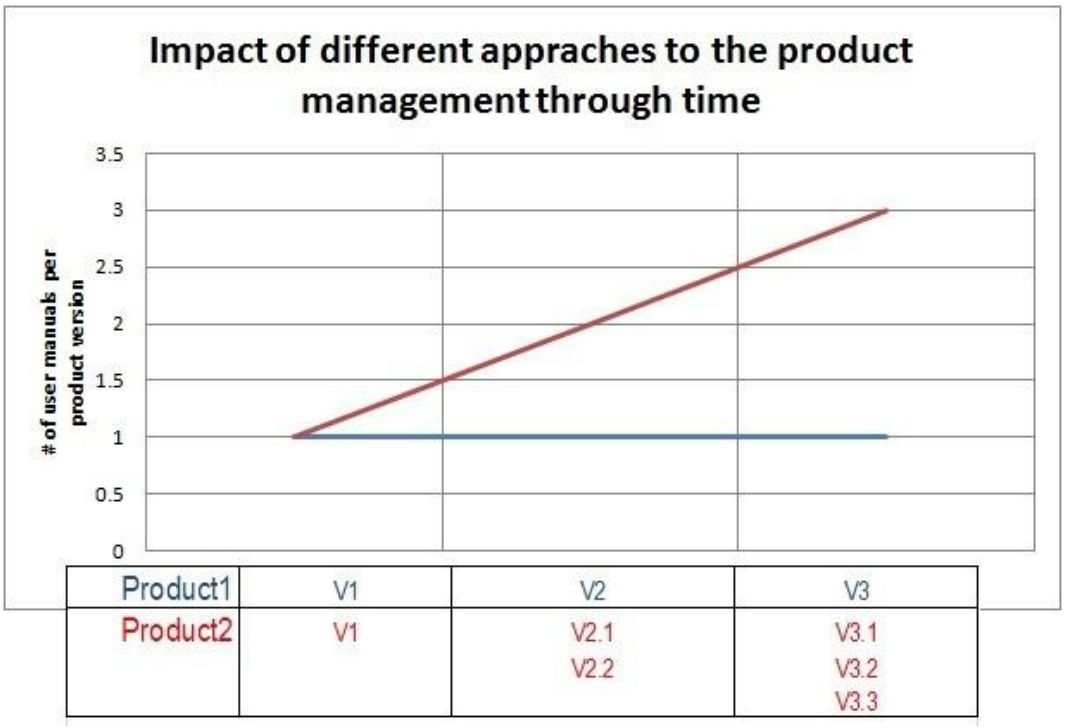


Figure 2 Example of number of user manuals per product versions in case of two different product management approaches

The intention of the example is to provide an indication of how much complex could be managing different "stand-alone" concurrent versions of the same product. The effort for such

management increases proportionally with the number of existing concurrent versions causing the need for the line to ask for more budget and resources to keep maintaining those solutions.

Case Study

The methodology described in this paper was applied in a real case study that involved the development of a mechatronic subsystem. The module was fitting in a packaging machine and both it and the complete product are in the maturity phase of their life cycles. Though the methodology is applicable to any life cycle phase, the maturity one is where it gives the most benefits because it is the phase were the risk to have more than one concurrent versions is higher.

In the starting situation a list of projects (and related requirements) that the subsystem had to fulfil was available. The module provided also some supporting documentation that described its design and related technical decisions. The initial module roadmap showed that a new version of the subsystem would have been released per each project.

The applied methodology steps are described in the following list:

- Requirements vs. Projects review
- Architectures definition
- Versions Definition

The first step rearranged requirements together with project in order to provide the full picture of what the module had to fulfil. In the Architecture definition task subsystem commonalities and variants were defined according to projects and packaging machine families where the module was applied. Last step defined the module roadmap by identifying its release versions (Figure 3).

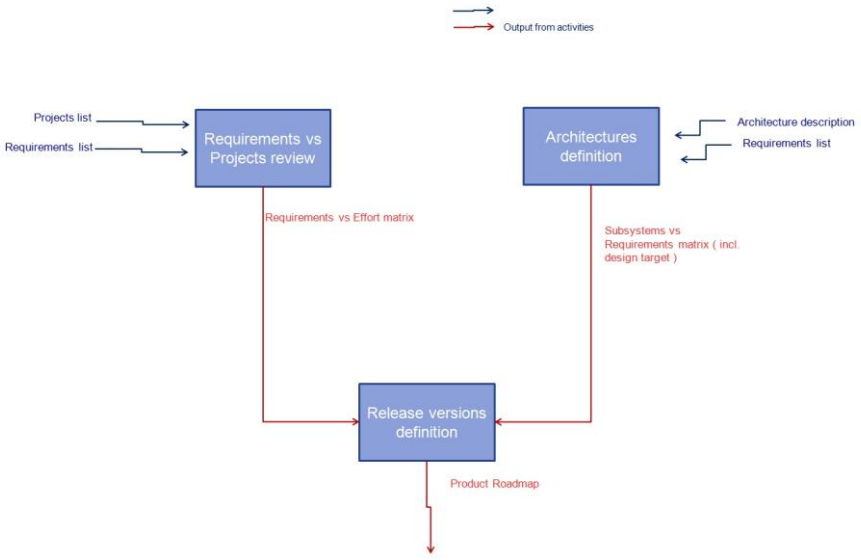
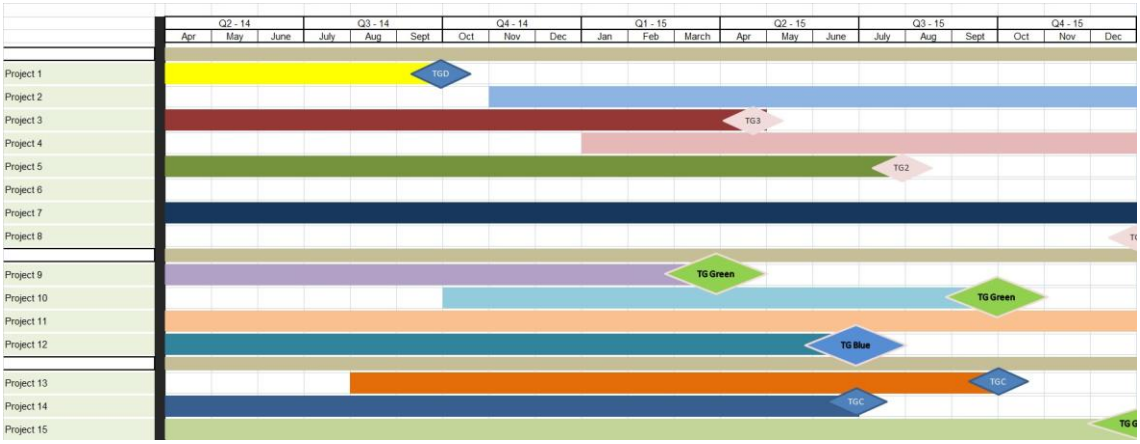


Figure 3 Methdology workflow with input and output

Inputs

In this section the inputs for the described case study activities will be described in detail. In particular to start with the activity the complete list of project (and related time-plans) was required. The list consisted of all the projects (both ongoing and future) that were involving the module providing in this way a three years view of the subsystem developments (Figure 4).



	Project 1	Project 2	project 3	Project 4	Project 5	Project 6	Project 7
Req 1 (m/s)	1.6	2.0					
Req 2 (p/h)			15000	15000	8000	8000	9000
Req 3	x	x					x
Req 4			x	x	x	x	
Req 5					x	x	
Req 6	x	x	x	x			
Req 7					x	x	x
Req 8						x	
Req 9		x		x	x	x	
Req 10		x		x	x	x	
Req 11	x	x	x	x	x	x	x

Figure 4 Extract of Project Roadmap and Requirements vs Projects matrix

Together with the project roadmap the full list of requirements per each project was requested as well. All the requirements were independent each other, identifying thus one feature each. No priority was defined yet to those requirements.

The product baseline that consisted of a functional breakdown of features that the product provided and a system breakdown structure showing how the system was built (which are the components of the system) were available. Some other architectural views (Architecture description, AD) were included in the baseline (Figure 5). In addition to the architectural documentation also a System Requirements Specification (SRS) was provided in order to keep track of the rationale behind the solution decisions.

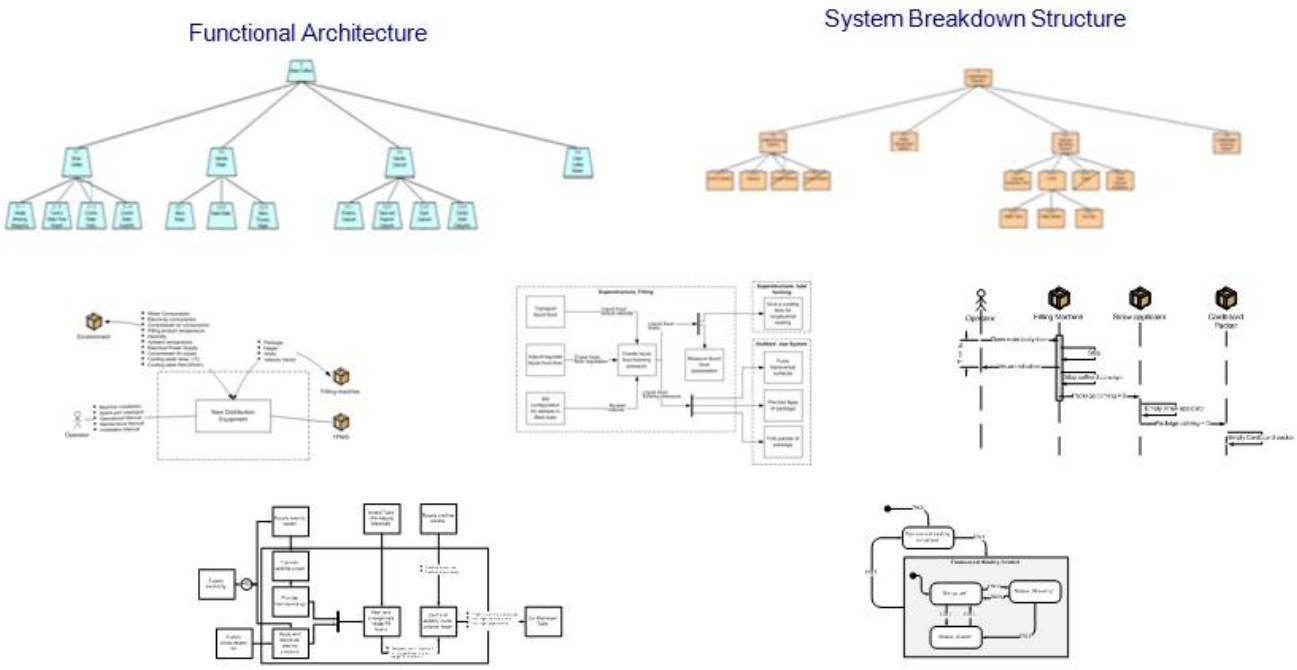


Figure 5 Architecture Description Documentation

In next paragraphs further details on how those inputs were used are described.

Requirements vs. Projects Review

On the reference baseline architecture an impact analysis of the requirements was performed. During this activity relevant people (technical experts) were involved in order to assess implications of product modifications in terms of effort and technical feasibility. The outcomes of the activity were recorded into a Requirements vs. Effort matrix that puts in relation requirements to the effort to fulfil them (time, cost and resource).

Architectures Definition

The available architecture baseline was rearranged in order to group functionalities / subsystems into three categories (Figure 6):

- Product generic were all those features that are needed by the product as a framework, they were in common to all the variants and configuration of a certain product baseline (e.g. the functionality to fill and seal a package).

- Application generic were the features that are applicable to a defined application of the product, they defined the functionalities of the product in a certain application (e.g. features in low cost machines).

- The Specific application features were all the characteristics that are requested to a product to work according a specific need (e.g. type of food to be packed or specific type of package to be produced).

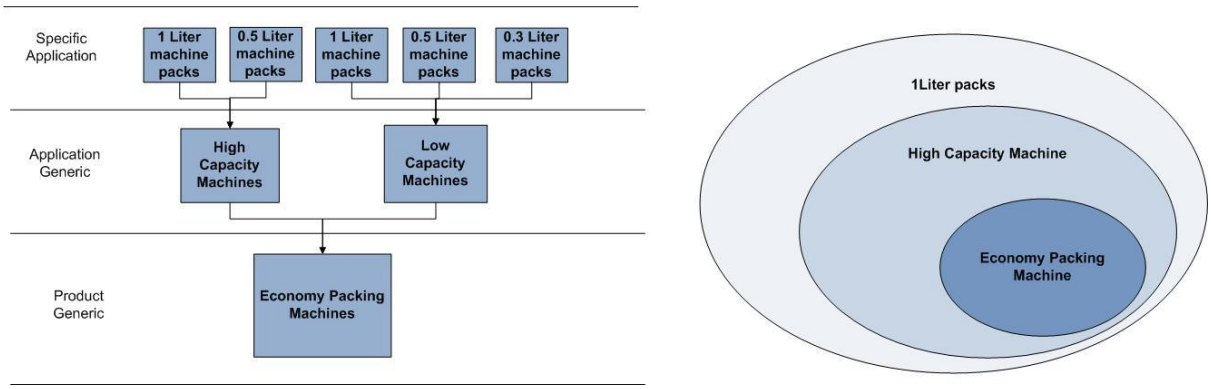


Figure 6 Example of Architecture Items categories

The same grouping approach was applied to the new requirements in order to gather the full picture of the new versions features grouped into categories. The outcome of this step was to update the subsystems vs. requirements matrix with consolidated targets. In order to take into consideration specific needs of the line owning the product subject of this case study, the following list of additional architectural drivers was developed:

- Maximize Commonalities: the more commonalities are present in terms of product components among the different product configurations the better is from a product management point of view (expand as much as possible the Product generic and Application generic part).

- Optimize Development and Management costs: optimization of the resources and costs during the development and life cycle management phases.

- Formalize PLC: a clear picture of the life cycle of the product (i.e. product roadmap, strategy, etc.) enables a faster and effective decision making process.

- Optimize Deliveries: an optimization of the number of product releases improves the product management process.

The above listed drivers were also used to measure the benefits derived from the application of this methodology on this case study.

Release Versions Definition

At this point of the process the project requirements were allocated to the different architecture categories creating a link with the relative projects. Considering the projects timelines a prioritization of subsystems developments activities was performed and a map of the subsystems releases was defined per each identified specific application. All the subsystems releases were grouped to system releases in order to fulfil the projects deadlines (Figure 7). The outcome of this final step was the product roadmap according to projects timeline (Figure 8).

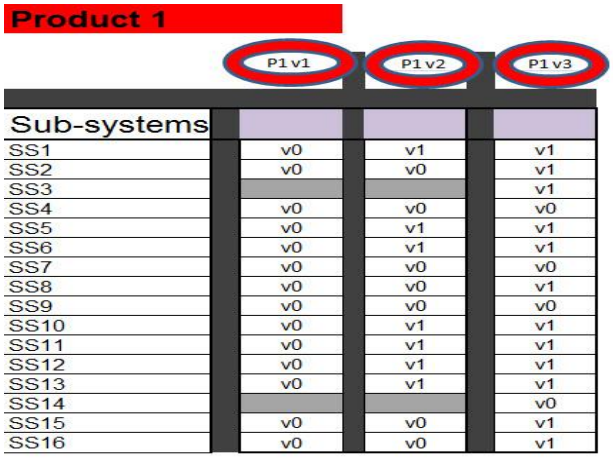


Figure 7 Example of Subsystems Versions within the Specific Application n.1 (Product 1)

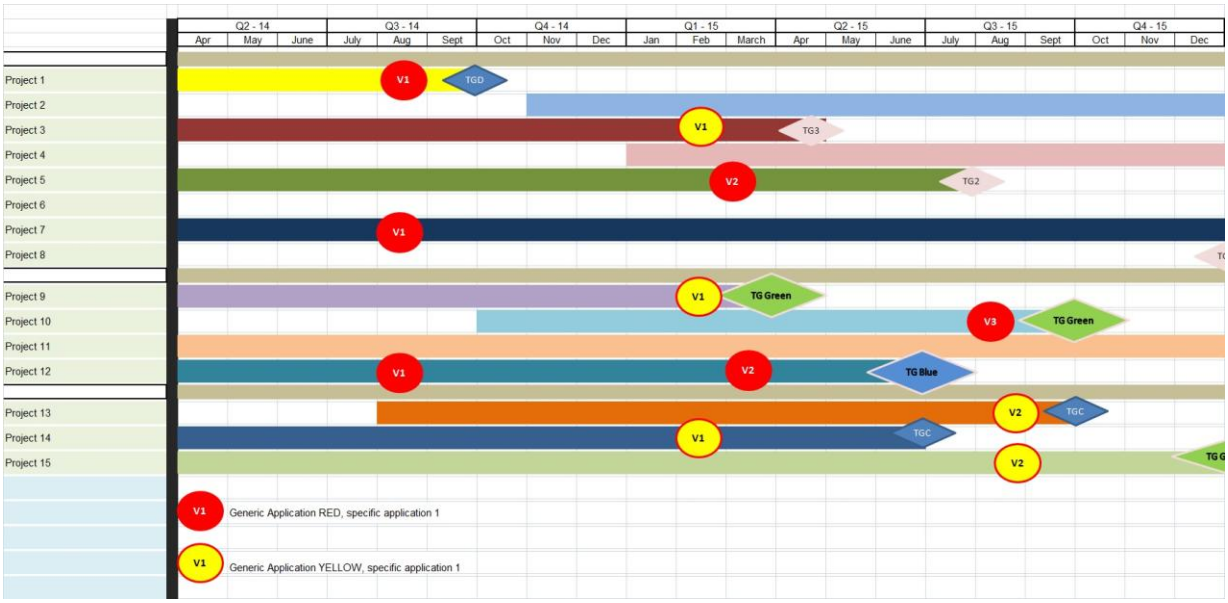


Figure 8 Example of resulting Product roadmap

Conclusion and Next steps

In order to assess the benefits of the proposed methodology, an estimation of how the product roadmap would have been, in accordance with the drivers, was performed. That roadmap was developed without the application of the process. It was characterized by the following relevant information:

- \item Number of releases for the projects = Number of projects = 15
- Total number of resources involved = Number of Projects running in parallel = max 13
- Number of Product versions to be managed = 8

The same exercise was performed after the application of the described methodology. The related results can be found in the following list:

- Number of releases for the projects (Specific Applications) = 7
- Total number of resources involved = 5
- Number of Product Architectures to be managed (Generic Applications) = 4

After the analysis a significant improvement in the short term product management was found. It is expected that it will affect the long term one as well. The following list represents the main improvements detected:

- Resource optimization
- System complexity reduction
- Simpler product life cycle management
- Improved communication between functional units in the company

While the first three points are largely explained during the paper, it is important to spend some words about the last item. It refers to the capability of the system/subsystem team to clearly communicate the development activities to the projects and proactively act in order to fulfill new requirements or changes.

This case study development heavily relied on the expertise and skill of the product experts especially in the requirements grouping and prioritization activities. Hence, in order to further improve the described methodology, it should be integrated with structured and formal techniques supporting the above mentioned requirements management activities.

References

[1] C. Haskins, ed., "Systems Engineering Hand-book". International Council on Systems Engineering, v. 3.2 ed., 2010.

[2] VDI, Design Methodology for Mechatronic Systems (VDI 2206). Berlin: Beuth Verlag, 2004.

[3] S. V. Vasi_c and L. P. Mihailo, "Standard Industrial Guideline for Mechatronic Product Design, FME Transactions, vol. 36, no. 3,pp. 103{108, 2008.

[4] A. Pysteer and D. Olwell, "The Guide to the Systems Engineering Body of Knowledge v. 1.1". Hoboken, NJ: The Trustees of the Stevens Institute of Technology, 2013.

[5] M. S. Hundal, "Systematic Mechanical Designing: A Cost and Management Perspective". ASME, 1997.

[6] R. A. Boggs, “The SDLC and Six Sigma. an Essay on Which is Which and Why?" Issues in Information Systems, vol. 1, no. 1, pp. 36-42,2004.

[7] INCOSE, “Systems Engineering Vision 2020," tech. rep., International Council on Systems Engineering Seattle, USA, 2007.

[8] J. A. Estefan, "Survey of Model-Based Systems Engineering (MBSE) Methodologies, Rev. B, in International Council on Systems Engineering, (San Diego (CAL)), INCOSE, INCOSE Technical. Publication, Document No.: INCOSE-TD-2007-003-01, 10, June 2008.

[9] W. Schfer and H. Wehrheim, “The challenges of building advanced mechatronic systems.," in Future of Software Engineering, pp. 72-84, 2007.

[10] Boston Consulting Group, “Innovation 2006,"

[11] R. Kemp, M. Folkeringa, J. de Jong, and E. Wubben, "Innovation and Firm Performance". 2003.

[12] H. Loof and A. Heshmati, “On the relationship between innovation and performance: A sensitivity analysis," Economics of Innovation and New Technology, vol. 15, no. 4-5, pp. 317-344,2006.

[13] D. Dalcher, O. Benediktsson, and H. Thorbergsson, “Development life cycle management: a multiproject experiment," in Engineering of Computer-Based Systems, 2005. ECBS '05. 12th IEEE International Conference and Workshops on the, pp. 289-296, 2005.

[14] M. Chechik and J. Gannon, \Automatic analysis of consistency between requirements and designs," Software Engineering, IEEE Transactions on, vol. 27, no. 7, pp. 651-672, 2001.

[15] R. Harrison, A. West, and L. Lee, \Lifecycle engineering of future automation systems in the automotive powertrain sector," in Industrial Informatics, 2006 IEEE International Conference on, pp. 305{310, 2006.