DESIGN FOR SIX SIGMA APPLIED ON SOFTWARE DEVELOPMENT PROJECTS FROM AUTOMOTIVE INDUSTRY

Sergiu NICOLAESCU[1], Claudiu Vasile KIFOR[1] and Lucian LOBONȚ[1]

ABSTRACT: As products are constantly evolving, most projects resume all stages of life even if much is carried over, and the gain is greater if the errors are identified in early stages of development. Design for Six Sigma is a structured method for developing new products, aligned with the needs of customers, which is involved during the early stages of product development. Various techniques are used within the methodology like anticipation of potential problems / needs, identification of current problem / needs, predictive engineering and management of actions used to improve product design. This paper examines the role of improvement and application of Design for Six Sigma methodology in the automotive industry and presents a case study that can become a guide for Define, Measure, Analyze, Design, Verify (DMADV) applicability in software development projects from research and development centers of industry. Approaches are proposed to improve the software development process focusing on the customer, capitalization and management of knowledge in the project and quality of product.

KEY WORDS: Quality, Improvement, Design, Six Sigma, DMADV

1 INTRODUCTION

Six Sigma methodology was developed by Motorola in the mid 80s, and has been made well known in 1995 when Jack Welch from GE company used it as the key factor of business strategy (Maass, E. McNair, P., 2010). It was "responsible" for saving companies billions of dollars in the early 1990s (Pyzdek, T., Keller, P., 2010). The system has since been constantly increasing, and can be felt in the today's industry that is the business management system with the fastest spreading.

Although the savings for companies were big, they discovered that only by improving existing processes and products to reduce defects, the desire of customer for higher quality cannot be stopped; in response to customers' desire for quality and the initiative to achieve Six Sigma quality level of 3.4 defects per million products was proposed to integrate Six Sigma methodology in designing of new products, using Design for Six Sigma.

Design for Six Sigma is a subset of Six Sigma that is focusing on preventing problems and not just repairing them; it is also known by the name of DFSS, which comes from "Design For Six Sigma". Methodology shares many principles with classical Six Sigma method but DFSS go further on recognizing that decisions made during the design phase largely affects the quality and cost on following activities of building and delivery of the product.

[1] "Lucian Blaga" University of Sibiu, 10, Victoriei Blvd, Sibiu, 550024, România;
E-mail:sergiu.nicolaescu@ulbsibiu.ro;
claudiu.kifor@ulbsibiu.ro, lucian.lobont@ulbsibiu.ro

The core objective of Design For Six Sigma methodology is designing correct from the first time in order to avoid limitations of functions that are or just possible to be in requirements / needs of customers; on the other hand, the principle of DFSS is to minimize variation (Shenvi A., 2010). Quality Level "Six Sigma" in the context of DFSS can be represented by the vulnerabilities that are minimal or are not present into product design.

For implementation of DFSS methodology have been proposed and used many process, as: Identify-Define-Develop-Optimize-Verify, IDDOV (Cudney E. , Furterer S., 2012) and Concept-Design-Optimize-Verify, CDOV that implies identification phase or concept, define phase, design phase, optimization and verification phases; RADIOV (Maass, E., McNair, P., 2010), composed by six phases focused on requirements, architecture, design, integration, optimization and verification and also gives the name of the methodology; DMADV which will be detailed in this paper. Although the processes use different approaches, most of the steps, tools and methods are the same.

Design for Six Sigma DMADV consists of five phases: definition, measurement, analysis, design and verification that design engineers must follow. Each of these phases have assigned methods and tools that help in designing the product / process and ensure that the design is according to customer exigency, to increase his satisfaction when using the product.

Formula that defines the methodology Design for Six Sigma is the following:

$Y = f(x)$ Or $Y = f(X_1 + X_2 + ... + X_n)$.

Where Y is a quantitative representation of the customer requirements and f(x) are inputs that contribute to the outcome Y.

2 DMADV AND DMAIC METHODS IN PRODUCT LIFE CYCLE

To be more clear on how Six Sigma methods are applied there must be identified the field of application into the product life cycle and also the way it is managed. We refer to product lifecycle management in development (PLM) that is different from marketing lifecycle management. The first seeks engineering aspects of the product, from its concept stage and continuing to integrate it into production, service and removal from the market. Marketing approach relates to the commercial management of the product life on market, focusing on measuring sales and costs.

Classic life cycle of a product is divided into five main phases: research, design, development, production and service. Depending on the product or industry order of events, activities may vary but the main processes are the same.

In the diagram below (figure 1) can be seen how classic Six Sigma method DMAIC and DMADV are applied within the product life cycle.

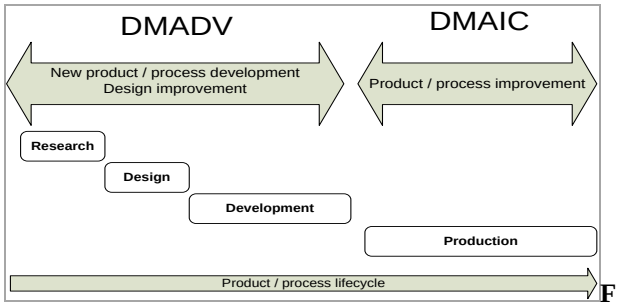

Figure 1. DMADV and DMAIC methodologies in product lifecycle

3 CASE STUDY. DESIGN FOR SIX SIGMA DMADV METHODOLOGY APPLIED ON SOFTWARE DEVELOPMENT PROCESS FROM AUTOMOTIVE INDUSTRY

DFSS methodology helps to identify problems in the product development but the real value is given be the prevention of potential problems, so it is suitable on using in software development processes where the area of interpretation is much higher than in production processes.

3.1 Mathematical approach

The formula that defines the software development process on which Design for Six Sigma methodology is applied is presented below:

$$Y_i = f_i(x_{11},x_{21}\ldots x_{n1},\alpha_1,\beta_1,t_i)+f_i(x_{12},x_{22}\ldots x_{n2},\alpha_2,\beta_2,t_i)+\ldots+f_i(x_{1m},x_{2m}\ldots x_{nm},\alpha_m,\beta_m,t_i),\ i=\{1\ldots n\}$$

$Y_=$ { Y_T, accomplished customer requirement; Y_F, not accomplished customer requirements }

Where:

- Y is the resulted software package,
- $x_{11},x_{21}\ldots x_{n1}$ are customer requirements planned for software package number 1,
- $x_{12},x_{22}\ldots x_{n2}$ are customer requirements planned for software package number 2,
  …
- $x_{1n},x_{2n}\ldots x_{nn}$ are customer requirements planned for software package number m,
- $\alpha_1 \ldots \alpha_m$ are human resources used during development of software package (1... m),
- $\beta_1 \ldots \beta_m$ are equipments / tools used for development of software packages (1... m),
- t_i is time of software product development,
- f() is the process of input transformation used in software development,

Process improvement purpose:

Y_F -> min (number of not accomplished requirement inside software must be minimum)

$t_f = t_c - t_c*0,1$ (time of development must decrease with 10%), t_f is the time of development after Design for Six Sigma is applied, t_c is the current development time

3.2 V-Model Process

The process used on projects in which the study was performed is based on the V-Model, a process model for software development, incremental, which has similarities with other software development mode, named "Waterfall". Instead of falling in a linear manner from one stage to another like in the latter model, in V-Model the stages are "mirrored" on the implementation phase, forming a V that is used to highlight the links between stages.

The original purpose of the model was to use it as a standard development model for projects on information technology (IT) in Germany, and was adapted for innovation in IT only in 1997 (Basem H., 2010).

The result of each stage of the model is a product, a product that must be verified and approved until jumping to the next step; this

precondition is ensuring the quality of product and help in planning and monitoring of activities.

The process starts with defining the project which includes the creation of a concept, writing of functional and non-functional requirements from customer needs, creation of product architecture at static and dynamic level, writing of detailed design. Once the project is defined, reviewed and approved the actual implementation is started. The final part of the process is the testing phases of the project that includes testing at module level, software integration, verification of requirements, maintenance and observation of operation products (Figure 2).

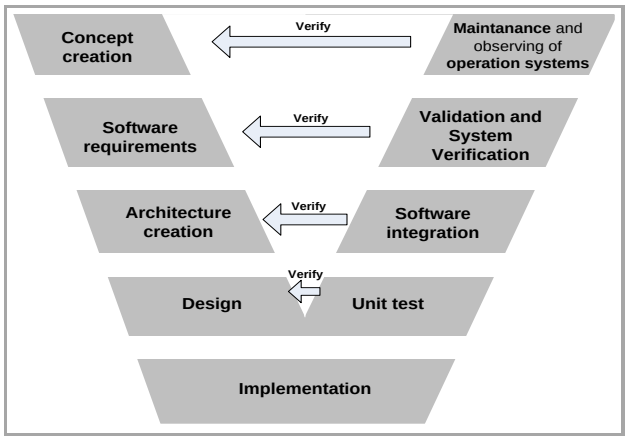

Figure 2. V-Model Diagram

3.3 Define phase

As result of a discussion with management and software team members from a big organization of automotive industry, the improvement objectives that are necessary in order to satisfy the internal and external customers were identified and the methodology Design for Six Sigma DMADV was selected as framework.

It is important to note that Six Sigma methodology is not applied in this study on the V-Model process but on the process form that is used in software development projects after customization; one of the particularities of the V-Model is that can be adapted to the needs of the projects.

Product cycle design starts with a clear definition of the project, including product or process that will be improved, purpose of the project, project planning, resources and deliverables, similar to a project management plan. Change Management Plan is not excluded, it aims to identify which documents or parts of the organization will be affected by change, to what degree and how are resistant to change. The phase includes creation of a risk management plan where the known and predictable risks from project are identified, such as human (the team with few experience in Six Sigma projects); technical (complex design); planning (dependent actions with short time limit); business (cost increase). Risks are identified, analyzed, a level of severity and actions are assigned in order to stop or minimize the risk. Below is the project charter created for application of Design for Six Sigma methodology.

<table>
<tr><td colspan="2">Project Charter</td></tr>
<tr><td colspan="2">Project name: Design for Six Sigma implemented into software development process.</td></tr>
<tr><td colspan="2">Opportunity/ Problem description:
- Time of development is too long.
- It's hard to involve all team members throughout the activities of the process.
- Difficult in case many changes are requested lately in development.
- Number of not accomplished requirements must be decreased for each release</td></tr>
<tr><td colspan="2">Business Case/Impact:
- Number of projects should increase and time of development should decrease in order to face competition.
- Quality of projects must be higher.</td></tr>
<tr><td>Team members:
SW Project Manager
SW Developers
SW Integrator
SW Validator
SW Quality Engineer</td><td>Project responsible:
Nicolaescu Sergiu
Stakeholders:
Internal customers (software development, test, system and quality team members)
External customers (Product customers)</td></tr>
<tr><td>Project performance indicators
- Reduction of development time with at least 10%.
- Usage of project members shall be handled in an efficient way, more than 80%.
- Reduction of errors identified by customer with at least 10%.
- A new management of change method should be integrated in order to handle changes from customer demanded into late phase of project.</td><td>Objectives
- Creation of an overview with necessary time on every phase from the project, depending on complexity of changes.
- Building a system for tracking the resources, for their effective use.
- Identify actions to reduce errors from software packages.
- Verification of the project in details.
- Storage of knowledge gained from the project in order to use it in further development of projects.</td></tr>
</table>

Limitations and purpose of the project

The goal is to create an efficient process for development of software packages that reduce the time of development, the number of unfulfilled requirements and can adapt to customer needs.

The limitations are the following: only resources from inside the team can be used, stay within budget and be conforming to quality standards such as SPICE software, ISO26262.

Risks	
Severity	**Description**
4	Technical – Complex design
4	Human – Too small team
3	Planning – Six Sigma project does not fit the planned timing.
5	Technical – Process will not be mapped on quality standards like SPICE, ISO26262, TS).

Critical success factors	
Severity	**Description**
3	Convincing the engineers that improvement is needed, to achieve a high degree of involvement.
5	Quality must be maintained after changes are done.
4	Process verification should be done carefully, to assure that there are no looses and the gain is real.

Critical factor for success

- Investing time to identify time consuming activities that can be automated.
- The changes made during designing the process should have impact on customer satisfaction and on development time reduction but the changes and risks must be low.
- Verification of new developed process during software development on 1-3 projects, depending on the results.

Major events		
Item	**Description**	**Data**
1	Define – Purpose, planning and risks are defined; final documentation.	**1.Sep.14**
2	Measurement – It is created an overview on the necessary steps and is determined the effort on every step.	**15.Sep.14**
3	Analyze – It is performed a research of steps that can be improved.	**29.Sep.14**
4	Design – Architecture and implementation of new software development process is performed.	**17.Oct.14**
5	Verification – It is examined the designed development process in 1-4 project, depending on the results.	**1.Dec.14**
6	Documentation of knowledge - Data, information, and relevant knowledge for the project are stored for reusing.	**4.Dec.15**

In the definition phase of the project should be identified internal and external customers and also their needs and wishes, which can be converted into measurable requirements; the used technique is "voice of the customer". The term "voice of the customer" describes all customer requirements, both explicit and implicit. Internal customers are the project responsible, development engineers, test engineers, quality engineers, system engineers, manufacturing engineers and external customers are those who get the product developed. Using the following techniques: brainstorming, interview and also analyzing the management system of the quality issues, were extracted the requirements which are presented in a KL diagram (figure 3).

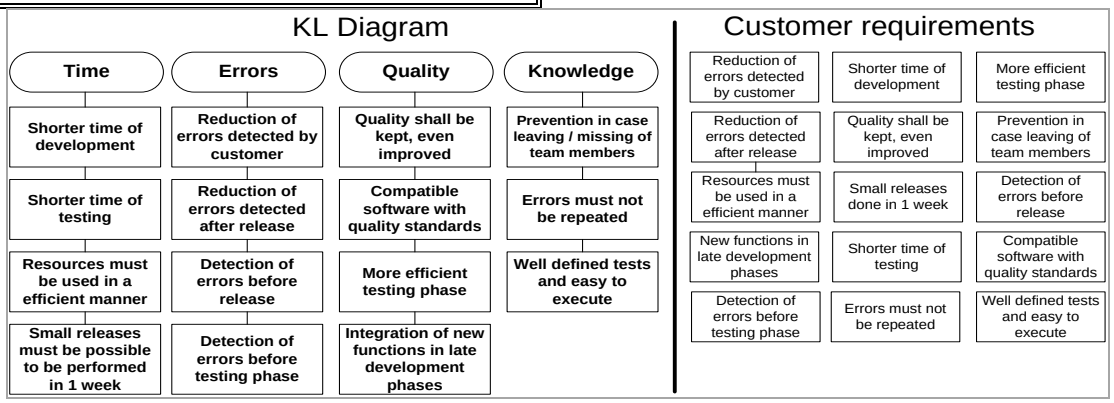

Figure 3. KL Diagram on software development customer requirements

Using these results as input data designing can be started and a product that conforms to customer's expectations can be offered. May be granted priorities for these requirements by using KL diagram correlated with KANO model, but in this case it was decided to continue the analysis with implementation of all requirements.

3.4 Measurement phase

In the step of measuring the time & effort were checked for each stage of software development within a release and also the numbers of errors discovered by the customer. Estimations were made according to the complexity of the package to be implemented.

The development time of a project is usually between 1 and 3 years, depending on its complexity, and each project has established early in development the major milestones and time schedule in which the software packages appropriate for the planned maturity will be delivered.

For example, the project which the process was implemented lasted 1 year and 6 months, was divided into 7 packages of release, a release lasting on average 38 working days.

Each release is divided into 9 stages: planning software, software requirements, software design, software implementation, "freeze" software, software testing at module level (software unit test), integration testing, testing requirements, preparation of release.

Planning software. Here are written and updated the documents such as the project plan, project planning, the management plan of changings, risk management plan, test plan. Measured length (depending on complexity release): 2-4 days.

Software Requirements. Here are written the requirements for which responsible is the software discipline and are created links between general requirements and sources. Measured length (depending on complexity release): 2-5 days.

Design software / software implementation. Is created / updated the design of the project and the code is written. Measured length (depending on complexity release): 5-20 days.

"Freeze" software. A version of the software is created to be tested, a version on which no changes are made. Duration measured: 1/2 days.

Testing software at module level. At this stage software design is verified. Measured length (depending on complexity release): 3-6 days.

Integration testing. Verification of the architecture is performed during the mentioned stage. Measured duration: 6 days of full testing / 2 days of regression testing.

Requirements testing. At this stage is checked the fulfillment of the software requirements. Measured duration: 9 days of full testing / 4 days of regression testing.

Preparation release. Here are created software packages for customer, production, including documentation. Measured duration: 3 days.

Were measured the errors found after the software package freezing step, discovered during testing at the system level or at the customer. To create an overview, the media was made between data taken from 3 projects, 10 software packages. The number of errors found during testing was on average 6 on each software package, generally based on the project maturity the number decreases; the average number of errors reported by the customer is 2. SIPOC tool is commonly used in Six Sigma projects to create a visual image of the workflow in product development. During the creation are identified suppliers and inputs from them.

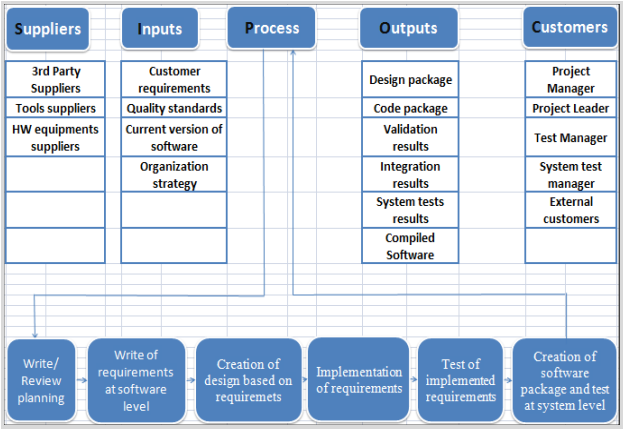

Figure 4. SIPOC Diagram on software development

These inputs are processed to generate vents, the process being described to create an overview and can be used to improve product development using quality tools; for example, can be eliminated non-value steps, can be determined the points where you add actions for improving the process, can parallelize certain actions. The last step is to identify customers for the product / process and performance requirements from them.

3.5 Analyze phase

During this phase the requirements from customer are translated with the help of QFD instrument as a team work. House of quality was created with Qualica tool, quality characteristics were arranged

by importance and are detailed in figure 5 and 6; the house was split in two figures for a better visibility.

CTQs by Priority 1 / Needs by Priority	3 Development of software package in an incremental way, in small packages, weekly or bi-weekly.	4 Using of continous integration technique.	1 Using a knowledge management system.	2 Automatization of integration and validation tests.	Number of significant relationships	Importance to Customer	Relative Importance
1 Shorter time of development/testing.		○	△	⊙	4	5,0	9,4%
4 Reduction of errors identified by customer.	⊙	○	⊙		5	5,0	9,4%
8 Maintain quality.	⊙	○	△	△	2	5,0	9,4%
9 Software developement process must be compatible with qulity standards.	○	○			2	5,0	9,4%
3 Small releases should be possible to be done in 1 week.		○		⊙	3	4,0	7,5%
5 Reduction of errors detected after release.	⊙	○	⊙		5	4,0	7,5%
11 Possible to integrate new functions in late stages of development.	○	○		○	3	4,0	7,5%
13 Errors must not be repeated.			⊙	△	2	4,0	7,5%
2 Resources must be used in a efficient manner.	○			△	4	3,0	5,7%
6 Detection of errors before release.	⊙	○		○	5	3,0	5,7%
10 More efficient testing.	○	⊙	△	⊙	4	3,0	5,7%
12 Prevention in case of leaving / missing team members.			⊙	△	3	3,0	5,7%
14 Well defined tests and easy to execute.		⊙	△	○	3	3,0	5,7%
7 Detection of errors before testing phases.	⊙	○			4	2,0	3,8%
Number of significant relationships	9	11	4	6			
Importance	20,6%	15,7%	15,2%	14,6%			

Figure 5. House of quality part 1

Identified quality characteristics for process design with significant importance are: development of software packages done in small incremental steps/packages, using of continuous integration technique, using of a knowledge management system and automation of integration and requirements validation tests. The priority is calculated based on the correlation with the customer's requirements and the value of importance for customer.

CTQs by Priority 1 / Needs by Priority	6 Creation of documents that guide a tester throught testing phases / tools and also contains lessons learned.	3 Software testing during development.	2 Parallelization of certain phases from software product development.	8 Review of work after each phase with relevant persons.	5 Specialization of team members on components must be various.	Number of significant relationships	Importance to Customer	Relative Importance
1 Shorter time of development/testing.	○		⊙			4	5,0	9,4%
4 Reduction of errors identified by customer.	△	⊙		○		5	5,0	9,4%
8 Maintain quality.	△	△		△	△	2	5,0	9,4%
9 Software developement process must be compatible with qulity standards.				△		2	5,0	9,4%
3 Small releases should be possible to be done in 1 week.	△		○			3	4,0	7,5%
5 Reduction of errors detected after release.	△	○		○		5	4,0	7,5%
11 Possible to integrate new functions in late stages of development.			△			3	4,0	7,5%
13 Errors must not be repeated.	○					2	4,0	7,5%
2 Resources must be used in a efficient manner.		○	○		○	4	3,0	5,7%
6 Detection of errors before release.	△	○		○		5	3,0	5,7%
10 More efficient testing.	○	△				4	3,0	5,7%
12 Prevention in case of leaving / missing team members.	○				⊙	3	3,0	5,7%
14 Well defined tests and easy to execute.	⊙					3	3,0	5,7%
7 Detection of errors before testing phases.		○		⊙		4	2,0	3,8%
Number of significant relationships	5	5	3	4	2			
Importance	8,8%	8,5%	6,7%	6,1%	3,9%			

Figure 6. House of quality part 2

During the design phase solutions for the most important quality characteristics are proposed and a detailed design is created for every change.

3.6 Design Phase

Change one. Reduction of time spent on test phases through automation and lessons learned.

Current status is the following: execution of integration tests takes 6 days effort and validation tests takes 9 days effort. Improved planned status will decrease effort to 4 days for integration tests and 6 days for validation.

In order to reduce test time steps of DMADV methodology were applied again.

- In the definition phase were established objectives, the main objective being to reduce testing time by 30%. The secondary objectives are to create a clear picture of the effort for each test and people who has experience with them.
- In the measurement phase, duration was stored during the execution of each test and statistics were made to observe the number of automated tests compared to the manuals.
- The analysis phase includes examining each separately and manually test and estimating the effort for its automation. Depending on the manual execution time, the effort estimation of automation and the release number, of which we estimate using the tests that estimate deciding whether the test will be automated or not. Decision formula is the following: Decision = AutomationEffort – PackageNumber * EffortExecutie, if decision <= 0 then decision is positive, otherwise test will be untouched.
- During the design phase the automated tests are written and it is defined the process of tests execution.
- Last step is the verification, during which was measured the execution time of tests. As result it was observed that effort for integration tests was reduced with 2.5 days and for validation tests with 3 days, so it was a gain of about 37% time.

Note: Execution Effort = Execution time * Assigned persons

Change two. Division of release package into weekly packages

To reduce the number of errors found after release and the number of errors found by the customer the software development process will be re-designed. The risks must be minimized, so development will follow same phases, following the V-Model, it will just be divided into several packages that will have as client the testing team. Of course this approach can be used only if the tests are largely automated and testing period is relatively short, therefore not in the first releases of the project and is advised to use it in carried over projects.

The objectives of the approach are the followings:

- ➔ The detection of errors / not accomplished requirements is done before final release package.
- ➔ Ability to solve errors / not accomplished requirements in the final package.
- ➔ Planning activities in an efficient way, without free time lost.
- ➔ Reducing development time by parallelizing certain action.

The changes being made in small incremental steps, if problems are discovered, the time of analysis is much shorter and to cause of the issue is easier to identify. The approach can be understood in Figure 7.

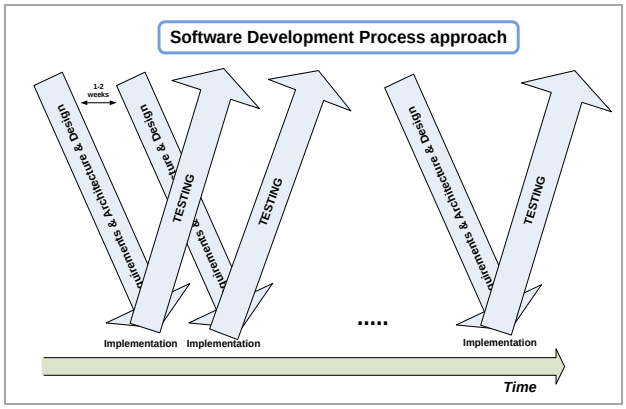

Figure 7. Overview of development process approach

A detailed image of the process is presented in figure 8, containing the base phases of development and underlining the phases that can be done in parallel during software development.

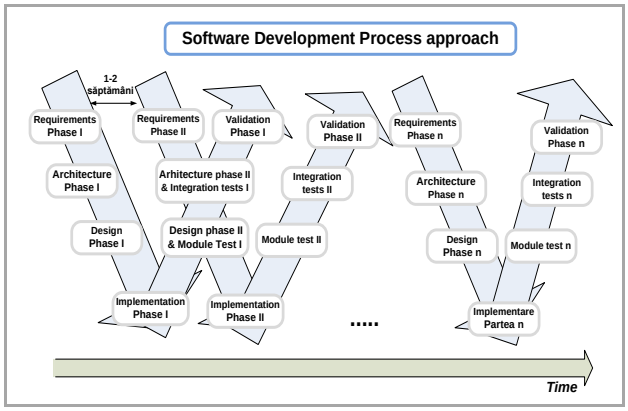

Figure 8. Development process approach, detailed image

Change three. Integration of knowledge management into used development model, namely V-Model.

Management of knowledge is essential in complex projects from automotive industry. For introducing a knowledge management into project without a lot of effort spent on teaching and learning for all team members, was chosen to structure the knowledge upon already known steps, namely: concept phase, requirements, architecture, design, development, software integration, requirement validation, system test and maintenance. Knowledge gained during interaction with customer has a special place into the model; it should be highlighted thought the other knowledge articles. The approach for integration of knowledge management into project management is described in figure 9.

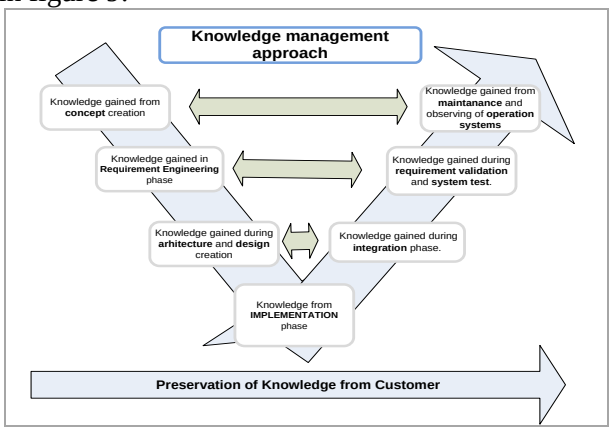

Figure 9. Knowledge Management in V Model
(Nicolaescu S., Kifor C., 2014)

The objectives of the approach are the followings:

- ➔ Integration of knowledge management into project management phases.
- ➔ Structure the knowledge in a well known shape by all members of the team.
- ➔ Adding the possibility to filter the knowledge after work phases of the team members.
- ➔ Management of knowledge inside project team, providing a knowledge transfer with minimum effort.
- ➔ Capitalization of knowledge gained thought interaction with customer.

Change four. Use of continuous integration technique

Continuous integration in software development projects can be described by integrating the work done by each team member, daily work can be integrated or a customized approach can be use. A set of automated tests is defined and executed at each integration, offering an advantage because in case of a failed test, the error is identified in advance and the problem can be sought only in the changes in the current integrated package. The technique is used in Agile methodology but can be effectively adapted and integrated into V-Model.

Change five. Adding in test strategy a day dedicated to testing

This strategy involves the planning of a day when the entire team, including development engineers, test engineers and project managers verify basic functionality integrated into current

software package. Testing can be done individually or in a team, black box / white box and it is important that each team member can review the final product that has contributed. In this way one can see mistakes that occurred in the early stages of the development process (due to mistakes or misinterpretations), such as writing requirements, architecture design, design or can occur ideas for improvement, because each team member understands the functionalities different, through the experience of testing.

3.7 Verification phase

We used 2 projects, 3 software packages for verification and stabilization of implemented improvements in the project.Measurements were made taking into account three important elements in the development of software products: time, errors /unfulfilled requirements and quality. The results are the following:

- ➔ Time of test activities was reduced by approximately 35%.
- ➔ The number of not respected requirements at testing identified before release, were increased by an average of 30% (the average was made between the obtained results from the software packages that have been checked)
- ➔ Identification of the not respected requirements during testing was done in a timely manner, during the first sub-packages before freezing the software package; therefore requirements and functionalities have been implemented correctly in the final package. Verification will be continued further on other 5 software packages for more reliable data.
- ➔ The number of unfulfilled requirements identified by the customer was zero. Checking will be continued on other 5 software packages for more reliable data.
- ➔ The process was followed and no quality steps were skipped during development of software packages. The continuous integration, it even offers improvement into quality and evaluation of quality.

4 CONCLUDING REMARKS

Using Design for Six Sigma methodology is the best way to ensure that all requirements and customer needs are integrated into the process / product that is developed.Aim of the study was to prove that Design for Six Sigma DMADV methodology is suitable for improving the projects and processes used in software development from automotive industry R&D centers. As a result, DMADV methodology, together with used instruments showed an increase in the quality of process / product. The strategy of placing the customer first, exploiting and management of knowledge from project and product quality in Design for Six Sigma methodology has proved effective.

5 REFERENCES

► Balaji, S., Murugaiyan, S. (2012), Waterfall Vs V-Model Vs Agile: *A Comparative Study on SDLC*, International Journal of Information Technology and Business Management 29th June 2012. Vol.2 No.1

► Basem, H., Shaout, A., *Software Design For Six Sigma, A Roadmap for Excellence*, John Wiley & Sons Inc.

► Cholewa, M. (2011), *Product Lifecycle Management*, Printpap Lodz

► Cudney, E. , Furterer, S., (2012*), Design for Six Sigma in Product and Service Development, Applications and case studies*, Taylor & Francis Group

►Kifor, C., Oprean, C. (2006), *Ingineria calităţii: metoda şase sigma*, Editura Universităţii Lucian Blaga din Sibiu

► Nicolaescu, S., Kifor, C. (2014), *Knowledge Management in Automotive Industry - Phases and customer focus,* Quality – Access to success, Vol. 15, No. 143, December

► Maass, E., McNair, P. (2010) *Applying Design for Six Sigma to Software and Hardware Systems*, Pearson Education,Inc.

► Pande, P., Neuman, R., Cavanagh, R., (2000), *The Six Sigma Way*, The McGraw-Hill Companies

► Pyzdek, T., Keller, P. (2010) *The Six Sigma Handbook*, 3rd Edition, The McGraw-Hill Companies, Inc

► Saini, D., Hadiman, L., Vaidya, P., Maskari, S., *Software Quality Model Six Sigma Initiatives*, Proceedings of the World Congress on Engineering 2011 Vol II WCE 2011, July 6 - 8, 2011, London, U.K.

► Shenvi, A. (2010), *Design for Six Sigma* (DfSS) in Software, Intech

► Qianmei, F., Kailash, K., *Integrated Design and Optimization Models for The Six Sigma Process*, Second World Conference On POM and 15th Annual POM conference, Cancun, Mexico, Aprilie 30 May, 2014

► Tancoa M., Mateoc R., Santosa J., Jacaa C (2012) *On the relationship between continuous improvement programmes and their effect on quality defects: An automotive case study*, Total Quality Management Vol. 23, No. 3, March, 277–290