AN ASSESSMENT MODEL TO FOSTER THE ADOPTION OF AGILE SOFTWARE PRODUCT LINE IN THE AUTOMOTIVE DOMAIN

Surekha.V[1], Kavitha.S[2]
Research Scholar M.Phil. (CS)., Assistant Professor,(CS)
Department of Computer Science
Auxilium College, Vellore, India.

Abstract : A Software Product Line is commonly used for the Software Development in Large Automotive Organizations. Software development organizations frequently face changes that require them to be flexible. The principles and practices of Agile software are often associated with improving software organizations' flexibility. However, Introducing agile practices have its benefits and limitations. To amplify benefits and all evirate challenges, Agile adoption guidelines are being proposed to provide strategies for introducing Agile practices. One instance of such guidelines is known as Agile Maturity Models (AMMs). A strategic reuse of software is needed to handle the increasing complexity of the development and to maintain the quality of numerous software variants. However, the development process needs to be continuously adapted at a fast pace to satisfy the changing market demands.

Introducing agile software development methods promise the flexibility to react on customers' change requests and market demands to deliver high quality software. Despite this need, it is still challenging to combine agile software development and product lines. The maturity of an agile adoption is often hard to determine. Assessing the current situation regarding the combination is a first step towards a successful inclusion of agile methods into automotive software product lines. Based on an interview study with 16 participants and a literature review, build the so-called ASPLA Model allowing self-assessments within the team to determine the current state of agile software development in combination with software product lines. The model comprises seven areas of improvements and recommends a possibility to improve the current status.

The combination of agile software development and software product lines in the automotive domain is seen as a promising approach. With this approach, a shorter time to market and a faster learning loop about the maturity of the software could be achieved. The current status of the agile adoption within software product line is hard to define. In this paper, to examine the aspects that needs to be considered for an adjusted assessment model that assess an organization's current situation regarding agile software development and software product lines. Several assessment models for CMMI and ASPICE, XP, ISO 26550 models are used.

IndexTerms - Agile software development, software product lines,process maturity framework, software process improvement,automotive domain,embedded software development, ASPLA Model.

I INTRODUCTION

Flexibility is important for any organization, including software organizations to keep up with changes in the business environment and maintain a competitive advantage. In software engineering, flexibility is often associated with the principles and practices of agile software development. Agile software development is a group of software development methodologies, e.g., Extreme Programming (XP), Scrum, and Crystal that focus on delivering working software products in small iterations, being adaptive towards requirement changes, and collaborating closely with customers.

The increasing complexity of software can be addressed by a strategic reuse, to manage the development and to maintain the quality of numerous customized software variants. Software product lines are a software paradigm for systematic software reuse and commonly used in the automotive software development. In the automotive embedded development it is necessary to manage the high number of different software variants that meet different requirements across multiple markets, while simultaneously maintaining the quality of the software.

Current software development in the automotive domain is heavily structured by standardized processes. Process assessments are used to evaluate the processes of the organizational unit against a predefined process assessment model. The most popular standards in the automotive domain are CMMI and Automotive SPICE (ASPICE). Assessing the current status of the development is a prerequisite for a successful combination of agile methods and software product lines in the automotive domain. The identified the need for an adjusted assessment model, addressing Agile Software Product Lines in the Automotive Domain (ASPLA Model).

Maturity Model Integration (CMMI) or Software Process Improvement and Capability Determination (SPICE) is not suitable for Agile processes. AMMs claim to provide Agile-specific guidelines for practitioners to engage in Agile transformation while managing its risks and challenges. AMMs usually follow an evolutionary progression with levels similar to CMMI or SPICE. Typically AMMs map Agile practices and maturity levels, indicating some practices are to be introduced before the other. Three examples of typical AMMs. As we can see from, AMMs suggest that Agile practices should be gradually and continuously added. However, AMMs are not in agreements which Agile practices should be introduced at which maturity level. A similar observation is also reported by Leppanen. With contradictory suggestions among the AMMs, practitioners do not yet have the means to determine which AMM or which order of practice introduction would suit them best. Most importantly, currently there is no way for practitioners to tell if one strategy suggested by one AMM would lead to a more successful Agile practice implementation.

II BACKGROUND

1.Software Productivity

Principally, software productivity depends on overall software process, tools/technologies,

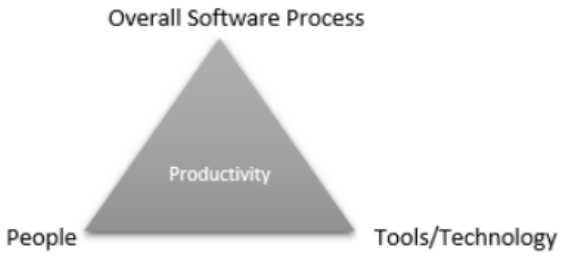


Fig. 1: Software Productivity

Processes for executing software projects have been studied actively for over three decades. Tools and technology to improve productivity has also been an active area of development and improvement and continue to evolve. However, there is insufficient understanding of programmer productivity, particularly at a task level. This thesis focuses on this aspect.

2.Measuring Software Size

The software size measures commonly used in the calculation of software productivity include lines of code (LOC), function points (FP), testable requirements, and use case points.

3.Software Process Improvements for Improving Productivity

During the last three decades, focus was on the importance of the overall process of software development on software productivity. This naturally led to an increase in emphasis on improving software process for improving software productivity. Improvements in the overall software process increase software productivity by identifying and eliminating waste during the software development and optimizing existing methods to reduce the software development effort. Recognizing the challenges in software process improvements, some frameworks have emerged to help organizations improve their process. Some of them are briefly described here.

- **CMMI:** Capability Maturity Model Integration (CMMI) is a framework introduced by SEI, CMU to assess and improve overall software process for better productivity. Processes of projects/organizations are evaluated to know the maturity level of an overall software process. Maturity levels, under CMMI framework, are classified into the following five levels: Initial, Repeatable, Defined, Quantitatively Managed, and Optimized. Organizations improving its CMMI maturity level by one have reduced their development effort resulting in productivity improvement. Projects/Organizations certified with level 5 (optimized) rating are considered to have mature overall software processes and strive for continuous quality/productivity improvement.
- **ISO:** International Organization for Standardization (ISO) determines the process and product capabilities and improvements. Unlike CMMI that concentrates only on software processes, ISO is a generic platform used across many industries for evaluation of both processes and products. Adopting the guidelines and standards of ISO in software projects had improved quality and productivity
- **DOI78:** This framework is mainly used to assess avionics-related software. This framework first classifies the avionics software into one of the five levels (based on the existence of bugs in the software): A, very critical and causes damage to life; B, critical when immediate action is not taken; C creates panic when action is not taken; D to E, non-critical. Further, according to the level of the software, this framework imposes various engineering and project management practices in software development.

D) Backdrop of Agile Manufacturing

The business dynamics in the manufacturing environment has changed drastically over the last two decades due to rapid changes in manufacturing and information technology, changes in market conditions, increased customer requirements (i.e. quick response, lower costs, greater customization etc.), product proliferation with shorter and uncertain life cycles, intensified off-shoring and outsourcing strategies, and increased competition from local to global arena. Therefore, the survival and success of a manufacturing organization has become even more difficult. It is crucial for any manufacturing organization to deal with the changes much quickly; otherwise there is a threat to becoming extinct. Manufacturing organizations that refused to heed to the changes have shut shop. The refusal to heed to the changing scenario usually stems from the fact that the organizations presume what is their core competency will tide them over during the turbulent times. Change in technology, materials and processes sometimes render these rigid decisions as failures. Many Iron and Steel industries that did not update their technologies/ processes, had to close down as high operation costs ASPLA them commercially unavailable.

Manufacturing organizations need to incorporate processes to deal with changes. There have been major shifts in the core business principles it was a “manufacturer centric” in nature, where the business model was simple with least number of variables and a lot of confidence about what the customer really wanted. The premise on which the business was conducted has become obsolete. The socio-economic environment in which the manufacturing companies are expected to operate now, have become unstable owing to multitude of factors viz. non-uniform local legislations, risk due to financial upheavals, paucity of resources, vacillating loyalty of customers and suppliers, and a strong emphasis on “customer desires and satisfaction”. This has led to a situation where the sustainability of a manufacturing company is directly related to its ability - to face the growing.

1.Agile Manufacturing Enablers

Agile Manufacturing Enabler (AME) is the factor that has the capability to provide or enable or enhance the level of agility in the agile manufacturing system. Many researchers have carried out research on AMEs and identified AMEs which may be specific or generic in nature. The manufacturing organization focusing on AM should identify the AMEs and then define the

domain of each enabler so that right AMEs can be selected in a specific manufacturing environment. Although number of enablers have impact directly or indirectly or both ways on the agility performance of a manufacturing system, it is not possible for an organization to focus on all the enablers at a time in order to enhance agility performance level.

2. Agile Manufacturing Impediments

The AM implementation process would most likely get delayed if root causes of Agile Manufacturing Impediments (AMIs) are not identified and effectively addressed. These AMIs have deep roots along various tangible and intangible issues of the organizations. Therefore, an organization needs to target the appropriate AMIs to enhance the agile development as putting efforts on all AMIs is not feasible. But many a times organizations fail to identify the appropriate AMIs due to improper analysis. Thus, considering all the aforementioned issues, this study proposed an approach to identify the appropriate impediments for monitoring the smooth implementation of AM in specific environment.

3.Data Analysis Techniques

The data that are collected from the Models are statistically analyzed to validate the hypotheses formulated to investigate the research questions under consideration. The summarize below some of the statistical analysis methodologies the have used. Descriptive statistical methods will be used for data exploration to gain an overall understanding of the nature of the data collected. The inter-question reliability will be tested using Cronbach's alpha test. This will help us to understand whether the responses to the different questionnaire items show high inter-question correlation. Correlation and regression analyses will be performed to understand the relationships between the dependent and independent variables.

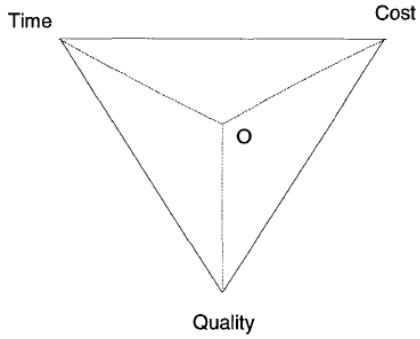


Fig. 2: Quality/Time/Cost Tradeoff Triangle in Software Development

Jim High smith, in his popular book, "Agile Project Management" lists five business objectives of exploration processes, i.e., processes that are capable of operating in uncertain environments.

1.Continuous Innovation: This requires delivering products and services according to current customer needs and requirements. These needs are requirements that do not stem out of structured, plan driven environments. Continuous innovation is supported by adaptive organizational culture involving self-organization and self-discipline. It is measured by how a project an deliver customer value today.

2.Product Adaptability. This requires delivering products according to changing customer requirements. These changes may happen within the duration of few weeks to few years. The product should be adaptable to the changing requirements. Ideally, such changes should be efficient and cost-effective.

3.Reduced Delivery Schedules: This requires reducing delivery schedules to meet the market requirements. Reducing delivery schedule should be accompanied by increasing return on investment (ROI). Careful attention is given to delivering primarily those features that are important to the customer. Marginally beneficial features are considered secondary.

4.People and Process Adaptability: This requires being dynamic with the changes in the business and the product. Similar to the adaptability of product, the people and the process need to adapt to the changes in the time-varying nature of the market. Instead of viewing changes with a resistive attitude, they should be ASPLA part and parcel of the businesses.

5.Reliable Results: Traditionally, good traditional/plan-driven (predictable) processes required delivering products using processes having repeatable outcome, i.e., processes that would deliver the same even when undertaken after a lapse of time. These processes resulted in predictable outcome within the specified time and budget. However, for exploration processes, what is important is whether the project was able to deliver a valuable product to the customer within the specified time and budget. In exploration processes, although the outcome is not repeatable, innovative results are delivered to the customers in line with their vision.

E) Agile Software Product Lines in Automotive

It analyzes the combination of agile and plan-driven processes. This combination could be seen as a typical characteristic of current automotive software development. They emphasize that a combination is beneficial under certain conditions, such as rigid quality and safety requirements.

Adopting agile software development in the automotive typically concentrates only on selected agile practices such as Continuous Integration or Pair Programming. The published literature does not show any recommendations to use a comprehensive set of agile elements and practices together in the automotive domain.

The majority of the suggests that agile models and processes should get customized to the specifics of the automotive domain before they are implemented in practice. – Agile models and processes that are already customized to the specifics of the automotive domain are proposed in the published literature. An example is the Feedback Loop Model that especially considers the collaboration between different organizations (such as OEMs and suppliers). – Combination approaches include interesting new concepts such as virtual integration on the system level.

1.Agile Principles

The Agile Alliance also documented the principles that underlie the manifesto. Agile methods are principle-based, rather than rule-based and have predefined rules regarding the roles, relationships, and activities. The principles that guide the software developers comprising the team and project manager include:

1) Customer satisfaction through early and continuous delivery of valuable software is the highest priority.
2) Agile processes harness change for the customer's competitive advantage and hence are open to changing requirements, even late in the development process.
3) Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4) Necessarily involve business people and developers so as to work together on a daily basis all through the project.
5) Build projects with motivated individuals. Provide them the necessary setting and support they need, and have confidence in them to get the task done.
6) Convey information in the most proficient and effective method to and within a development team preferably through face-to-face conversation.
7) Working software is the primary measure of progress.
8) Agile processes encourage sustainable development. The sponsors, developers, and users should be able to keep up an unvarying tempo forever.
9) Continuous attention to technical quality and superior design enhances agility.
10) Simplicity, the art of maximizing the amount of work not done is essential.
11) To achieve best architectures, requirements, and designs from self-organizing teams.
12) On a regular basis, preferably at fixed intervals, the team reflects on how to develop into a more effective team then regulates and adjusts its activities consequently.

2.Agile Methodologies

Several Agile techniques have been proposed and used by researchers in difference domains. These Agile methodologies share common principles among themselves but differ in practices. This section identifies some of the well-known existing Agile software development methods and their objectives.
These are described in detail below:

1) Extreme Programming (XP)
2) Scrum
3) Lean Software Development (LSD)
4) Kanban
5) Adaptive Software Development (ASD)
6) Feature Driven Development (FDD)
7) Dynamic System Development Method
8) (DSDM)
9) Agile Modeling (AM)
10) Crystal
11) Agile Unified Process (AUP)

F) Extreme Programming (XP)

Extreme Programming (XP) is a well-known and a light weight discipline of software development that focuses on engineering practices. XP seeks to enable successful software development regardless of ambiguous or continuously changing software requirements. It is a system of practices which is intended to improve software quality and quickly addresses the changing customer requirements to meet business needs. It comprises collection of informal requirements from on-site clients, arranges teams of pair programmers, developing simple designs, continuous refactoring, and continuous integration and testing; and advocates frequent releases in short development cycles that improves productivity as well as introduces checkpoints where new customer requirements can be embraced.

Scope

XP is best suited for projects that require collocated teams of small to medium size team. On the project side XP is meant for assignments where the requirements are unstable and unpredictable.

Advantages

- Communication: It is definitely a key factor to the success of any project as most projects fail because of poor communication. It is achieved by combined and co-located workspaces and development and business spaces, paired development, recurrently changing pair partners, often changing assignments, public status displays, short stand-up meetings and unit tests, demos and oral communication, not documentation.

- Simplicity: This refers to developing the simplest product that meets the customer's needs. It supports delivering the simplest functionality that meets business requirements, designing the simplest software that supports the needed functionality, building for today and not for tomorrow and writing code that is simple to read, comprehend, maintain and amend.
- Feedback: It means that developers must obtain and value feedback from the customer, from the system, and from each other. It is provided by aggressive iterative and incremental releases, frequent releases to end users, co-location with end users, automated unit tests, automated functional tests.

XP Activities : Coding, Testing, Listening, Designing
XP Practices : It is based on following 12 practices

- Planning Game: It is collaboration between a customer and the developers where iteration planning for next release is performed, customers provide user stories followed by determining budget and schedule estimates.
- Small Releases: It supports the planning game. Working software is delivered in small and frequent releases and is determined in terms of functionality.
- System Metaphor: XP teams develop a common vision of how the program works which is called metaphor. It is the oral architecture of the system which describes how the program works.
- Simple Design: Do as little as needed and provide simplest possible design to get job done. The requirements will change tomorrow, so only do what's needed to meet today's requirements.
- Test Driven Development: Extreme programming supports verifying and validating the software throughout the entire project development lifecycle. Developers start by writing test cases first and then write codes as reflected in test requirements followed by user acceptance test and customer approval.
- Refactoring: XP development teams enhance the design of the software all through the whole development lifecycle which is done by refactoring out any duplicate code produced in a coding session. Refactoring is simplified by using automated test cases comprehensively.
- Continuous Integration: New features and changes are incorporated into the system instantly. The development team focuses on continuous integration of the software by verification and validation of the software throughout the product development lifecycle.
- Collective Code Ownership: This suggests that developers own their code and facilitates refactoring.
- Pair Programming: XP Programmers write all production in pairs, two programmers working together at one machine.

XP Roles and Responsibilities

- XP Coach: Guides team to follow XP process
- XP Customer: Writes stories, functional tests and sets implementation priority
- XP Administrator: Setup programmer environment and acts as local administrator
- XP Programmer: Writes tests cases and code
- XP Tracker: Tracks iterations and provides feedback on accuracy of estimates
- XP Tester: Helps customer write, run functional tests and maintains testing tools
- XP Consultant: An external member who guides the team to solve problems

Limitations

- XP is not suitable for large, difficult or complex projects.
- It requires great amount of coordination between the programmers while doing pair programming and any small conflict may damage the objective of collective score ownership and hence impact the iterations.
- Development of 'metaphor' is required to be shared within team carefully to ensure the common understanding of the terminology.
- Pair programming is a noteworthy practice in XP; in which two developers work on the same machine at the same time and hence it cannot be applied projects with only one developer. Since the testing and coding is done by the same developer, all the probable problems may not be identified as developers test from the same insight the software is created.

Objectives

The research gaps have been identified based on the literature reviews of the individual studies. Following research objective(s) were outlined that commences with understanding of honeycomb application process in detail and also includes various honeycomb application characteristics, issues and challenges, and best practices adopted by honeycomb development community for a successful honeycomb application development process.

a. To study the honeycomb application development process using Agile software development methodologies, such as XP, ASPLA and Automotive assessment models. Each of these Agile approaches is focused on different aspect and comparing these methodologies is imperative.
b. To conduct a technical Model for gaining a better understanding of prevalent development practices for honeycomb applications thereby identifying the problems and challenges faced by the honeycomb professionals related to application development.

c. To investigate and implement a robust approach for each phase of honeycomb software engineering process using various Agile methodologies; identifying various challenges faced by honeycomb developers and best practices followed to build and deliver a successful honeycomb applications.

d. To identify the best fitting Agile approach and integrating specific Agile practices, to meet the needs of volatile honeycomb projects and to assist honeycomb developers and managers during the honeycomb application development process.

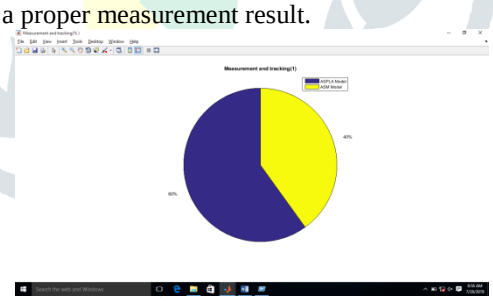
III LITERATURE SURVEY

The current situation and future scenarios of the automotive domain require a new strategy to develop high quality software in a fast space. In the automotive domain, it is assumed that a combination of agile development practices and software product lines is beneficial, in order to be capable to handle high frequency of improvements. This assumption is based on the understanding that agile methods introduce more flexibility in short development intervals. Software product lines help to manage the high amount of variants and to improve quality by reuse of software for long term development.[12] [24]Software reuse enables developers to leverage past accomplishments and facilitates significant improvements in software productivity and quality. The contribution of this paper is a recommended process model for the implementation of software reuse effectively. A critical problem in today's practice of software reuse is the lack of a standard process model which describes the necessary details to support reuse based software development and evolution. Our research thesis is that software development based upon a reuse-based process model improves quality of products and productivity of processes. A quantitative survey of 100 software organizations is used to test the new process model and the hypothesis of the study. The process model presented in this paper identifies process level, organizational and technical aspects which have to be improved to achieve success in the reuse world. [9] In modern cars, most of the functionalities are controlled by software. The increased significance of software-based functionality has resulted in various challenges for automotive industry, which is slowly transitioning to-wards being a software centric industry. Challenges include the definition of key competencies, processes, methods, tools, and organization settings to accommodate combined development of software and hardware. Based on qualitative research, this paper aims at understanding the applicability of agile methods to automotive software development. The explorative case study with one of the development sections at Volvo Car Cooperation identified challenges in their software development process related to process perception and reactive mode, multi-tasking and frequent task switching, individualism and lack of complete knowledge, as well as long communication chains and low cross-function mind set. Moreover it prepares a transition of software development at this multinational automotive company towards agile by relating agile principles and practices to automotive software process challenges.

IV RESULT AND ANALYSIS

The Elements in the ASPLA model are mapped to parts of relevant and current automotive domain standards. A complete list of all issues can be found in. The focus on the specific issues of (1) Measurement and tracking, (2) Traceability, and (3) Verification and validation.

(1)Measurement and Tracking: Measurement tasks in product line engineering and the management of the product line itself, are complex. Due to the separation of domain engineering and application engineering life cycles, the data collection, the measurement and tracking needs to be synchronized. Furthermore, the organizational and technical management of the product line needs to be considered to receive a proper measurement result.



a.

Fig. 3:Measurement and Tracking

(2) Traceability: The development in a software product line is typically knowledge-intensive and a lot of collaboration and coordination. It is important to trace and manage the knowledge to control the complexity of the overall soft- ware product line and the development of single variants. Traceability helps to keep track of decisions regarding the development.

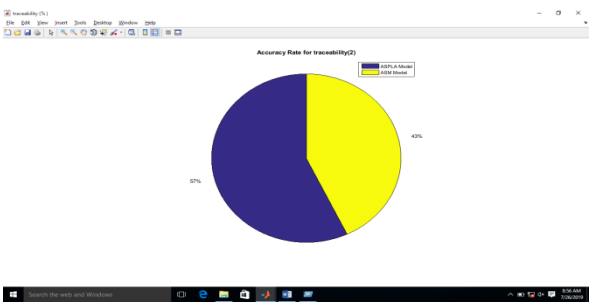


Fig. 4: Traceability

(3) Verification and Validation: Verification and validation confirms that the requirements for all domain assets and member products are fulfilled. Verification and validation in product line context must consider all software variants and are therefore fundamentally different from the single-system engineering context. summarizes the coverage of the ISO 26550 process specific issues and presents a mapping to the presented outcomes.

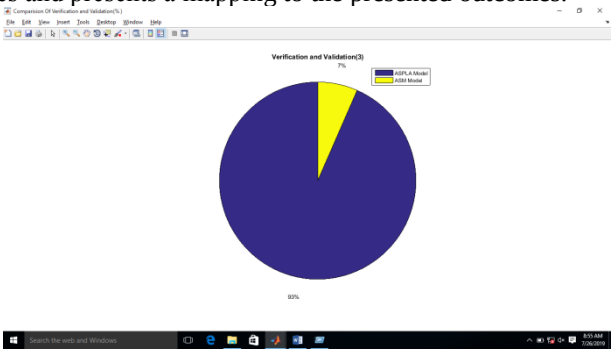


Fig. 5: Verification and Validation

A)Comparison between the performance Changes on Testing

The Elements in the ASPLA model are mapped to parts of relevant and current automotive domain standards. A complete list of all issues can be found in. The focus on the specific issues are comparison between the performance Changes on testing to be proved.

Unit Test

Unit Testing is a level of individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

System Test

System Testing is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements. That tasks in product line engineering and the management of the product line itself, are complex. Due to the separation of domain engineering and application engineering life cycles, the data collection, the measurement and tracking needs to be synchronized. Furthermore, the organizational and technical management of the product line needs to be considered to receive a proper measurement result. The development in a software product line is typically knowledge-intensive and a lot of collaboration and coordination. It is important to trace and manage the knowledge to control the complexity of the overall software product line and the development of single variants. Traceability helps to keep track of decisions regarding the development.

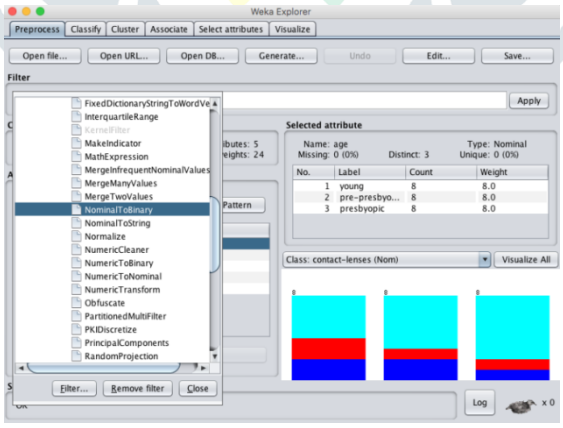


Fig. 6:The Performance Improvement on Before Validation

Verification and validation confirms that the requirements for all domain assets and member products are fulfilled. Verification and validation in product line context must consider all software variants and are therefore fundamentally different from the single-system engineering context. summarizes the coverage of the ISO 26550 process specific issues and presents a mapping to the presented outcomes.

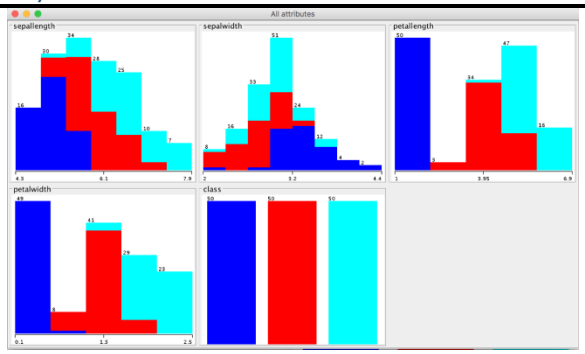


Fig. 7: Testing on Measurement Tracking, Traceability, Verification and Validation

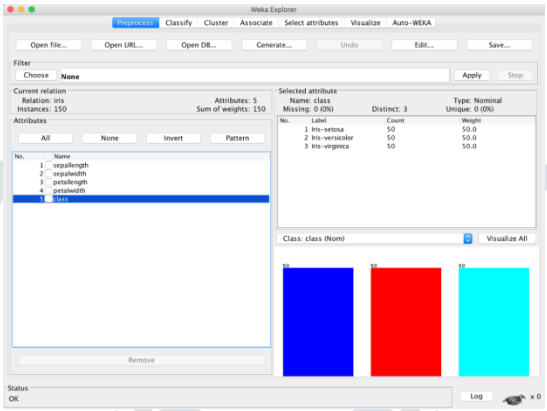


Fig. 8: The Performance Improvements on After Validation

A)Support and Maintenance

Most of the survey participants highly emphasized the necessity to capture the defects based upon user-market feedbacks, through memory optimization, automated crash reports, and change requests from users. They highly recommended that product maintenance (support) and product improvement (upgrade) should be done in frequent iterative releases with bug fixes via app store or enterprise deployment, This support should be improving the app with platform updates, new features, and functionalities

PHASE	*STAGE*	*DESCRIPTION*
Step 1 Envision	***Analysis***	***Initial requirements envisioning (Identify potential projects, Gather Precise business requirements, Functional and Systems Engineering Specification)***
		Identify a "PROBLEM/PURPOSE" for which App will be developed and Address the exact services the app will offer to the business and decide the features of the app Requirement Analysis and Finalization
		Identify a "PROBLEM/PURPOSE" for which App will be developed and Address the exact services the app will offer to the business and decide the features of the app Requirement Analysis and Finalization
Step 2 Solution	***Design***	***Design Specification (Detailed Module Level Design Specification, Create user interface steps model, Create security model) integrated, Finalize UI design and make wireframes) Create a Test Plan (Write story cards. test plan and test code)***
Step 3 Quality Assurance	***Testing***	***Defining Test Cases (Module Stand Alone Test Spec, Integrated and System Test Specification) Testing (Write Unit Test Code, Automated Testing, Regression Testing, Unit testing, Implementation and user testing***

		and Acceptance Testing ***Testing on Emulator and over wide variety of devices for Usability, Functionality, Compatibility, Performance, Interoperability, Security, Localization, Connectively, Test Documentation)***

Table 1:Maintenance of ASPLA Model

The identify the need for an automotive specific assessment model in previous research the examine the aspects that need to be considered for an adjusted assessment model that assess an organization's current situation regarding agile software development and software product lines several assessment models use. However, the assessment models do not focus on agile practices in detail. The address these insufficient assessment models with the definition of the ASPLA Model. The ASPLA Model comprises the results from an Application. The experimental result show that the software product Measure the tracking, traceability and validation and verification for the ASPLA and ASM model the presented assessment model is the only possible way to introduce an improvement.

V CONCLUSION AND FUTURE WORK

A)CONCLUSION

The combination of agile software development and software product lines in the automotive domain is seen as a promising approach. With this approach, a shorter time to market and a faster learning loop about the maturity of the software could be achieved. However, the literature often recommends to introduce single agile practices into plan-driven processes in one particular context. A holistic approach to combine agile software development and software product lines in the automotive.

The current status on the agile adoption within software product line is hard to define. The identify the need for an automotive specific assessment model in previous research the examine the aspects that need to be considered for testing process. Communication is another important field to consider. The ASPLA Model recommends a close customer contact. Breaking down the "knowledge silos" and establish an open communication is recommended by the ASPLA Model.

The ASPLA Model can be used a guideline in an assessment to identify Honeycomb which need to be considered. Due to the adaption to the context of agile software product lines in automotive, it foster an agile introduction more than other assessment model. Furthermore, it is based on best practices what leads to an acceptance of the model. The model was primarily designed for the automotive domain and may not be generalized for other domains. Furthermore, it cannot be guaranteed that the presented assessment model is the only possible way to introduce an improvement.

B)FUTURE WORK

For Future Work, plan to evaluate the different types of models, further validation, evaluate the different model identifiers and represent the current state of the team under assessment regarding the combination of agile software product line in the automotive domain.

REFERENCES

1. T. J¨ulicher and M. Delisle, "*Step into the circle-a close look at wearables and quantified self*," in Big Data in Context: Legal, Social and Technological Insights, T. Hoeren and B. Kolany-Raiser, Eds. Cham: Springer International Publishing, 2018, pp. 81–91. [Online]. Available: https://doi.org/10.1007/978-3-319-62461-7 10
2. D. Thomas, "*Cellphone addiction and academic stress among university students in thailand*," INTERNATIONAL FORUM JOURNAL, vol. 19, no. 2, pp. 80–96, url = http://ojs.aiias.edu/index.php/ojs/article/view/187, 2017.
3. Leitner, R. Mader, C. Kreiner, C. Steger, and R. Weiß, "*A development methodology for variant-rich automotive software architectures*," e &iElektrotechnik und Informationstechnik, vol. 128, no. 6, pp. 222–227, 2011.
4. S. Thiel, M. A. Babar, G. Botterweck, and L. O'Brien, "*Software product lines in automotive systems engineering*," SAE International Journal of Passenger Cars - Electronic and Electrical Systems, vol. 1, no. 1, pp. 531–543, 2009.
5. L. Wozniak and P. Clements, "*How automotive engineering is taking product line engineering to the extreme*," in the 19th International Conference, pp. 327–336.
6. M. Galster and P. Avgeriou, "*Supporting variability through agility to achieve adaptable architectures*," in Agile Software Architecture, 2014, pp. 139–159.
7. J. Bosch and P. M. Bosch-Sijtsema, "*Introducing agile customercentered development in a legacy software product line*," Software: Practice and Experience, vol. 41, no. 8, pp. 871–882, 2011.
8. U. Eliasson, R. Heldal, J. Lantz, and C. Berger, "*Agile model-driven engineering in mechatronic systems - an industrial case study*," in Model-Driven Engineering Languages and Systems, 2014, vol. 8767, pp. 433–449.