A Development Framework to Standardize Software Engineering Practices

Jishu Guin, Michele Macrì and Andrus Kuus

Software Engineering Research Group
Proekspert AS
Tallinn, Estonia

***Abstract*—The success of an engineering organization evidently depends on the growth of its engineers and the advancement of engineering. In order to realize these two broad factors, there must be inherent support for them at the organizational level. A well suited organization structure can, by virtue of its design, guide the engineering process to attain continuous growth of engineers and advancement of the field. This work explores the potential of matrix organizational structure combined with elements of Rational Unified Process (RUP) as a candidate to drive the success factors towards a desirable direction. The functional units of the matrix are mapped to the major phases of conventional software development process from requirement engineering to testing. The units or groups operate in compliance with the principles of RUP. The work at its current stage is a proposal of the framework and does not attempt to build a theory that can be verified empirically. However, empirical research methods have been considered as a way forward for future work. This paper, based on a preliminary analysis, attempts to show that the proposed structure not only can provide a platform for sustainable growth of engineers and advancement of engineering by incorporating standard engineering practices and methods in software development but also build a synergy to support more significant and challenging endeavors in future.**

Keywords–Matrix; RUP; Empirical; Software; Engineering; Practice.

I. INTRODUCTION

Professional education of engineers demands the acquisition of specialized knowledge as one of the key domains in addition to problem-solving skills and good judgment for the service of society. The nature of this knowledge has a broad spectrum, from fundamental to contextual [1]. Engineering profession provides a platform to apply and evolve this body of knowledge. In the decades of software engineering, various methods and practices have made their way into academia as part of the software engineering curriculum. These methods have proven their effectiveness and importance in industry. Application of these methods over the years built a synergy with the discipline of software quality leading to the development of various practices and tools to improve software engineering e.g., Rational DOORS. In addition to quality, the application of methods to various particular problems evolves the method to encompass larger and more complex scenarios. The goal of incorporating engineering practices in the development process is different from merely creating better software in terms of quality. Means to introduce standard methods and practices can level up the software quality by enriching the overall engineering discipline in the organization.

The work stems from a vision of improvement in the current organization of the authors. The software development workflow in the company is mainly directed towards product delivery. A typical project uses agile methods to ensure business demands are met. The focus of the development process is more on the delivery of features than the engineering methods. The assessment of quality delivered is usually based on information such as defects reported and customer feedback. The drawback of this approach is the suboptimal usage of the merits of engineering which comprises of its standard methods and metrics. The incorporation of industry-proven practices not only adds value to software development, but also towards the engineering culture and advancement. The primary aim of the work is to design a strategy to incorporate practices and methods as an integral part of the software development process to ameliorate the spheres of *culture*, *quality* and *advancement* as shown in Figure 1 to attain the following improvement.

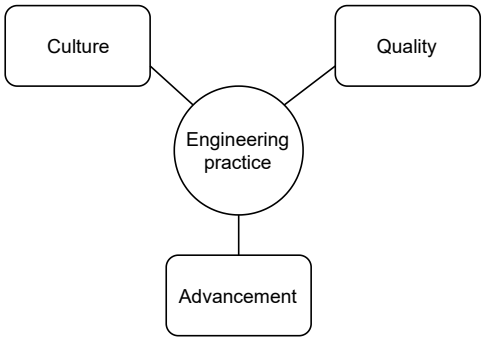

Figure 1. Vision of improvement

Culture - Familiarity of standard engineering concepts among developers and software engineers in order to improve technical communication and cooperation.
Quality - Enhance the quality of development by applying industry-proven methods with measurable outcomes.
Advancement - Assist automation of activities and produce reusable artifacts e.g., binary file, models, source code.

The words *method* and *practice* are both used in the work as means to improve the engineering process. Specifically, *method* refers to the standard technical solution applied in a systematic way to the development process. A solution is standard if it is used in the industry and more likely has a scientific foundation e.g., modeling of requirements. *Practice* refers to certain well recognized activities that can improve software development e.g., use of artifacts as prerequisite for each phase of the process. Further, the words method and

 ISBN: 978-1-61208-776-4

practice will be used interchangeably for brevity on account of their common goal.

The practices that are subject of this work are specifically tied to the software. They cater mainly to the engineering needs of the software and its development. The purpose of the work is not to enforce the use of a set of practices across all projects but to propose a foundational framework that can provide an environment to nurture application of standard practices leading to reuse of practices across projects. The re-usability aspect of the expected result can pave the way for a desired level of standardization.

The initial part of the work draws an overview of software engineering in an organization. This bird's eye view assists to identify certain areas that are associated with software engineering - *Standard Practices* being one of them. The study is significant to understand the co-relation of these areas with the idea of standardizing practices and the way certain factors are impeding the aim of this work. The following part of the work presents a framework to incorporate methods into the software development process. Two solutions are analyzed as part of this work and, based on the results, an attempt is made to reach a candidate solution. The work concludes by providing a guideline for selection of standard practices. The next section discusses the related work in the area of software engineering practices followed by an account of the software engineering domain from the organization's perspective. Section IV derives a strategy to incorporate standard practices and tabulates the guidelines for selection of practices. Finally, Section V presents the conclusion and directions for future work.

II. Related work

The specific nature of the work that embarks on an attempt to create a framework to incorporate engineering practices has limited the number of available related works. The idea of incorporating best engineering practices, however has been proposed by several Software Process Improvement (SPI) models. In Software Process Capability Maturity Model (CMM), the concept of benchmarking is used to accentuate the importance of methods and practices in software process [2]. The SPICE model of SPI distinguishes engineering activities as one of the key process categories [3]. The adoption of these standards remain a challenge for small organizations [4]. Generally, small companies are extremely responsive and flexible, because that is their advertised competitive advantage. Small companies don't have enough staff to develop functional specialties that would enable them to perform complex tasks secondary to their products. The business demand and lack of resource leads to the perception that SPI methods are expensive and time consuming [5]. This gives way to impediments in the optimal usage of standard engineering methods and practices. A number of works discussed in [4][5] propose to improve the software process in small companies. These models of improvement aim to attain certain level of maturity through a process of assessment. Despite their suitability to small companies the methods assume additional tasks that are secondary to the product. These tasks are not specifically focused to incorporate standard practices in the development process. The Technical Debt (TD) literature [6] identifies the lack of best practices as one of the ways to incur debt. Usage of good technical practices is a recommended way to prevent TD [7]. In reuse-oriented software engineering [8][9], the emphasis is on storing reusable knowledge in a repository to improve new developments in future. The model is focused on the storage and accessibility of the knowledge in the form of standard artifacts produced by development activities. Although both, TD and reuse oriented methods, rely on practices, they do not provide sustainable means to incorporate standard methods into development process. Application of standard practices is valuable for attaining aforementioned vision of improvement that this work aims to accomplish. The related works, despite their inclination towards standard practices, fall short in providing a directed strategy to incorporate practices. A targeted strategy to introduce practices is essential to retain the importance of standard methods in the face of business demands. The framework proposed in this work specifically aims to provide an ecosystem to apply and nurture practices. Instead of enforcing a secondary process, the framework makes standard practices an integral part of software development, thus, attaining a balance between business demands and engineering needs.

III. Software engineering domain

The model as shown in Figure 2 identifies *Standard practices* as an area associated with Software Engineering. The work aims to strengthen this area. Introduction of standard practices in the software development process necessarily causes a change in the current engineering domain in the organization. A study of the domain in the company provides insight into this correlation from two perspectives - The factors that impede the use of industry proven methods and the impact of standard practices on the domain. Certain key associations are identified to draw a picture of the engineering domain. The associated areas are elicited by considering the most fundamental connections in a software undertaking based on the conventional knowledge of the field. These associations embody complex relationship with each other. The study doesn't aim to provide a comprehensive exploration of these relations but attempts to draw certain key points that assists in stipulating the guidelines to model a solution. The areas depicted in the model, despite the complex relationship among each other, yields some useful information to model and assess the framework as discussed in the remaining part of the section.

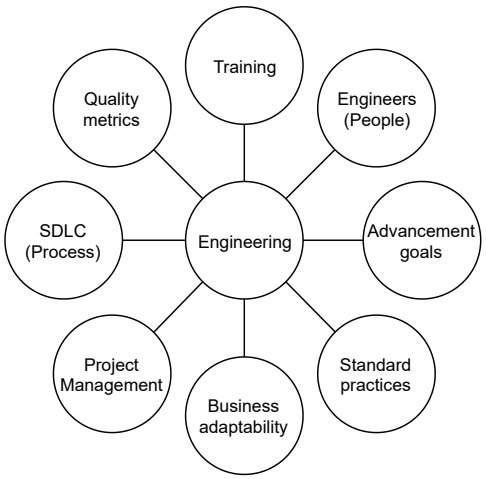

Figure 2. Software engineering domain model

A. Challenges from business

The current state of the development process uses agile methods, e.g., Scrum, that emphasizes on the value it produces for the product and customer. It is imperative to consider this value addition to business for successful operation of this industry. The areas of *Project Management* and *Business adaptability* ensure that the engineering process conforms to those needs. The former enriches business by producing deliverable that generate more value for resources invested and the later ensures that the process adapts to the expected standard of quality and constraints imposed by business. Some industries may not require the high standards of quality as demanded by safety-critical systems. Safety-critical projects bear the cost to ensure the quality demanded by the domain [10]. An implementation of industry-proven practices and methods contribute to the engineering needs of the project and it comes at the cost of time, expertise and budget for tools. The balance between customer and engineering needs of a project decides the balance between business and quality. A shift of focus on the business needs may lead to a counter-intuitive result of degradation in quality. The lack of a mechanism to secure this balance impedes the introduction of standard engineering practices in development.

B. Challenges from engineering

Software development life cycle lies at the core of software engineering process. Software Development Life Cycle (*SDLC*) incorporates main phases of development, from requirement analysis to testing. Standard practices aim to accomplish specific activities of these phases. In order to motivate the use of standard practice, the SDLC must encourage granularity of the activities that standard practices aim to accomplish. The blurred boundary between different engineering tasks compromises the granularity, leading to failure in accommodating standard practices. In contrast, an artifact based development as shown in Figure 3 can lead to a clear prerequisite and output for tasks. Despite the practical challenges of achieving clear boundaries between activities, an effort in that direction can lead to increased usage of standard practices.

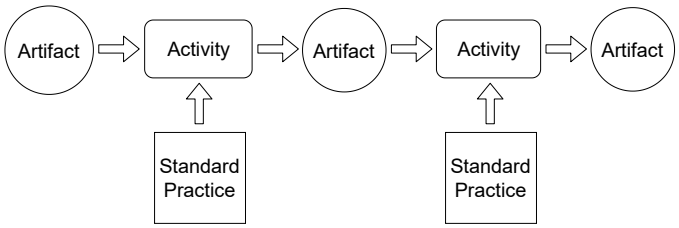

Figure 3. Artifact based workflow

C. Correlation with other areas

The areas of *Quality metrics*, *Training*, *Engineers* and *Advancement* do not directly resist the goal of this work but are influential to *Standard practice*. *Quality metrics* measures the attributes of process and product. The discipline of software quality being a driver for continuous improvement is an area where the organization has major scope for improvement. Standard engineering practices provide measurements that assist in acquiring data to measure quality. As more metrics are instilled in the process, it motivates increased usage of standard practices. The *Training* area in its current state in the company is mostly guided by choices of employees and not according to the demand posed by the development process. A set of standard practices can guide training and thus help to standardize the practices by imparting required knowledge to employees. The area of *Engineers* comprises the people aspect of engineering from hiring specialists to their growth in the company. Criteria for recruiting engineers greatly benefits from a set of practices required for the position. Training of these industry-proven methods expands the skill set of engineers with respect to overall software engineering discipline thus contributing to their growth. *Advancement goals* is a crucial dimension that drives engineering to embark on more significant endeavors e.g., Domain Engineering, Safety-critical systems. A successful realization of this vision necessitates a foundation that comprises standard practices as a key constituent. The next section explores two solutions and attempts to evaluate them based on this canonical domain model.

IV. Incorporation of standard practices

The challenge against standardizing practices is the resistance from aforementioned areas primarily from business and engineering. The aforesaid discussion on the challenges connotes the following two key points that a strategy to incorporate methods must take into account.

Motivation - Structure of the engineering unit must motivate focus on the engineering needs of the project to attain a balance with business aspects.
Sustainability - The process needs to provide a sustainable platform that demands standard industry practices e.g., use of artifacts as integral part of the development process.

The rest of this section describes a primary and alternate solution to the challenge of introducing standard practices followed by an analysis of the benefits and challenges of each solution. Based on the analysis a candidate strategy is drawn that reasonably attempts to address the challenges while retaining the benefits. The section concludes with a set of guidelines to assist the selection of practices.

A. Matrix structure

The balance between business and engineering is a key factor and demands to be maintained. Each of these two dimensions is necessary and significant part of the software engineering model. Empowerment of only one poses the risk of subverting the other. Standard practices contribute to the engineering dimension. This work proposes a framework inspired

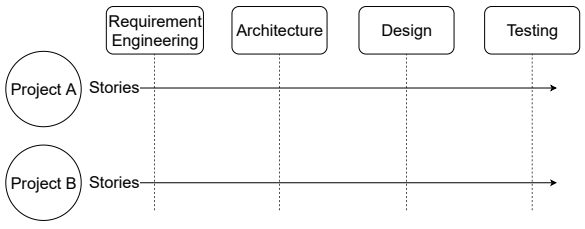

Figure 4. Framework based on matrix structure

by the Matrix organization structure [11] as a solution to attain a balance. The vertical dimension of this matrix represents engineering and the horizontal business. As shown in Figure 4 the functional units of the matrix are based on the phases of software engineering. The task of the design unit subsumes

the coding activities. There are standard design artifacts that can be used by tools to partially generate source code. Further improvement in design practices can maximize the code generated by tools. In this structure, instead of the project, functional units serve as the home base of engineers in it. The units focus on the application of engineering solutions. The member of a unit may work on tasks from multiple projects in context of the unit's engineering function. Implementation of stories is realized by the contribution of units in their relevant area.

The two dimensional structure brings forth the significance of engineering aspect by giving it the position of a body in the development process. This empowerment, although necessary to motivate the culture of engineering practices, is not sufficient to provide a steadfast platform for standard practices to operate and mature. This firm foundation can be established by using elements of Rational Unified Process (RUP) where artifacts are essential outcomes of activities. In RUP, artifacts are the tangible products of the project, the things the project produces or uses while working towards the final product [12]. In the matrix structure, the artifact produced by a standard method is a prerequisite for another as shown in Figure 3. This in effect makes artifacts form the operational interface between the functional units. The inclination towards an artifact based approach creates a sustainable demand for the use of standard practices. The possibility to incorporate standard practices into the development process, withstanding the opposing factors, by virtue of the design of the framework is the main contribution of this work.

The framework, by design, aims to meliorate the area of *Standard practices* but due to the presence of focused functional units the practices are also cultured and mature over time. In a matrix structure, when teams of functional specialists work together, a synergistic effect occurs, resulting in increased innovation and productive output, even though individually they may be working on different projects [11]. In addition to incorporating methods there are other notable benefits in the area of *Advancement goals* and *Engineers*. Re-usability is an attribute that is desired for the advancement of engineering and since the units serve multiple projects in context of a specific functional area there is a drive towards unifying the knowledge and attaining re-usability in the process. Growth of engineers comprises learning and a desired mobility in the organization. A functional unit being specific to a discipline in software engineering, i.e., design, architecture, requirement provides the opportunity to attain both learning and mobility to desired units for engineers.

The abstract framework proposed in this work pose certain potential risks that, although not established a priori or empirically, demand attention. The project domains in the organization have a wide spectrum from embedded application to mobile and web applications. The complexity of carrying out the activities of these diverse domains by one unit poses a challenge in the development and so does the overhead of making a change in the development model across all projects. The *Business adaptability* factor suggest that project may have different constraints and demands for the rigor of practices. Not all projects demand the rigor of formal methods because the cost of error may not be as high as it is for safety-critical system thus posing a challenge in standardizing the practice in a unit across all domains. The next subsection provides a brief account of a simple alternative.

B. Alternate solution

Addressing the challenges in matrix structure there can be alternative solutions based on incentives. In this model instead of the structure empowering the engineering aspect by design, engineers are motivated to use standard practices by incentivisation. The incentives can be in a form that contributes to their performance and growth. The application of a practice by the engineer can be evaluated by an independent body based on certain guidelines. The benefit of this structure is the lack of overhead to the current process and this can be applied across all domains. However, a major pitfall is that the key point of balance between business and engineering is not addressed and is left as a choice that the incentive may fail to influence in favor of engineering. In contrast to the matrix structure, the incentive solution does not directly provide the benefits of maturity of practices and re-usability due to lack of focused functional units.

C. Candidate solution

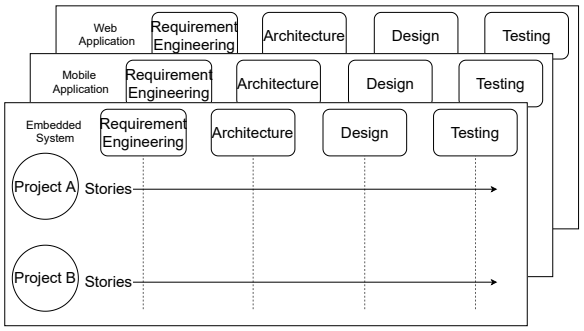

Figure 5. Domain based matrix structure

The directed and firm nature of the matrix structure towards addressing the *Motivation* and *Sustainability* aspects augmented by its contributions to *Advancement* and *Engineers* makes it a desirable solution. The aforementioned risks arising from *Project Management* and *Business adaptability* can be alleviated by applying the structure to individual domains instead of pursuing it across all projects. The domain based matrix structure retains the canonical form of the matrix but operates under the umbrella of a particular domain e.g., embedded systems, web applications. The simple modification as shown in Figure 5, apart from reducing the risk posed by the scale of change and multifariousness of domains, provides an additional advantage of domain specific re-usability that can lead to advancements like *Domain Engineering*.

D. Guidelines for selection of practices

The candidate solution aims to provide an environment to nurture standard practices. However, its effectuation demands a set of selected practices. The vast number of variables and choices available in different domains makes the selection of practices a challenging task. The engineering domain model presented in this work allows eliciting some guidelines that can help in the selection process. A comprehensive and systematic selection process would require consideration of various factors including quantitative valuation of the criteria, their precedence order and the procedural aspects of the process of selection. Such a study is not in the scope of this work.

 ISBN: 978-1-61208-776-4

TABLE I. PRACTICE SELECTION GUIDELINES

Association	*Criteria*
Quality metrics	Availability of desired quality metrics. The metric may correspond to a recognized software quality standard e.g., ISO/IEC 9126
SDLC	Produces artifacts that can be used by other practices. This can produce an artifact based workflow. Available tool support.
Training	Affordability and availability of resources for training
Engineers	Recruitment - Availability of experienced professionals with the competence required for the practice. Growth - Practice is recognized industry wide contributing to the growth of employees.
Business adaptability	Applicable directly or indirectly to wide range of business domains.
Project Management	Cost and turnaround time of the practice
Advancement goals	Assists the realization and maturity of concepts like automation and reuse that helps to lay the foundation for advancement of engineering

However, the points described in Table I can reasonably guide the selection process.

V. CONCLUSION AND FUTURE WORK

The current software engineering model in the organization successfully caters to the business needs in the software development process. However in order to sustain high level of quality irrespective of the size and complexity of the project, the model must heed the engineering aspect. Introduction of a set of standard engineering practices is an important step in that direction. This work attempts to lay a foundation to build the set of practices and proposes a framework to incorporate them in the software development process.

The premise of the work is a software engineering domain model that provides a landscape to elicit the set of guidelines for selection of standard practices and derive two solutions. Based on the benefits and drawbacks of each, one of the solutions is modified to reach a candidate for further empirical evaluation. The analysis and elimination process used to derive the solution strengthens the logical soundness of the approach. The solution proposed is a framework based on the matrix organization structure applied to specific domain. The framework notably empowers the engineering aspect of the development process thus providing a platform to apply and standardize engineering practices. The framework in its current state is a proposal and lacks the rigor of a theory. Thus, it does not produce a comprehensive set of testable hypothesis at this stage.

The work gives rise to two distinct lines of research to pursue in future. Firstly, the theory building process must be applied to the proposed framework, which includes the delineation of the term standard practice [13]. The formulated theory will provide the foundation for practical evaluation. Secondly, an empirical research strategy needs to be designed and conducted to test the hypothesis drawn from the formulated theory [14]. A test can provide the necessary experimental data required to establish the validity of the proposed framework. The scope of the preliminary empirical evaluation shall be confined to a single domain e.g., embedded system.

REFERENCES

[1] S. Sheppard, A. Colby, K. Macatangay, and W. Sullivan, "What is engineering practice?" International Journal of Engineering Education, vol. 22, no. 3, 01 2006, pp. 429–438.

[2] W. Humphrey, "Introduction to software process improvement," Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU/SEI-92-TR-007, 1992. [Online]. Available: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=11611 [retrieved: 01,2020]

[3] Tr, "Information technology — software process assessment — part 2 : A reference model for processes and process capability." ISO, 1998.

[4] G. Valdés, M. Visconti, and H. Astudillo, "The tutelkan reference process: A reusable process model for enabling spi in small settings," in Systems, Software and Service Process Improvement, R. V. O'Connor, J. Pries-Heje, and R. Messnarz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 179–190.

[5] I. Richardson and C. G. Von Wangenheim, "Guest editors' introduction: Why are small software organizations different?" IEEE Software, vol. 24, no. 1, Jan 2007, pp. 18–22.

[6] E. Allman, "Managing technical debt," Queue, vol. 10, no. 3, Mar. 2012, p. 10–17. [Online]. Available: https://doi.org/10.1145/2168796.2168798 [retrieved: 01,2020]

[7] K. S. Rubin, Essential Scrum: A Practical Guide to the Most Popular Agile Process, 1st ed. Addison-Wesley Professional, 2012.

[8] E. Ras, J. Rech, and B. Decker, "Workplace learning in software engineering reuse," in Proc. Int. Conf. Knowledge Management, Special Track: Integrating Working and Learning, 2006, pp. 437–445.

[9] M. T. Baldassarre, A. Bianchi, D. Caivano, and G. Visaggio, "An industrial case study on reuse oriented development," in 21st IEEE International Conference on Software Maintenance (ICSM'05). IEEE, 2005, pp. 283–292.

[10] D. Turk, R. France, and B. Rumpe, "Limitations of agile software processes," In Proceedings of the Third International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP2002), 05 2002, pp. 43–46.

[11] L. C. Stuckenbruck, "The matrix organization." Project Management Quarterly, vol. 10, no. 3, 1979, pp. 21–23.

[12] R. U. Process, "Best practices for software development teams," A Rational Software Corporation White Paper. TP026B, Rev, vol. 11, no. 01, 2001.

[13] D. I. Sjøberg, T. Dybå, B. C. Anda, and J. E. Hannay, "Building theories in software engineering," in Guide to advanced empirical software engineering. Springer, 2008, pp. 312–336.

[14] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting empirical methods for software engineering research," in Guide to advanced empirical software engineering. Springer, 2008, pp. 285–311.

 ISBN: 978-1-61208-776-4