Chapter 3
Disciplined Teams vs. Agile Teams:
Differences and Similarities in Software Development

Antonio Alexandre Moura Costa
Federal Institute of Paraiba, Brazil

Felipe Barbosa Araújo Ramos
https://orcid.org/0000-0002-0937-811X
Federal Institute of Paraiba, Brazil

Dalton Cézane Gomes Valadares
Federal University of Campina Grande, Brazil & Federal Institute of Pernambuco, Brazil

Danyllo Wagner Albuquerque
Federal University of Campina Grande, Brazil

Emanuel Dantas Filho
Federal University of Campina Grande, Brazil

Alexandre Braga Gomes
Federal University of Campina Grande, Brazil

Mirko Barbosa Perkusich
VIRTUS, Brazil

Hyggo Oliveira de Almeida
Federal University of Campina Grande, Brazil

ABSTRACT

Software development has been considered a socio-technical activity over the past decades. Particularly, in the case of software engineering, the necessity to communicate effectively with stakeholders and team members has been progressively emphasized. Human resources play a critical role in the success of software projects. Many techniques, methods, tools, models, and methodologies have been proposed, applied, and improved in order to help and ease the management of the software development process. Regardless of the software development methodology adopted, delivering a quality product in a predictable, efficient, and responsive manner is the objective for every team. Disciplined and Agile teams have different characteristics, but also share common aspects when working to accomplish their goals. The main motivation of this chapter is to present the differences and similarities of both teams in the context of software development.

DOI: 10.4018/978-1-6684-3702-5.ch003

Copyright © 2022, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

INTRODUCTION

Delivering high-quality products in time and without budget overrun is still a significant struggle for most software organizations. Among the most common reasons are inaccurate estimates of needed resources, unmanaged risks, sloppy development practices, poor project management, commercial pressures, among others (Charette, 2005; Macnab & Doctolero, 2019). Software projects failures have a major negative impact on the organizations and can deeply compromise its future.

Projects continue to proliferate in society today, in both the public and private sectors of the economy. Investments in projects number in the trillions of dollars annually. Just as ubiquitous as these projects, unfortunately, are their significant failure rates. The CHAOS reports have identified the current state of project success rates across organizations, noting that in spite of much higher visibility and importance placed on project performance, failure rates have remained high and relatively stable across over a decade of research. (Serrador & Pinto, 2015)

When a software project fails, it jeopardizes an organization's prospects. If the failure is large enough, it can steal the company's entire future. In one stellar meltdown, a poorly implemented resource planning system led FoxMeyer Drug Co., a $5 billion wholesale drug distribution company in Carrollton, Texas, to plummet into bankruptcy in 1996. (Charette, 2005)

For a long time, Disciplined approaches had been used to increase project success. These approaches, also known as plan-driven or heavyweight methodologies, are conducted in a linear way, whereas process activities of specification, development, validation, and evolution must be performed in sequential order, which means that an activity must be completed before the next one begins. Due to the heavy aspect of the Disciplined approaches, several consultants have developed methodologies and practices with more emphasis on people, collaboration, customer interaction, and working software, rather than on processes, tools, documentation, and plans. These approaches, known as Agile methods, have been gained popularity over the past years, especially after the advent of the Agile Manifesto in 2001 (Fowler & Highsmith, 2001). Unlike Discipline approaches such as the Waterfall model, the Agile ones promote continuous iteration of development and testing throughout the software development lifecycle.

Agile methods are based on adaptive software development, while Disciplined approaches are derived from the predictive approach. Disciplined teams work with a detailed plan and have a complete list of requirements that must be implemented in the next months or in the entire lifecycle of the product. Predictive approaches entirely depend on the requirement analysis and detailed planning at the beginning of the cycle. Any change must go through a rigorous change control management and prioritization. In adaptive methods, there is no extensive planning and only clear tasks are related to the features that must be developed. Agile teams are more adaptable to dynamic changes in product requirements. Communication and minimal documentation represent ordinary characteristics of agile software development. Also, customer interaction is another strong suit for these approaches. While Agile methods are indicated for small and medium projects, disciplined approaches are more suitable to the large ones.

Once Disciplined and Agile approaches are highlighted by different aspects, their teams also have different characteristics regarding size, roles, planning, executions, and others. For instance, Disciplined teams are composed of dozens of professionals, not necessarily co-located, usually divided into teams of specialists. For example, the project may have a programmer team, a quality assurance team, and a design

14 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/disciplined-teams-vs-agile-teams/294457

Related Content

A Novel Sentence Completion System for Punjabi Using Deep Neural Networks
Gurjot Singh Mahi and Amandeep Verma (2022). *International Journal of Software Innovation (pp. 1-25).*
www.irma-international.org/article/a-novel-sentence-completion-system-for-punjabi-using-deep-neural-networks/293271

Model-Driven Data Warehouse Automation: A Dependent-Concept Learning Approach
Moez Essaidi, Aomar Osmani and Céline Rouveirol (2014). *Advances and Applications in Model-Driven Engineering (pp. 240-267).*
www.irma-international.org/chapter/model-driven-data-warehouse-automation/78618

Recommendations for Conducting Software Reviews
Yuk Kuen Wong (2006). *Modern Software Review: Techniques and Technologies (pp. 268-280).*
www.irma-international.org/chapter/recommendations-conducting-software-reviews/26908

To Prevent Reverse-Enginnering Tools by Shuffling the Stack Status with Hook Mechanism
Kazumasa Fukuda and Haruaki Tamada (2015). *International Journal of Software Innovation (pp. 14-25).*
www.irma-international.org/article/to-prevent-reverse-enginnering-tools-by-shuffling-the-stack-status-with-hook-mechanism/126613

An Incremental Model Projection Applied to Streamline Software Architecture Assessment and Monitoring
Salim Kadri, Sofiane Aouag and Djalal Hedjazi (2021). *International Journal of Information System Modeling and Design (pp. 27-43).*
www.irma-international.org/article/an-incremental-model-projection-applied-to-streamline-software-architecture-assessment-and-monitoring/285952