NOVA IMS Information Management School

MGI

Master's Degree Program in
Information Management

Data-Driven Decision-Making: Customizing Agile Development Application in ServiceNow for Enhanced Management Insights

Ricardo Raab Saenger

Project Workpresented as a partial requirement for obtaining a Master's Degree in Information Management

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

DATA-DRIVEN DECISION-MAKING: CUSTOMIZING AGILE DEVELOPMENT APPLICATION IN SERVICENOW FOR ENHANCED MANAGEMENT INSIGHTS

by

Ricardo Raab Saenger

Project Work presented as a partial requirement for obtaining a Master's Degree in Information Management, with a specialization in Knowledge Management and Business Intelligence.

Supervised by

Prof. Vítor Duarte dos Santos, PhD, Nova IMS

July, 2024

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism, any form of undue use of information, or falsification of results along the process leading to its elaboration. I further declare that I have fully acknowledged the Rules of Conduct and Code of Honor from the NOVA Information Management School.

Lisbon, 2024.

ACKNOWLEDGMENTS

First of all, I would like to express my gratitude to my parents for giving me the opportunity and motivation to come to Portugal to complete this Master's degree. You and my sister represent the best of a family in terms of love, support, encouragement, and dedication. Thank you for always being close, even on another continent, and for supporting my crazy dreams. Without your unconditional love, I would be nothing.

I would like to thank my Portuguese family, especially my girlfriend, for always being there for me. Thank you for all your support and for spending some weekends at home with me while writing this project. I love you. I would also like to thank my parents-in-law and my father-in-law for his guidance in writing a scientific paper. I am also grateful to all my friends and colleagues who have become part of this incredible Portuguese family I cherish deeply.

I want to give a special thanks to the company manager I work for, who helped me design the technical idea behind this project and provided the flexibility I needed to develop the project while working. I also want to thank my teammates for their tips and advice on code development. Your help was crucial to the success of this project.

Finally, I would like to thank my advisor for the knowledge and experience you have shared with me. Thank you for being available to answer my questions, for encouraging me, and for believing in my abilities. Your willingness to help me revise the project resulted in the acceptance of the paper we submitted for publication at the conference. I would also like to acknowledge the crucial role of NOVA Information Management School in shaping my academic journey.

ABSTRACT

Embracing digital transformation and process automation is critical to managing organizations' growing amounts of data. This allows for informed decision-making through easy access to the correct data, as made possible by platforms like ServiceNow. This cloud-based platform uses an Information Technology Service Management (ITSM) system to manage IT services and automate business processes. However, ServiceNow's Agile Development 2.0 application lacks the capability to monitor time allocation and quality metrics during story development. Therefore, the main objective of this project was to create customizations that add new features and capabilities to the Agile Development 2.0 application in ServiceNow. Automation was designed to power these customizations and create a master dashboard to achieve the project goal. This dashboard illustrates performance and quality at each stage of a story's development. It provides a real-time view of the metrics and information collected through the customization package, enhancing decision-making based on data.

KEYWORDS

Performance Metrics Dashboard; ServiceNow Customization; ITSM Automation; Time Allocation; Scrum

PUBLICATIONS RESULTING FROM THIS PROJECT

Saenger, R. & Santos, V. (2024). Data-Driven Decision-Making: Customizing Agile Development Application in ServiceNow for Enhanced Management Insights. The 16th Mediterranean Conference on Information Systems (MCIS 2024) and the 24th Conference of the Portuguese Association for Information Systems (CAPSI 2024).

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

API	Application Programming Interface
ATF	Automated Testing Framework
BR	Business Rule
CS	Client Script
DSDM	Dynamic System Development Model
FDD	Feature-Driven Development
FS	Fibonacci Sequence
ISO	International Organization for Standardization
IT	Information Technology
ITIL	Information Technology Infrastructure Library
ITSM	Information Technology Service Management
KM	Knowledge Management
PDI	Personal Developer Instance
RS	Ricardo Saenger
SVS	Service Value System
TDD	Test-Driven Development
XP	Extreme Programming

1. INTRODUCTION

1.1. CONTEXT

In today's business world, companies face many data management and compliance challenges. With technological advancements, businesses must be agile and utilize digital transformation and process automation to succeed. In this regard, digital transformation becomes an important way to improve efficiency, competitiveness, and responsiveness.

Another situation is that the amount of data businesses generate keeps growing, making it more challenging for leaders to extract insights and make informed decisions. To overcome this obstacle, reliable access is needed to the correct data in an easy-to-understand format. Within this, dashboards have become popular due to their ability to present data clearly and concisely, making it easier for business leaders to make fact-based decisions quickly (Pappas & Whitman, 2011).

Likewise, proper data management can help users acquire added value, such as accurate information, to support the decision-making process, enhancing operational efficiency and effectiveness (Akbar et al., 2020). Following this thinking, IT Service Management (ITSM) collects and analyzes data to make reports and implement improvements agilely. However, managing these organizational assets effectively can be challenging (Sahid et al., 2020).

In this scenario, intelligent platforms started to emerge, enabling the automation of repetitive tasks and empowering organizations to make informed decisions based on data, streamline operations, and provide unparalleled customer experiences. This is the case of ServiceNow, a cloud-based platform used by the IT Help Desk, which uses an ITSM system to handle various aspects of IT Services and automate business processes (Gouryraj et al., 2021).

In terms of methodologies or frameworks for software development, many companies use the Agile methods to develop products or services more efficiently, reducing the time needed to complete this process (Hinderks et al., 2022). Moreover, Srivastava et al. (2020) explain that Scrum is the most widely applied of the current Agile methods.

The project centers on installing the Agile Development 2.0 application in a ServiceNow instance and creating customizations that will be used as an acceleration package to optimize the usability process, data collection, understanding, and visualization. These customizations will focus on tracking the time spent in the stages of a story and gathering information about its development issues, as well as the documentation and other information in the main stages of a story.

Regarding market perspectives, Pang (2022) concluded that the ITSM application market is about to reach $9 billion by 2026, compared with $7.7 billion in 2021 at a compound annual growth rate of 3.2%. ServiceNow is the leading application for IT Service Management

software vendors. In 2021, it led the pack with a 40,1% market share, riding on a 23,6% jump in ITSM license, maintenance, and subscription revenues.

I am a Professional Scrum Master and certified as an administrator, developer, and implementer of the ServiceNow platform. My passion for the IT world has driven me to create this project, which aims to enhance the functionality of ServiceNow's Agile Development application in terms of time tracking, reporting, and data visualization.

In this project, with the theoretical framework and the support of ServiceNow's technical documentation, I have gathered the knowledge necessary to implement the proposed improvements and customizations, following the methodology and best practices described by the platform.

1.2. PROBLEM IDENTIFICATION

Currently, the Agile Development 2.0 application on the ServiceNow platform does not offer a simple, ready-to-use solution for monitoring the time allocated during the story development process and other quality metrics.

As a result, there are no metrics to monitor the time spent in each development stage of a story so that the management layer and the Scrum master can understand how the teams work and perform. This makes it challenging to keep track of team tasks and assess the accuracy of their work. In addition, some useful fields and information are missing from the story form, like documentation attachments, informative messages, and quality measurement fields.

As per Chauhan et al. (2023), estimating software work is crucial for software project management. The evolving nature of software development compounds the difficulty of estimating effort. This indicates a need for a more practical and assertive way of understanding the efforts expended by teams during software development.

To address this problem, customizations have been proposed for the platform through an acceleration package that will add new features and capabilities to the Agile Development 2.0 application in ServiceNow and improve some existing ones by taking advantage of the platform's cross-scoped characteristics.

Thus, this project aims to help answer the following questions:

1) What are the benefits of using the Agile Development 2.0 application on ServiceNow?
2) How can we evaluate work efficiency regarding the time spent on each story development stage, from the draft point to the final implementation?
3) How can we present data related to Agile development, including time tracking and issue identification, in a user-friendly way for analysis?

1.3. OBJECTIVES

Created as an acceleration package, this project will make it easier to use and extract data from the Agile Development 2.0 application in any company that already uses a ServiceNow instance. Moreover, facilitating visualization of story-related data will help management make data-driven decisions.

To reach this purpose, some milestones were defined as objectives:

1) Develop a customization package to be used as an accelerator. This package will include new functionalities and improvements in ServiceNow's Agile Development 2.0 application.
2) Create automatisms to boost these customizations, combining the functionality of the new fields with informative messages and knowledge articles.
3) Create a dashboard that illustrates performance and quality at each development stage of a story and, through interactive filters, offer a real-time view of the metrics and data collected through the proposed customization package.

These milestones provide a path, divided into phases, to guide the achievement of the main objective.

1.4. EXPECTED RESULTS AND CONTRIBUTIONS

The proposed customization package is designed to increase the capabilities of the Agile Development 2.0 application in ServiceNow, presenting an adaptable solution that any organization can use. This acceleration package will work as a facilitator to construct a dynamic dashboard, offering a complete view of the development efficiency in terms of time worked and issue tracking throughout the project lifecycle, from conception to implementation and testing.

One advantage of this customization package is its capability to undertake performance evaluations among different teams by leveraging real-time data. This feature facilitates the identification of undersized or oversized teams, allowing the management team to make more informed decisions and allocate resources.

Furthermore, it also answers key questions pertinent to project developments, such as individual work hours per sprint or assertiveness of a determined type of testing in a more critical development. The expected outcome will empower organizations with easy-to-understand analytical tools and a better understanding of team dynamics, fostering a culture of continuous improvement and informed decision-making in project development.

2. WORK PLAN

This chapter defines the project roadmap, from design and implementation to final testing and evaluation of results. It also presents the tools used to develop the project and the proposed implementation schedule.

2.1. PROJECT PHASES

To ensure the successful implementation of the project and for organizational purposes, it was divided into five phases, from its conception and preparation of the platform to the final tests and evaluation of the achieved results. These phases are illustrated in Figure 2.1.

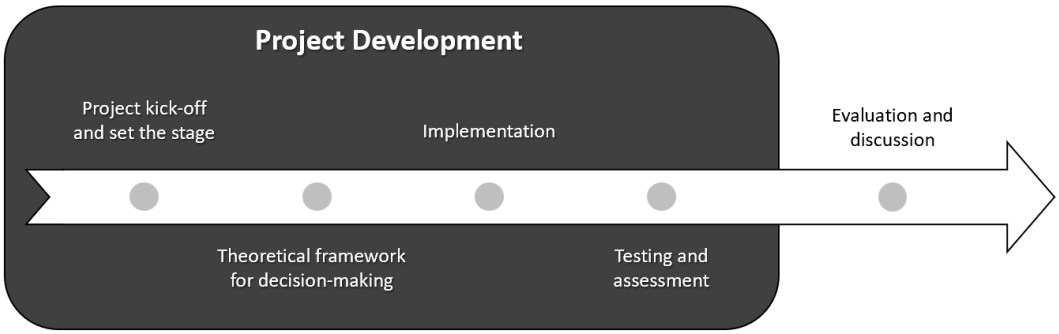


Figure 2.1 – Project management phases

Source – Prepared by the author

2.1.1. Phase 1: Project kick-off and set the stage

This initial phase is where the master plan for implementing the project is stipulated, as well as everything necessary to follow the best practices defined by ServiceNow, thus achieving the proposed objectives.

The first step is to initiate the project, which involves setting up the ServiceNow instance to begin functioning. ServiceNow employs a single-tenant architecture, meaning that every developer is provided with personal or multiple copies of ServiceNow in the cloud, as explained by Toulson (2019). These copies are known as ServiceNow instances. Also, the platform recommends installing the Agile Development 2.0 plugin directly from the ServiceNow App Store into the instance to ensure proper functionality.

After installing the application, validations will be necessary to ensure all required components are present and performing as expected.

2.1.2. Phase 2: Theoretical framework for decision-making

During this phase, the project utilizes a literature collection framework, called a theoretical framework, to support decision-making. A thorough analysis of relevant topics is conducted, increased by a presentation of the theoretical foundation.

The theoretical framework will gather knowledge from the academic community on the most relevant aspects of the Agile methodology, its main frameworks, capabilities, and application advantages. In addition, information will also be gathered on ITSM, its applications, capabilities, and advantages, as well as ISO 20000 and ITIL, the most popular ITSM frameworks. This chapter will end with definitions and concepts related to dashboards and data visualization, which will be used as guidelines to build the dashboard proposed in this project.

It is important to consider that this phase will undergo continuous development alongside the project, as its outcomes will validate some project decisions.

As these are new customizations to the platform, it is necessary to store this information in a way that can be shared with everyone in the organization. For this, the ServiceNow Knowledge Management (KM) application allows the exchange of information in knowledge bases. These knowledge bases hold articles that offer users information like self-help, troubleshooting, and task resolution (ServiceNow, 2022b).

Based on this, parts of the literature review will be turned into articles on the platform and made available to users. This allows knowledge of the scientific part of the project to be combined with the practical and technical part, making the knowledge base an added value and another outcome of this project.

2.1.3. Phase 3: Implementation

The implementation phase is the most complex and critical, as it is here that the entire process of implementing the customizations will be detailed. This phase will be split into sub-phases to better divide and understand the work.

During this stage, the customization process of the Agile application in ServiceNow will begin. The primary focus will be creating the required fields, defining their data types, and specifying their behavior. Furthermore, rules will be developed for these fields, such as making them mandatory or appearing/hiding based on the information displayed in other fields.

To accomplish this, the capabilities of the platform, such as business rules and scripts are utilized. These run on the server side and facilitate the creation of rules for the form and the operation of the application as a whole.

2.1.4. Phase 4: Testing and assessment

This is the phase when development testing and quality assurance begin. ServiceNow provides an automatic testing framework, ATF, described in section 2.2.3. This framework enables the creation of necessary steps for accurately testing all developments. Once the ATFs are defined, the steps will run automatically.

This phase of the project is equally important, as it validates the developments that have been made and helps to assess the functioning of the customizations, allowing for the identification of improvements in the developments.

2.1.5. Phase 5: Evaluation and discussion

This phase marks the end of the project and will occur after all the developments have been implemented and thoroughly tested. The purpose of this phase is to assess the project holistically. This assessment helps to compare expected and actual outcomes and identify areas for future improvement.

In this context, the customizations will be evaluated for accuracy and functionality during the tests and through a questionnaire distributed to certified users of the ServiceNow platform.

2.2. TOOLS

As this project revolves around the ServiceNow platform and its ITSM capabilities, combined with the Agile Development 2.0 application and the Automated Test Framework (ATFs) solution, the main application to be used is a ServiceNow instance. As explained, a ServiceNow instance is a copy or multiple copies of ServiceNow in the cloud (Toulson, 2019).

According to Toulson (2019), ServiceNow can run several applications concurrently within a single instance. This single instance can function as a centralized record system for various business applications, such as IT Service Management (ITSM), Human Resources Service Delivery (HRSD), Customer Service Management (CSM), and Custom Apps, all under the same base URL. As a result, sharing data between applications becomes easier, enhancing the functionality of each application.

In this context, ServiceNow offers its registered users a free Personal Development Instance (PDI). This instance can be used to develop and customize applications on the platform, improve the user's skills, and test the new features available (ServiceNow, 2023c). The PDI used in this project runs the latest version of ServiceNow's instance, called Vancouver, which the Now Platform made available in September 2023.

Furthermore, according to Pang (2022), ServiceNow is the most used ITSM platform. The next sections will explain ServiceNow solutions, modules, and applications used during the project.

2.2.1. ServiceNow and the IT Service Management (ITSM)

ServiceNow is a smart platform built to promote digital transformation through a user-friendly unified system called the Now Platform, used as the main component of the platform, bringing together solutions in IT, Operations, Customer Service, HR, and more (ServiceNow, 2023e).

As stated by ServiceNow (2023e), it is a software-as-a-service cloud-based platform that can automate tasks and processes across a company. It promotes efficiency and optimization through customizable and low-code tools, making it easy for businesses to adapt and scale up.

These tools allow enterprises to accelerate the digitalization of processes, reducing complexity and optimizing them through the automation provided by the platform's workflows, artificial intelligence, and machine learning.

This project focuses on the IT Service Management (ITSM) component, the main and most used solution of the Now Platform. ITSM provides accessible workflows to manage and deliver IT services to its users, increasing agent productivity and user adoption of the platform. Users can access the ITSM solution via mobile or web portal interfaces (ServiceNow, 2023b).

2.2.2. ServiceNow Application: Agile Development 2.0

ServiceNow Agile Development 2.0 provides an Agile software development environment that helps manage software development efforts that can be product-based or project-based. It also supports hybrid efforts using Agile methods within a project structure through the Scrum framework defined by Scrum Alliance (ServiceNow, 2023a).

This application allows a pure Agile implementation over the entire life cycle of a product or project. Based on the ServiceNow platform, it is fully integrated with products to manage IT services and operations (ITSM).

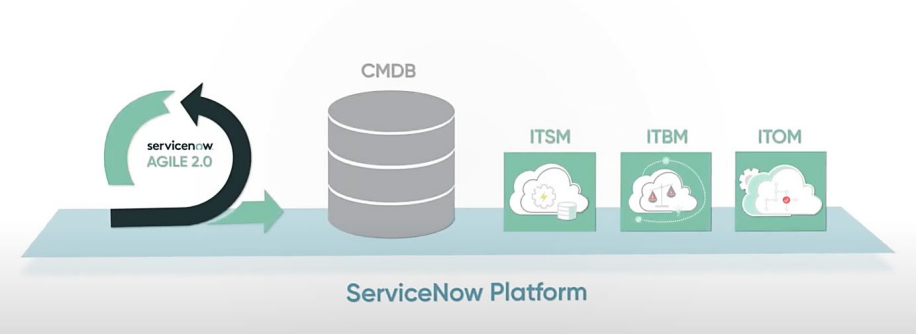


Figure 2.2 – Agile in the ServiceNow platform

Source – Adapted from ServiceNow (2023)

ServiceNow supports all the elements of the Scrum framework, including:

- Workload: The product backlog is divided into release backlogs, themes, and epics with prioritized stories.
- Personnel: Developers are organized in Agile groups led by a Scrum Master who oversees the implementation of stories in a sprint. The Product Owner manages the product and releases backlogs.
- Activities: The team creates stories for products or projects and adds them to product backlogs.

2.2.3. ServiceNow Feature: Automated Test Framework (ATF)

In line with ServiceNow documentation, a test is a logical grouping of steps that verify a specific functionality or feature. With the Automated Test Framework (ATF), it is possible to

create and run automated tests to ensure that the instance functions properly after making any changes. In case of failed test results, it is also possible to identify the changes that caused the failure and the necessary changes to review.

It is important to note that ATFs offer various benefits, such as the ability to design tests once and reuse them in different contexts and with different test data sets, schedule test suite runs, reduce test design time by copying quick start tests and test suites, and expand test coverage by creating custom test steps (ServiceNow, 2022a).

2.3. CHRONOGRAM

The chronogram of this project is represented in Figure 2.3 as a summary of the execution schedule of the project. It outlines the five phases described earlier with their respective milestones and deliverables.

		2023			2024						
Phase	Activity	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul
1	Project proposal										
	Introduction										
	Set the stage										
2	Theoretical framework										
3	Implementation										
4	Testing										
	Assessment										
5	Evaluation and discussion										
	Conclusion										
	Final adjustments										
	Advisor's review										
	Milestones		M1		M2		M3	M4		M5	M6
	Deliverables		D1				D2	D3		D4	D5

Milestones

M1 - Finish the theoretical part of the project
M2 - Finish the first three chapters of the project
M3 - Customizations fully implemented
M4 - Customizations tested and evaluated
M5 - Article created to be published
M6 - Article accepted for publication

Deliverables

D1 - Project proposal
D2 - First three chapters delivered
D3 - Project chapter delivered
D4 - Article submitted in a conference
D5 - Project submitted

Figure 2.3 – Project chronogram

Source – Prepared by the author

3. THEORETICAL FRAMEWORK

In this chapter, and to achieve the established objectives, the knowledge that the scientific community has regarding the topics dealt with in the project can be found. This knowledge has been used as a guideline for the decisions made throughout the development of the project.

This chapter begins with a definition of software development and engineering, which, according to Giuffrida & Dittrich (2015), is a cooperative work that must be coordinated by the developers, combining their individual activities with the tasks performed by the other team members. In this context, the authors also state that communication is fundamental in cooperative work, mainly driven by face-to-face communication to enhance effectiveness in coordination.

3.1. AGILE DEVELOPMENT

Agile has gained worldwide popularity in the software development industry (Itzik & Roy, 2023), expanding even in organizations that rely on traditional or structured software development methods (Mishra & Alzoubi, 2023). Its success is mostly due to its ability to cope with requirements and adapt to change (Itzik & Roy, 2023).

According to surveys conducted by Serebryantseva (2022), 71% of companies chose Agile as a framework for software development, taking advantage of its intensive team collaboration, flexibility for changing requirements, fast bug detection, and product delivery.

3.1.1. Overview

Agile is a broad set of software development theories with a lightweight approach to overcome the limitations of conventional development methods, allowing the adoption of changes in requirements at any stage of development (Jammalamadaka & Krishna, 2013) by managing the project's tasks and activities according to a set of values and principles (Al-saqqa et al., 2020).

The values and principles provide the basis to guide the software development process, as per Rodríguez et al. (2019), which originated the Agile Software Development Manifesto, an official statement of principles formulated by a group of seventeen software-processing methodologists in 2001, to promote a better way of developing software (Itzik & Roy, 2023). The manifesto states, "*Agile uncovers better ways of developing software by doing it and helping others do it.*" (Beck et al., 2001).

In this context, the Agile Manifesto highlights four values. When faced with a decision, priority should be given to what is on the left side of each value rather than those on the right (Beck et al., 2001).

1. Individuals and interactions over processes and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan.

In addition to these values, the Agile Manifesto also establishes 12 principles, as follows:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for a shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need and trust them to do the job.
6. Face-to-face conversation is the most efficient and effective method of conveying information to and within a development team.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximizing the work not done – is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on becoming more effective, then tunes and adjusts its behavior accordingly.

These values and principles offer a foundation for guidance, not imposing the operational details of the software development process. The main goal is to enhance the software development process with responsiveness, flexibility, capability, and speed (Rodríguez et al., 2019).

Moreover, Agile itself is not purely considered a methodology. It is a set of practices, values, and principles. Its main characteristic is being adaptable to changes, thanks to its incremental delivery component, reducing costs and time (Choudhary & Rakesh, 2016).

3.1.2. Agile development frameworks

Agile methods use different combinations of practices to describe the daily tasks of software developers. The methodological differences result from the choice of specific terminology and practices adapted to each approach (Elbanna & Sarker, 2016). Although not completely new to the software development industry, these practices provide better results when applied under Agile values and principles (Anwer et al., 2017).

Al-saqqa et al. (2020) explain that there are several different Agile frameworks, including Extreme Programming (XP), Scrum, Dynamic System Development Model (DSDM), Test-Driven Development (TDD), Feature-Driven Development (FDD), and Crystal. Each has its own life cycle, advantages, and disadvantages, and all of them build the final software using iterations and incremental processes. The main ones are explained below.

1. Extreme programming (XP): This was one of the first Agile methods proposed by Kent Beck back in the 2000s. The main goal of Kent and Andres (2005) was to develop a methodology that could be used by multiple programmers in a single location. Choudhary and Rakesh (2016) explain that this methodology aims to minimize the cost of requirement changes by replacing a lengthy development cycle with multiple short cycles to enhance customer satisfaction.
   On the other hand, Abrahamsson et al. (2017) describe a development process centered on customer input, compact teams, and daily builds. The system is continuously redesigned for enhanced performance and adaptability to change.

2. Feature-driven development (FDD): This method manages short incremental iterations, leading to functional software (Al-saqqa et al., 2020). In this context, Anwer et al. (2017) explain that FDD aims to manage software development based on business needs through a feature-driven approach. It's a highly adaptive software development method, accommodating late changes in requirements and prioritizing high-quality outputs throughout the development phases.
   As Abrahamsson et al. (2017) stated, the FDD life cycle comprises five sequential incremental and iterative processes to deliver the final software. Anwer et al. (2017) say that one of the advantages of this method is that it is highly adaptable, accepting late changes by the customer while still delivering a great result after each phase.

3. Dynamic System Development Model (DSDM): The main idea of this method is similar to the others presented. It includes rapid application development with incremental approaches, valorizing the quality of the software (Al-saqqa et al., 2020).
   The core concept of DSDM is first to set project resources and time constraints, adjusting the achievable functionality within those limits. This differs from traditional methods, where functionalities are predefined first, and then time and resources are allocated based on those predetermined requirements (Abrahamsson et al., 2017; Anwer et al., 2017). The authors also point out that this methodology presents guidelines for project management, risk control, and development techniques. On the other hand, the large number of roles required in the DSDM can lead to some difficulties regarding the administration of the project.

4. Test Driven Development (TDD): This methodology begins differently from the other ones, as it aims to write an automated test first followed by a small code which, once passed the test, can later be improved (Al-saqqa et al., 2020; Anwer et al., 2017). In TDD, Anwer et al. (2017) say that the test is performed before coding, and the objective

is to reduce the defect rate by creating clean code to implement a requirement by the quick feedback gained through testing when this code was refactored.

5. Crystal: This methodology is a set of Agile software development tools utilized for various projects, varying based on size, criticality, complexity, and the number of team members involved (Anwer et al., 2017). According to Abrahamsson et al. (2017), in this methodology, every tool shares common core values and principles, although techniques, roles, and standards may vary. It can choose the most suitable approach depending on the project's size and criticality. Anwer et al. (2017) also explain that the Crystal family includes methods identified by ascending opacity colors, prioritizing collaboration, and communication over rigid processes. It promotes incremental system development, with each iteration limited to a maximum duration of four months.

The Scrum framework was chosen for this project due to its popularity and applicability in the ServiceNow platform application. The following section will be dedicated to its concepts, processes, and best practices.

3.1.3. SCRUM

Scrum is one of the main applications of an Agile framework (Rodríguez et al., 2019). It was explained in 2002 in a book published by Ken Schwaber, one of the seventeen members of the Agile Manifesto (Schwaber & Beedle, 2002).

According to Schmidt (2016), the main feature of this framework is its design, which facilitates project management within the iterative software development process, primarily emphasizing maximizing value in minimal time.

In this context, Rodríguez et al. (2019) say that in Scrum, a shared team objective is established and embraced by all team members, who collectively do their best to achieve it. The team operates self-organized and is empowered to determine the most effective approaches for reaching the set goal. Additionally, Al-saqqa et al. (2020) attribute its popularity to its simplicity and emphasis on addressing software management issues rather than technical software development practices.

Scrum is a straightforward framework composed of three main elements, as explained by Rodríguez et al. (2019) and illustrated in Figure 3.1: the Scrum team, represented by the avatar icons; Scrum events, steps in blue; and Scrum artifacts, steps in green. These elements are divided into three different phases in the Scrum process: the planning phase (pre-sprint), the development phase (sprint), and the closure phase (post-sprint) (Abrahamsson et al., 2017).

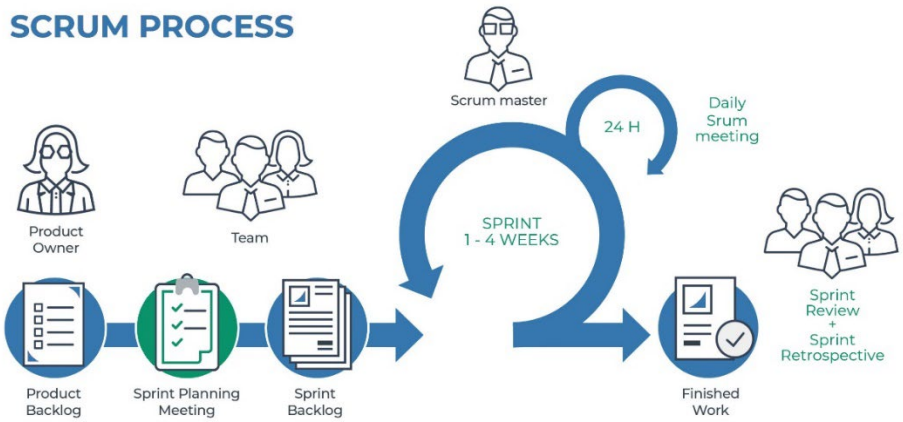


Figure 3.1 – The Scrum process

Source – Adapted from Toro (2022)

The first phase is the planning or pre-sprint phase, in which the general objectives for the system are being developed and established. Moreover, it is when the required tools and resources are stated and the project team is established (Abrahamsson et al., 2017; Al-saqqa et al., 2020).

To begin with, Rodríguez et al. (2019) clarify that the product is created through product backlog items, usually known as user stories and features, which contain all the requirements currently known by the customer, sales, or software development teams (Abrahamsson et al., 2017). Then, the requirements are analyzed, and specific priorities are assigned. After this, the product owner, responsible for maintaining a visible and transparent product backlog, conducts the implementation effort estimation with the Scrum team (Al-saqqa et al., 2020; Rodríguez et al., 2019).

Regarding the creation of user stories, Rodríguez et al. (2019) say that they often follow a pattern to facilitate understanding and standardization, which is, "*As a <User>, I want to <Have>, so that <Benefit>*". Each user story includes an *acceptance criteria*, defining the conditions for implementing the story in Scrum terms. Additionally, user stories provide estimated story points, representing the effort required for implementation.

Al-saqqa et al. (2020) also explain that even after these requirements become stories, the product backlog is continuously updated due to the incremental creation of user stories, and the priorities of existing stories may change throughout the development process due to the flexibility offered by Scrum.

The second phase is the development phase, also called the sprint phase, where the development is conducted in increments called sprints, iterative fixed-length cycles (Al-saqqa et al., 2020), lasting 2 to 4 weeks (Rodríguez et al., 2019). Scrum practices continuously observe and control changing environmental and technical variables (e.g., time frame, quality,

requirements) throughout the sprints, helping adapt to changes instead of only addressing them at the project's start (Abrahamsson et al., 2017).

In every sprint, a specific goal is established during the sprint planning meeting, identifying the user stories to be addressed. This shared objective unites the entire team (Rodríguez et al., 2019). Throughout a Sprint, daily 15-minute Scrum meetings are conducted to facilitate communication, synchronize activities, realign the team's focus on the shared common goal, and address any problems or obstacles encountered during the implementation and testing of features daily (Rodríguez et al., 2019; Schmidt, 2016).

As stated by Al-saqqa et al. (2020), at the end of each sprint, two meetings take place: the sprint review meeting, where the output – referred to as "a potentially releasable increment" – is analyzed and evaluated, and the sprint retrospective meeting, which focuses on potential enhancements for the future.

Last, the third phase, or the project closure phase, is when the requirements are completed, and no more items need to be addressed or taken care of (Al-saqqa et al., 2020). The most recent product version is prepared for release and distribution, involving integration, system testing, and documentation tasks (Abrahamsson et al., 2017).

There are three main roles in the Scrum Team, according to Al-saqqa et al. (2020), which are:

1. The **product owner** defines development team targets by prioritizing customer requirements based on market value and translating them into features in the Product Backlog. They review sprint outcomes and oversee the development team's work to maximize product value.
2. The **Scrum master** is responsible for ensuring that the Scrum values and rules are followed, communicating with external stakeholders, leading Scrum meetings, and protecting the sprint from outside interference. They ensure team optimization and eliminate impediments to Scrum rules during the work development.
3. The **development team** is a self-organized group that collaborates to achieve sprint goals and deliver a potentially releasable product increment at the end of each sprint. Their responsibilities include analyzing requirements, design, development, testing, and validation.

Given that, the main benefits of Scrum in Agile, mentioned by Choudhary & Rakesh (2016), are (i) enhanced team communication, primarily facilitated by daily Scrum meetings that bring developers and testers together; (ii) quick releases, enabling developers to create releases every few weeks; (iii) flexibility of design, short sprints combined with customer feedback resulted in increased efficiency in adapting to changing requirements; (iv) more reasonable process, relaxed processes in an Agile atmosphere.

Scrum, though simple, has three main pillars: transparency, inspection, and adaptation. It keeps process visibility through events like daily Scrum meetings, allowing frequent inspection

to identify and adapt to changing circumstances. Small iterations enhance predictability and risk control, ensuring decisions are based on known factors (Rodríguez et al., 2019).

3.1.4. The Fibonacci sequence and story effort estimation

Research conducted by Sandeep et al. (2022) concluded that one of the most used effort estimation techniques in Agile is story points, usually expressed in numbers that follow the Fibonacci sequence. Story points are a unit of measurement used to estimate the entire effort necessary to perform a piece of software work.

In this sense, the Fibonacci sequence, also known as FS, was introduced by Leonardo Fibonacci, gaining popularity in the 19th century. The sequence is derived from a rule where each number in the series is the sum of the two preceding numbers (Tran-Ngoc et al., 2023). This logic is represented in Figure 3.2.

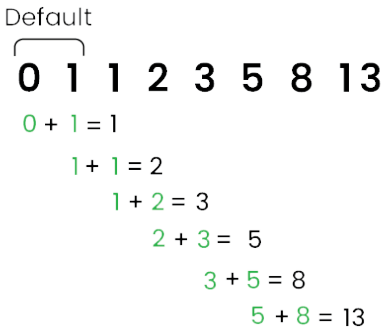


Figure 3.2 – The Fibonacci sequence

Source – (Kashishku, 2023)

Chauhan et al. (2023) explain that the Fibonacci sequence englobes the numbers 1, 2, 3, 5, 8, 13, 21, and so forth. This means larger numbers indicate greater complexity, representing the user stories' relative effort or difficulty. On the other hand, Sandeep et al. (2022) state that Fibonacci numbers are a unit of measurement, including ideal days or man-hours as measurement units.

3.1.5. Capabilities and advantages

In current software development, speed is vital. Rapid development involves quickly creating, releasing, and learning from software in short cycles, sometimes within hours or days. Unlike the traditional approach of periodic releases every few months, software is now viewed as a continuously evolving product with an immediate delivery of new functionality to customers (Rodríguez et al., 2019).

Agile improves understanding of customer needs and requirements through interaction among all team members, as explained by Kodmelwar et al. (2022), fast detection of bugs, delivery of products, and change of project direction according to customer feedback (Saarikallio & Tyrväinen, 2022).

As Schwaber & Beedle (2002) stated, Scrum can be adopted by any organization, as it does not require specific engineering practices. The authors also explain that Scrum can be adopted in an existing or new project, and it is a suitable method for small teams of less than 10 members. If more people are available, multiple teams should be created.

Despite Scrum not providing explicit recommendations for scaling beyond the team level, especially when the team consists of fewer than 10 members (Alqudah & Razali, 2016), Large Scale Scrum (LeSS) is described as a methodology that utilizes the Scrum framework for extensive product development across multiple sites and offshore locations, covering up to ten teams; in other words, it should be used up to 70 people (Larman & Vodde, 2013).

To clarify, Larman & Vodde (2013) say that, in LeSS Sprint Planning meetings, two members from each Scrum team, along with the overall Product Owner, decide on the Product Backlog items to work on, a departure from standard Scrum where the entire team participates. When there is a conflict, the Product Owner mediates. Regular Inter-team coordination meetings can enhance information sharing and coordination.

The main aspects that make a development method Agile, as concluded by Abrahamsson et al. (2017): (i) incremental, with small software releases and rapid cycles; (ii) cooperative, with continuous collaboration between customers and developers with frequent and close communication; (iii) straightforward, where the method is easily learnable and adaptable and (iv) adaptative, being able to make implement changes at the last moment.

In conclusion, Agile software development, which began with the publication of the Agile Manifesto, seeks to initiate a paradigm shift in software engineering. It prioritizes people, interaction, working software, customer collaboration, and adaptability over traditional emphases on processes, tools, contracts, and plans (Abrahamsson et al., 2017). Moreover, Agile software development models are widely known and accepted due to their capacity to meet the demands of fast-paced software development projects and ease of dealing with a changing project environment (Anwer et al., 2017).

3.2. IT SERVICE MANAGEMENT (ITSM)

Information technology (IT) services are crucial for businesses in all sectors, and measuring their performance is essential for managers to determine return on investment. As IT systems are widely used, their quality, reliability, and safety are very important to maintaining business integrity. Large organizations tend to have complex internal structures, which is often impossible for small and medium-sized organizations due to financial constraints (Melendez et al., 2016).

In Information Technology (IT), Service Management (ITSM) is a term used to describe the integration of IT management with industry best practices in the form of process-oriented services. The main focus of ITSM is on service support and delivery to improve IT services tactically and strategically continuously (Mcloughlin et al., 2014; Melendez et al., 2016).

In other words, ITSM is an approach that helps IT managers understand the needs of the IT function by providing guidance and reference models. Reference models, also known as "process reference frameworks", are formal norms, rules, meanings, and work practices that IT managers can implement to enable and constrain the activities of actors within the IT function and its customers. According to Simpson et al. (2016) and Iden et al. (2020), these models provide a top-down approach to managing IT services effectively.

3.2.1. Applicability, benefits and challenges

ITSM focuses on managing IT services, a subset of service science that deals with IT operations. It follows a systematic approach to managing the entire life cycle of IT services, from design to continual improvement, focusing on managing costs associated with the service life cycle (Marrone & Kolbe, 2011; Serrano et al., 2021).

Widianto and Subriadi (2022) have identified various standards and models for implementing ITSM. The most commonly used ones are the IT Infrastructure Library (ITIL) and ISO 20000. According to a survey conducted by Ruiz et al. (2018), ITIL is the most popular framework among them.

A recent study by Serrano et al. (2021) suggests that implementing ITSM frameworks deliberately and with careful planning can result in several advantages for organizations. These frameworks can help the management team create better documentation, enabling detailed audits and IT process reports. Additionally, improved documentation and monitoring can boost transparency and comparability within the organization, increasing process maturity and offering greater control to the management, especially in uncertain situations.

Although ITSM can bring some benefits, many organizations face challenges while implementing it. These challenges have been identified by research conducted by Serrano et al. (2021). The difficulties in ITSM can be classified into four areas: technology, data, process, and people. The most common challenge is resistance to change from co-workers, often attributed to the organizational culture and the difficulties in implementing a new strategy at the organizational level. It is suggested that the management team prepare the technical staff for the changes to overcome this challenge. The authors also mentioned other challenges, such as the complexity of ITSM frameworks and the lack of management support for their implementation.

3.2.2. ISO 20000

The International Organization for Standardization (ISO) 22000 standard is a set of guidelines for managing a service management system (SMS). It outlines the necessary requirements for service providers to plan, establish, implement, operate, monitor, review, maintain, and improve an SMS. These requirements cover service design, transition, delivery, and improvement to meet agreed service requirements (ISO, 2016).

In this context, service providers can also use this standard to demonstrate their capability in designing, transitioning, delivering, and improving services. It helps ensure a consistent approach to service delivery in a supply chain (ISO, 2016).

3.2.3. ITIL

IT services significantly impact competitive advantage, as explained by Serrano et al. (2021). Therefore, having an efficient management system becomes crucial. Many organizations invest in ITSM frameworks, such as the IT Infrastructure Library (ITIL), to improve their IT services.

Many consider ITIL the industry standard for ITSM, as Ruiz et al. (2018) stated. Its terminology is widely recognized and utilized. The approach that ITIL takes is process-driven and based on real-world business experience. In this context, ITIL provides a set of best practices for managing and delivering IT services within an ITSM framework.

The ITIL framework englobes 26 process areas used to describe how IT services evolve through the five main stages of their lifecycle, which are (i) Service Strategy, (ii) Service Design, (iii) Service Transition, (iv) Service Operation, and (v) Continual Service Improvement (Ruiz et al., 2018). These phases are interconnected, as represented in Figure 3.3.

Figure 3.3 – The five phases of ITIL

Source – Adapted from Micreiros (2013)

ITIL goes beyond just outlining phases and processes. It also provides seven guiding principles to assist an organization in all situations. These principles include focusing on value, starting from where you are, making iterative progress with feedback, collaborating and promoting visibility, thinking and working holistically, keeping things simple and practical, and optimizing and automating processes (Klentsova et al., 2020).

Figure 3.4 illustrates the Service Value System (SVS) of ITIL, which shows the critical components of ITIL4. According to Antonio (2023), SVS represents how different activities and components of an organization work together to create value through service. Therefore, they must be integrated flexibly, requiring coordination to keep the organization consistent.

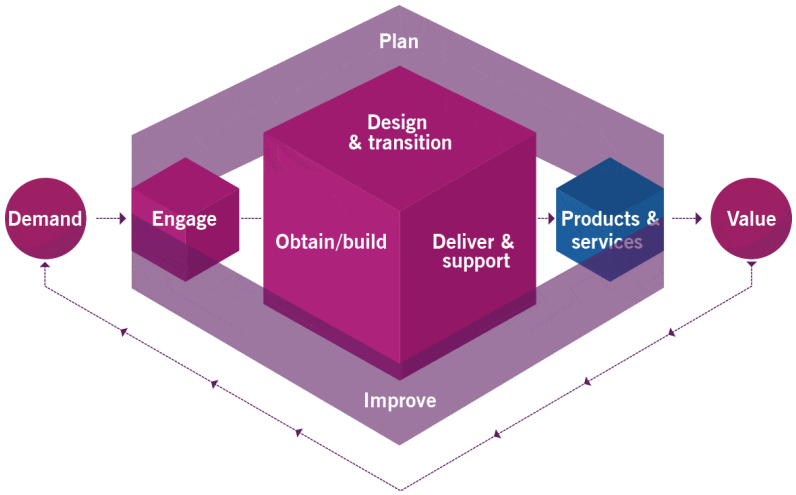


Figure 3.4 – The Service Value System (SVS) of ITIL

Source – Adapted from Antonio (2023)

ITIL has many terms in its practice guides. Adyrbai (2021) explained that ITIL 4 includes 34 management practices, all available online. The most used terms are:

1. **Incident management**: A process that aims to minimize unplanned service unavailability or degradation by detecting and resolving incidents automatically, immediately after they occur. This approach reduces downtime, improves initial data quality for resolving incidents, and may lead to self-healing and invisible incidents, resulting in higher user and customer satisfaction. Service providers can define incident models to optimize the handling and resolution of typical incidents.

2. **Service request management**: Service requests are a regular part of service delivery, having advantages like predictability, which is achieved through detailed planning, testing, and optimization of service request procedures. Service request fulfillment procedures for all services must be optimized to ensure success. It handles all user-initiated service requests in an effective and friendly way while supporting the agreed quality of service. Service requests are also included in the organization's service catalog and provide information about available services, products, required information, approval workflows, and other relevant details.

3. **Service desk**: The service desk manages various communication channels to deliver the right message through the most appropriate channel. Omnichannel communications are used to provide a unified communication experience. The service desk is critical for user and customer satisfaction and the success of service relationships. It collects information about user satisfaction through surveys. Effective and convenient communication channels influence user satisfaction. Communication channels should be trusted, effective, and convenient for users.

Another useful term to mention besides the three most commonly used ones is change management or change enablement. As per Hasibović et al. (2023), in ITIL4, the term "change

management" has been replaced with "change enablement." This new practice maximizes the success of service and product changes by properly assessing risks, authorizing changes to proceed, and managing the change schedule.

According to the authors, change enablement has the benefit of guiding processes to satisfy stakeholders. It is a customer-focused process that helps organizations continuously improve their infrastructure and processes. This ensures businesses can easily implement necessary changes without disrupting other business operations.

To conclude, Giuffrida & Dittrich (2015) say collaboration is crucial in software development, with tasks delegated to different teams. Efficient task management and prioritization based on specific criteria are essential. Although some tasks can run simultaneously, dependencies exist, necessitating coordination between tasks, processes, and teams. This coordination is fundamental to achieving optimal software or product outcomes with minimal costs.

3.3. DASHBOARDS AND DATA VISUALIZATION

In today's competitive market, businesses need constant access to analytical reports on their activities. Modern data analytics and visualization techniques, such as dashboards, can make this available by adding value to decision-making based on the data (Orlovskyi & Kopp, 2020).

As Few (2013) explains, a dashboard is a visual display that presents the most critical information necessary to achieve one or more objectives, consolidated and arranged on a single screen. This allows for quick monitoring of information at a glance. The author also emphasizes that the visualization charts used on a dashboard should be selected to suit best the nature of the data sets displayed. Additionally, the visualization charts should still serve their intended purpose even if resized to fit into small spaces on the dashboard.

Pappas & Whitman (2011) point out that digital dashboards are often used as executive dashboards intended to communicate the performance of an organization related to its corporate goals to management. Such dashboards usually present comparative data, comparing current performance with past performance or target levels. The authors also recommend designing an effective dashboard by choosing data visualizations that convey the message, are easy to interpret, use space efficiently, are visually appealing, guide the user, and enable them to complete tasks.

Having said that, the authors suggest using line graphs and bar charts for quick comparisons, as they can provide some interactivity, such as filtering. Less efficient visuals include dials and speedometers, while pie charts can draw attention to essential measures.

Regarding dashboard organization, Few (2013) explains that simply putting information on a dashboard is not enough. The way the information is arranged on the dashboard can be the difference between a useful and an ignored dashboard, even though they both present the same information. Moreover, when arranging data on a dashboard, it is essential to consider

the following: group data according to business functions, entities, and use; place items that belong to the same group together and support meaningful comparisons.

According to Eckerson (2012), one of the primary challenges in creating a helpful dashboard is designing it informative and appealing. The dashboard should display all the performance elements necessary to address business issues. The author provides some guidelines to achieve this, such as focusing on the requirements, available data, and understanding the users' needs.

The initial screen should be designed to catch the user's attention immediately. Additionally, the author suggests that "less is more" regarding visuals. It means that displaying only the necessary information on a single screen can be the key to success and that users should not have to scroll to view critical data.

Eckerson (2012) explains that a helpful technique in organizing dashboards is using white spaces to group objects, following the Gestalt principle of proximity. These principles were derived from psychological research done in Germany in the early 20th century. The proximity principle of Gestalt employs white spaces to enable viewers to perceive data as arranged in vertical or horizontal sets, as demonstrated in Figure 3.5.

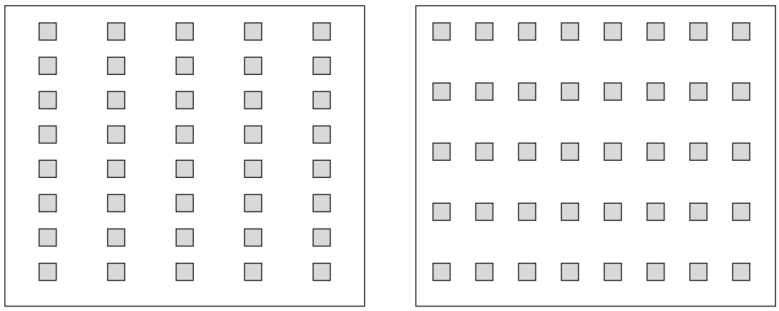


Figure 3.5 – Gestalt principle of proximity

Source – Eckerson (2012)

In summary, dashboard design should focus on conveying the meaning of the data with minimal design elements, and good design should highlight key trends and relationships (Eckerson, 2012). Few (2013) complements the idea, stating that to create successful dashboards, it is necessary to focus on communication, ensuring that users can quickly and easily understand the information presented through a simple and clear design.

4. PROJECT

This project consists of five main phases, described in the following sections. It is essential to highlight that the decisions presented here were based on the content of the scientific community obtained through the theoretical framework in the previous chapter.

4.1. PHASE 1: PROJECT KICK-OFF AND SET THE STAGE

This first phase begins the development of the project by gathering requirements, preparing the platform, and understanding the best customization and development practices provided by ServiceNow.

4.1.1. Getting started with the Agile Development application

The Agile methodology, as seen in the theoretical framework chapter, was rapidly adopted by the IT world due to its adaptive and collaborative approach to project management in an incremental way. This methodology increases team productivity by taking advantage of these features, improving customer satisfaction.

ServiceNow's Agile Development 2.0 application offers tools and features that support Agile methodologies like Scrum. Teams can create and manage user stories, tasks, and sprints while visualizing progress through boards and dashboards. The application allows for easy prioritization of work items, sprint planning, and tracking of team velocity. It promotes transparency by providing real-time insights into project status and enables seamless communication among team members.

Like other ServiceNow applications, Agile Development is organized into modules. Integration with other ServiceNow modules allows for a holistic approach, connecting Agile development with broader IT service management processes for a more streamlined and efficient workflow.

To start the project, a ServiceNow Personal Development Instance (PDI) using the latest Vancouver release was requested. This instance will develop, test, and implement the project. Once the instance has been created, the applications and plugins needed for Agile Development must be installed. This is done by accessing ServiceNow's official store of applications and plugins, the ServiceNow Store. This store is a marketplace that holds both free and paid applications built on the ServiceNow platform.

In this context, the Agile Development product contains three licensed plugins:

a) Agile Development 2.0: Provides enhanced functionality on top of Agile Development.
b) Agile Development – Unified Backlog: This backlog includes problems, incidents, enhancement requests, and other ServiceNow tasks in the Agile workflow.
c) Agile Development 2.0 – ATF Test: Provides test cases and test suites for the Agile Development 2.0 application.

The installation process is simple, and all plugins are installed together. After installation, the Agile application and modules will appear right away in the instance. In Figure 4.1, the left menu shows the Agile application with its modules; the rest shows the ServiceNow instance's home page.

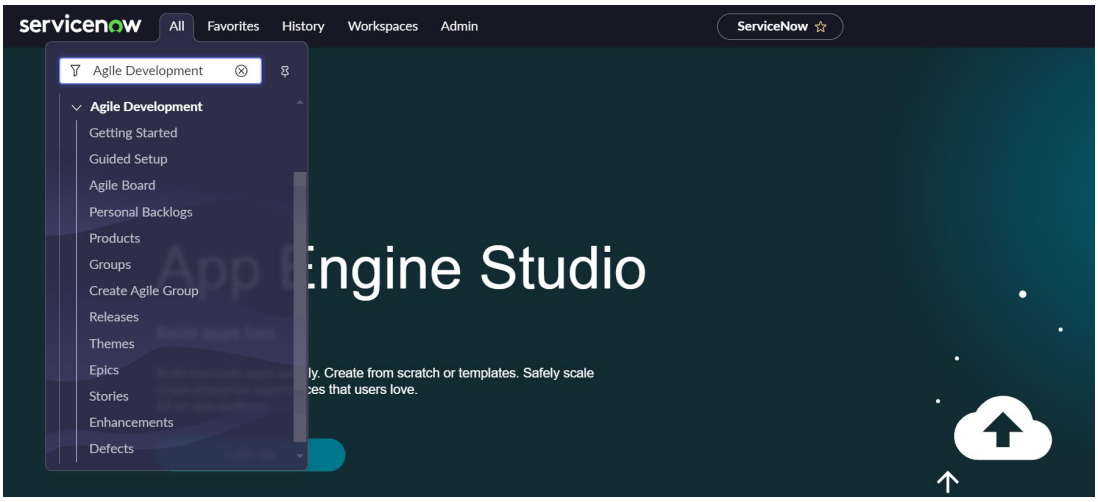


Figure 4.1 – ServiceNow Agile Application and Home Page

Source – ServiceNow PDI (2024)

4.1.2. System update sets to save work

According to ServiceNow (2023c), an update set is a pack of configuration changes grouped in an XML file that can be moved between ServiceNow instances. Administrators can group changes under names and transfer them for testing or deployment. Update sets were used to save the customizations made in this project and separate the developments by topics and modules.

In this case, and for organizational reasons, the naming pattern was constructed as follows: RS - Agile - Form v1.x, where RS are the initials of the author's name, Agile refers to the module or application customized, Form is the place where the change was made, and *v1.x* refers to the development version.

4.1.3. Installing Add to Update Set utility

As mentioned in the previous subsection, update sets save customizations in small XML files to transport them between ServiceNow instances. The Add to Update Set utility is handy when developing and saving customizations. Once installed, this utility becomes present on all forms, allowing the entire record to be added to the current update set the developer is working on.

This allows the developer to ensure that their customizations are preserved while also allowing them to add only what is necessary to the current update set. This utility is available for download from ServiceNow Share, a community where ServiceNow developers share code

and features created for the platform. For this project, the latest version, 7.7, published by Hollifield (2024), was used. The installation of this utility was successful, tested, and working.

4.2. PHASE 2: THEORETICAL FRAMEWORK FOR DECISION-MAKING

This is a continuous development phase that covers the entire project lifecycle. The focus is on adopting Agile methodologies and integrating IT Service Management (ITSM) and IT Infrastructure Library (ITIL) best practices to guide decision-making throughout the project.

Chapter 3, Theoretical Framework, presents and analyses the academic literature collected during this phase.

4.3. PHASE 3: IMPLEMENTATION

This is the main phase of the project, as it gathers all the developments and customizations created on the ServiceNow platform. Automation, notifications, knowledge articles, reports, and dashboards follow it.

4.3.1. Tables and data validation

ServiceNow operates through a structured database, that is, through tables and the relationship between them. To begin with, it is vital to understand the tables on the platform and those that come with the Agile Development application when it is installed. In this context, "out of the box" is a term commonly used by ServiceNow developers to refer to a feature or functionality of a product that works as is without requiring any custom modifications. In other words, it is a feature already available within the instance and can be used in the same way it was initially activated.

It is important to remember that the vast majority of customizations carried out within the scope of this project will be made to the tables and their respective forms, which are the visualization of the table's records in a customized way.

Having said that, the focus of the customizations will be on the main tables in the Agile Development 2.0 application, established according to the ServiceNow documentation (Boothpoor, 2023). They include:

a) **Epic** (table: *rm_epic*) represents related stories or requirements that have not yet been transformed into individual stories.
b) **Sprint** (table: *rm_sprint*) stores sprints, the backlog items to be addressed together during a given period.
c) **Stories** (table: *rm_story*) represents self-contained units of work that can be completed within a sprint.

In addition, after analyzing the customization needs, it was identified that there was no need to create a new table or change the relationship between them.

4.3.2. Columns and fields customization

The *Stories* table was analyzed to begin with. As the main customizations will be made to this table, several columns must be created, and this subsection showcases all of them.

As previously stated, the ServiceNow platform is a database that operates on a relational model. In this context, the Stories table is an extension of the *Task* table, which is considered one of the core tables of the system. The *Task* table provides a set of standard fields utilized by all tables that extend it. The ServiceNow documentation also highlights that any table that extends *Task* can take advantage of the task-specific functionalities (Romero, 2024).

After analyzing the *Task* table and its fields, it was identified that there is a field called *Time worked*. As per ServiceNow's documentation, this field is used for time-tracking purposes. Any table that extends Task can also utilize this field (ServiceNow, 2024g). As shown in Figure 4.2, this field operates as a simple counter or timer. As the record is viewed, the timer counts upward. Clicking the stop icon pauses the timer, and clicking the start icon resumes it.

Figure 4.2 – Time worked field

Source – Adapted from (ServiceNow, 2024g)

Having said that, the *Time worked* was the first field added to the *Stories* table. This will be one of the main fields, as it measures the time elapsed in each state of the story and will allow the partial times related to these states to be saved in the intermediate fields, explained in more detail below. After this, some audits were performed to see the automatism related to the functioning of this variable. An *onLoad* script makes this variable start running, meaning it will start counting the time whenever a form is loaded.

Following this approach to measuring partial and total working times, five new fields were created. These new fields have been designed to record and increase the total development time for each state, as detailed in Table 4.1.

Table 4.1 – New fields of the Stories table

Field name	Type	Expected feature
Total time worked	Duration	The sum of the times of all partial fields
Time worked (Testing)	Duration	The sum of time spent on Testing
Time worked (Work in progress)	Duration	The sum of time spent on Work in progress
Time worked (Validation)	Duration	The sum of time spent on Validation
Time worked (Draft)	Duration	The sum of time spent on Draft

Source – Prepared by the author

Three more fields were created to improve the dashboard filtering. They will store necessary information about developments that have not passed the tests and have been sent back to development.

The first field, *Sent back to development*, is true/false and stores information about whether the story went back. The second, *Number of times sent back*, is an integer field that will store and increment the number of times the story has been sent back. The third, *Sent back reason*, is a choice field, with options in a dropdown for the tester to choose the reason for sending this development back. The first and second fields will be filled in and automatically incremented with scripts created in subsection 4.3.4.

Lastly, two more fields have been created to deal with document attachments. These fields were created because, according to the Agile methodology practices collected in the theoretical framework, documentation is very important for keeping track of developments. One of the fields is called *Development documentation*, for the developer to upload their development document, and the other is *Testing documentation*, for the tester to upload their testing document.

It is important to mention that additional customizations, as described in subsection 4.3.4, will be made through scripting, such as making fields mandatory, read-only, or autofill, always observing the methodology guidelines collected in the theoretical framework.

4.3.3. Form layout – the Stories table

The *Stories* table form had to be customized to provide a better experience for the end user. This customization considered the layout and organization of the fields for better visualization and created individual tabs to show the relevant fields.

Figure 4.3 illustrates the final version of the *Stories* table form. The respective automation has been deactivated to view the form as a whole with all the new fields.

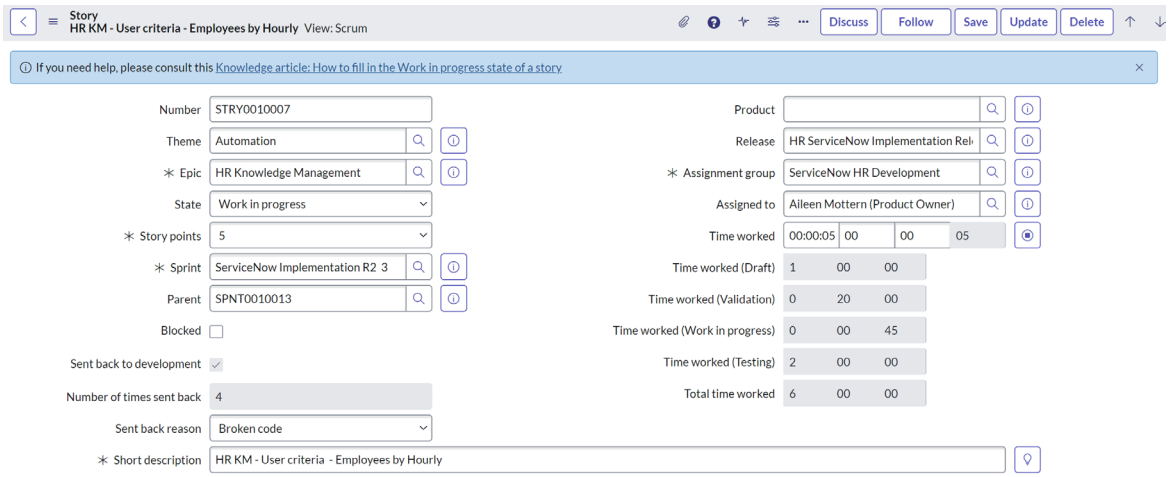


Figure 4.3 – Stories table form

Source – ServiceNow PDI (2024)

4.3.4. Automation and scripting

Configuring automation tools like Business Rules, Client Scripts, and UI policies for a table in ServiceNow is an important step for meeting customization requirements efficiently and effectively. These automation features simplify processes, enhance user experience, and ensure data integrity within the platform. They were developed by following the best practices collected in the theoretical framework.

4.3.4.1. Business Rules

In ServiceNow, Business Rules enable automated actions based on predefined conditions, reducing manual intervention and minimizing errors. According to the ServiceNow documentation, they are server-side logic that automatically executes when a database is queried, updated, inserted, or deleted. They respond to database interactions regardless of the access method (ServiceNow, 2024a).

In this context, four Business Rules were created to automate the new development behaviors, mainly by providing the functionality of incrementing and calculating the duration of the field *Time worked*, passing its values to the partial field, and creating a final sum in the *Total time worked* field.

The first and most complex is called *RS - Increment time worked totals Change*. As the name suggests, this Business Rule aimed to create code to automate increasing hours in the partial and total fields. It was created with the trigger condition "before - update" and the filter condition "state - changes", meaning it will always run before a form is updated and when the state of the story changes.

A Business Rule can consist of low-code conditions added to an advanced part written in JavaScript code. Figure 4.4 shows the trigger configuration for this Business Rule, created in low-code format.

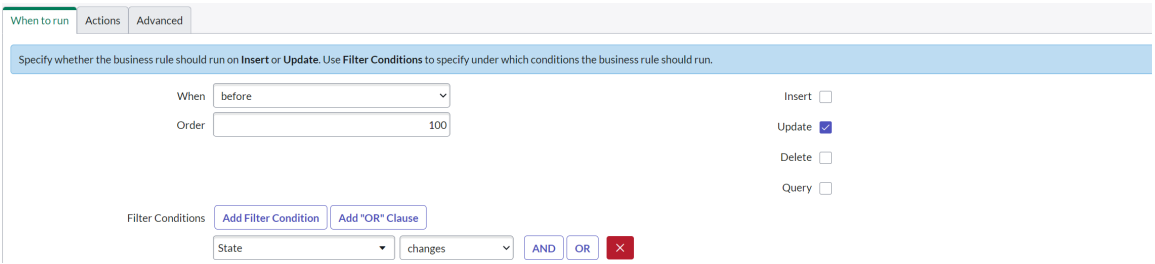


Figure 4.4 – Business Rule when to run: state changes

Source – ServiceNow PDI (2024)

The code for the advanced section was divided into two parts. To write the code correctly, the Application Programming Interface (API) provided by ServiceNow, called *GlideDuration*, was used.

This API helps instantiate objects and return their value in a time duration format. The *GlideDuration* function obtains the display value of the specified duration of the field and *getDurationValue()*, which returns the duration value in "d HH:mm:ss" format, where d = days, h = hours, m = minutes, and s = seconds (ServiceNow, 2024d).

The first part of the code was creating variables that will be used for the if conditions. These variables were created considering the API explained in the previous paragraph. First, it was necessary to create a variable that gets the duration of the *Time worked* field, which is the primary field for the calculations as it is the timer.

After this, the *GlideDuration* function of the previously created variable was used to convert the field value into an object. This conversion was necessary because the main purpose of this code is to increment the durations of the fields whenever the state of the story changes.

The following variables were created using the same logic: the first variable to get the duration of the partial field, the second to convert this duration into an object, and the third to add the current duration (*Time worked* field) to the existing duration of the partial field (*Time worked Draft, Validation, Work in progress,* or *Testing*). Figure 4.5 demonstrates a partial view of the first part of the code, showing the variables created and the comments.

Script

```
1  (function executeRule(current, previous /*null when async*/ ) {
2
3      // Gets the format of the Time Worked field, e.g. "5 12:00:00"
4      var timeWorked = current.time_worked.getDurationValue();
5      // Performs a GlideDuration of the previous variable, returning the format "5 12:00:00"
6      // Stores the value of Time Worked variable as an object, being possible to make sums and
       increment the total value
7      var gldTimeWorked = new GlideDuration(timeWorked);
8
9      // The same logic as below for Draft
10     var timeDraft = current.u_time_worked_draft.getDurationValue();
11     var gldTimeDraft = new GlideDuration(timeDraft);
12     var draftTotal = gldTimeDraft.add(gldTimeWorked);
13
14     // The same logic as below for Validation
15     var timeValidation = current.u_time_worked_validation.getDurationValue();
16     var gldTimeValidation = new GlideDuration(timeValidation);
17     var validationTotal = gldTimeValidation.add(gldTimeWorked);
18
19     // The same logic as below for Work in progress
20     var timeProgress = current.u_time_worked_work_in_progress.getDurationValue();
21     var gldTimeProgress = new GlideDuration(timeProgress);
22     var progressTotal = gldTimeProgress.add(gldTimeWorked);
```

Figure 4.5 – Business Rule “Increment time worked totals” partial script

Source – ServiceNow PDI (2024)

The second part of the code is built around the if conditions. It was composed according to the needs of the changing state during a story's development. The previous and current properties of the function were used to create state change scenarios, such as from Draft to Validation.

Following this reasoning, when the state changes from Draft to Validation, the value of the *Time worked* field is added to the immediately previous state, which is Draft. This means the time worked that elapsed while the story was in the Draft state is stored in the *Time worked (Draft)* field. Part of this JavaScript code can be seen in Figure 4.6.

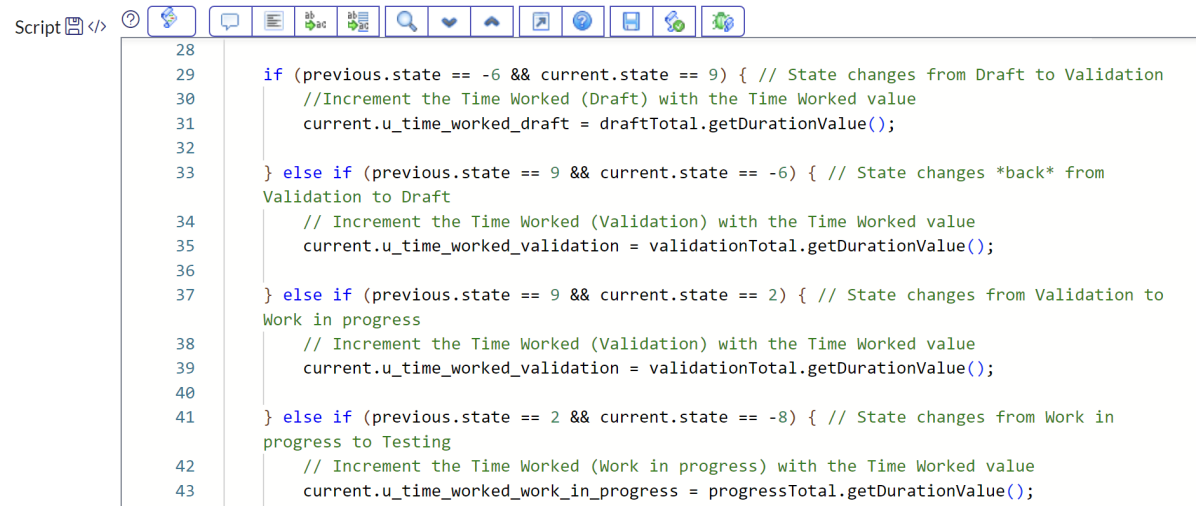


Figure 4.6 – Business Rule “Increment time worked totals” if conditions

Source – ServiceNow PDI (2024)

A similar Business Rule, *RS - Increment time worked totals Save*, has been created to complement the automation related to times and durations. It was also created with the condition “before - update” but without a state filter. This means it will be triggered before a form is updated and the state has not changed.

This Business Rule allows users to save their work progress and continue later without altering the current story state. Upon saving the form, the *Time worked* field is incremented in the partial field of the current state and subsequently reset.

In other words, if a developer is working on a story in the Work in Progress state and decides to stop to continue at another time, they must click on the stop button in the *Time worked* counter and save the form. This will trigger the Business Rule, and the value in the *Time worked* counter will be added to the *Time worked (Work in progress)* partial field, thus increasing the working hours. When the user decides to continue their development, the Time worked counter will start running as soon as they open the Story form, which is an out-of-the-box behavior. However, it will count from zero since this Business Rule also reset its value after running.

The advanced section of this Business Rule uses the same script logic as the previous one. Minor changes have been made to the if conditions that increase the value of the *Time worked* field to be incremented in the current state of the story instead of going to the previous one since, in this rule, there is no change of state.

Nevertheless, for this automation to work as expected, the user must always respect and be transparent about the hours worked. This means that the user is responsible for stopping or continuing the *Time worked* counter according to their actual work routine in the current story. This ensures that the data collected is reliable and that the dashboard created from this data can be used for correct decision-making.

The third Business Rule created is *RS - Story back from Test to Progress* and automates processes when a story changes from Testing to Work in progress. For this, the trigger condition was set to “before - update”, and the filter conditions are "state *changes from* Testing" and "state *changes to* Work in progress". Figure 4.7 shows the low code part of this Business Rule.

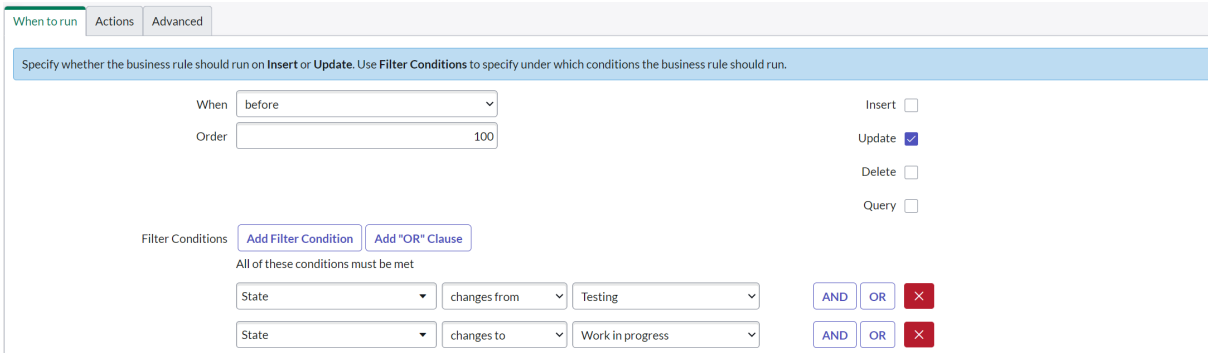


Figure 4.7 – Business Rule when to run: state changes from and to

Source – ServiceNow PDI (2024)

For the advanced section, the first part of the code consists of a *GlideRecord* to the stories table. The *GlideRecord* class, according to ServiceNow (2024c), is the way to interact with the ServiceNow database from a script. *GlideRecord* interactions begin with a query to the database, where the strategy is to create a *GlideRecord* object for the table of interest, then create the query conditions, execute the query, and apply script logic to the records returned in the *GlideRecord* object.

The code shown in Figure 4.8 stores the *GlideRecord* in an *assignTo* variable, which contains the name of the Assignment Group of the current story. With this variable, it was possible to construct the sentence: "This story was sent back to Work in progress (development phase). A notification will be sent to the development team " + *assignedTo* + " to which this story has been assigned to.". Here, the *assignTo* will be automatically populated with the display value of the Assignment Group, thus completing the sentence.

The next part of the code automatically sets the *Sent back to development* field to true to fulfill the requirement that is being selected whenever a story goes backward. In addition, the code increments the *Number of times sent back* field that will be used for filtering purposes. The advanced code can be found in Figure 4.8. This Business Rule was created to populate the fields that will later be used for filtering purposes in the dashboard's Quality tab.

Figure 4.8 – Business Rule “Story back from Test to Progress” partial script

Source – ServiceNow PDI (2024)

Finally, a fourth Business Rule was developed to trigger informative messages at the top of the form automatically. These messages inform users about available knowledge articles, described in subsection 4.3.5, that can help them fill in the form fields, use the *Time Worked* counter, understand the new fields, and get other valuable tips.

The Business Rule is called *RS - Show Knowledge Article on Load.* Its trigger condition is “display”, the filter conditions are "active *is* true" and "state *is not one of* Closed and Cancelled". This triggers this Business Rule whenever the form of an active story is opened and when this story is not in the Closed or Cancelled state. The JavaScript code of this Business Rule starts by using an out-of-the-box property called *instance_name*. This property takes the first part of the link to the current instance to build the access link to the knowledge article.

After this, four variables are declared for each of the four knowledge articles created to store their access link. These articles were created to match the need for an article to appear in its respective state. A part of the code can be seen in Figure 4.9, showing the links built from the property and added to the rest of the info message.

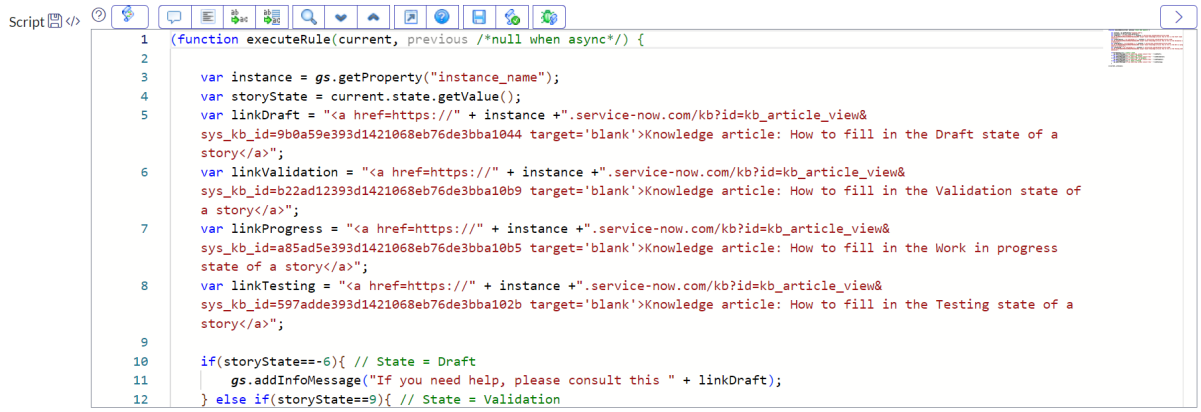


Figure 4.9 – Business Rule “Show Knowledge Article on Load” partial script

Source – ServiceNow PDI (2024)

Then, the following sentence composes the info message: "If you need help, please consult this:" followed by the link to the knowledge article corresponding to the current state.

All these automation and rules were developed considering the Agile methodology practices described in the theoretical framework, aligned with ServiceNow technical documentation. The main objective was to make the form easier to fill in and ensure the fields have the necessary information, maintaining data conformity.

4.3.4.2. Client Scripts

Client Scripts are a feature of ServiceNow that allows JavaScript to be executed on the client (web browser) when client-based events occur, such as when a form loads, after form submission, or when a field changes value. According to the ServiceNow documentation, Client Scripts configure form fields and field values while using the form (ServiceNow, 2024b).

In simpler terms, Client Scripts enhance the user interface by providing dynamic behavior and validations, improving usability and accuracy. In this sense, three Client Scripts were created to automate some behaviors that happen on the client side, such as limiting choices of a field and even stopping the *Time worked* counter when the state is Closed.

The first of the three, *RS - Limit stories state choices*, was created to limit the choices in the state field depending on the current state of the story. That said, this Client Script limits the state options that appear depending on its state, as shown in Figure 4.10.

To do this, the script was created with the *onLoad* trigger, meaning it will happen whenever a form is loaded. Instead of specifying the states that should appear in the dropdown, the states that should not appear have been removed to make the code as dynamic as possible. In addition, there is an if condition to make the *State* field read-only when the state is Complete. This means a story will not be reopened by mistake once it is finished.

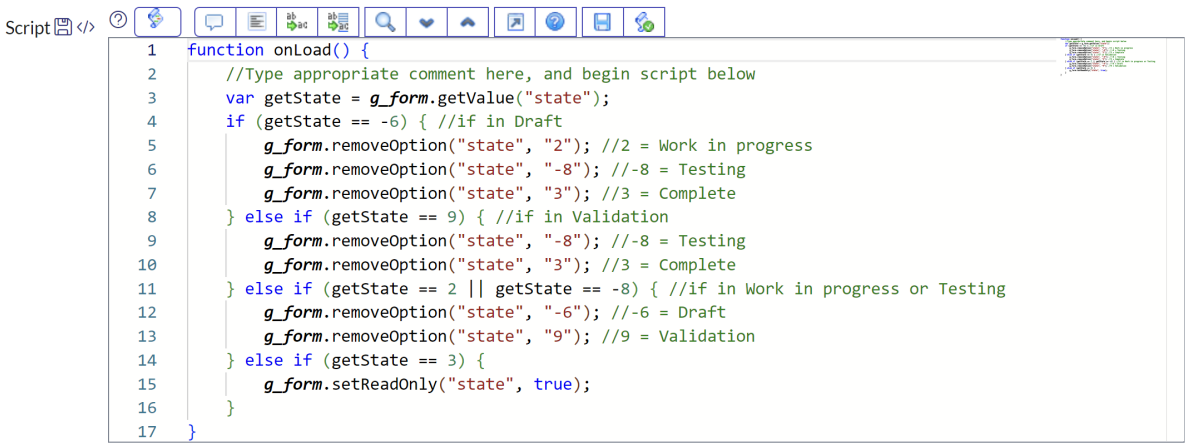


Figure 4.10 – Client Script “Limit stories state choices”

Source – ServiceNow PDI (2024)

Another practical example is when a story is in the Testing state. The dropdown will only show the states: (i) Work in progress, which means sent back to development; (ii) Complete if the developments are working properly; and (iii) Cancelled to cancel the story for a given reason. This reduces the chance of the story being passed to the wrong state or even skipping a state.

The second Client Script, *RS - Make "Sent back reason" mandatory,* was created to make fields mandatory based on state changes. It has been developed because these are client-side alterations, meaning state changes that occur directly in the client's browser and cannot be scripted via Business Rule.

This script was set as an *onChange* type of the field *State*, and the if conditions developed are regarding making the fields *Sent back reason*, *Development documentation* and *Testing documentation* mandatory. If the current state is Testing and the new state is Work in progress, the field *Sent back reason* becomes mandatory for the tester to choose why this story was sent back to development. If the previous state was Work in progress, the *Development documentation* field becomes mandatory for the developer to upload the documentation. The same behavior occurs when the previous state is Testing, where *Testing documentation* becomes mandatory.

Next, the third Client Script, RS - *Stop "Time Worked" when Complete*, was created to stop the *Time worked* counter when the story state is Complete. This code was built from an adaptation of the code created by developer Bernd, made available on his development blog. He explains that the script stops the counter and disables the counter's start/stop button until the state is changed and the task is saved (Bernd, 2021). The idea is to stop the counter and set it to “00:00:00”. The script function adapted from Bernd’s code can be seen in Figure 4.11.

Script

```
function toggleTimer(fieldName) {
    try {
        // Get the timer elements for fieldName given
        var timerEl = $('element.' + g_form.tableName + '.' + fieldName);
        var timerIcon = $('link.' + g_form.tableName + '.' + fieldName);
        if (timerEl != null) {
            // Make sure script does not break if field not present
            var timerPause = timerEl.select('input[id*=paused]')[0];

            // Toggle the timer
            if (timerPause.value == 'false') {
                timerPause.value = 'true';
                timerIcon = $('link.' + g_form.tableName + '.' + fieldName);
                timerIcon.removeClassName('icon-stop');
                timerIcon.addClassName('icon-play');
            } else {
                timerPause.value = 'false';
                timerIcon.removeClassName('icon-play');
                timerIcon.addClassName('icon-stop');
            }
        }
    } catch (e) {}
}
```

Figure 4.11 – Client Script “Stop Time Worked when Complete”

Source – Adapted from Bernd (2021)

In summary, creating appropriate Client Scripts that align with business requirements ensures smooth operations, user satisfaction, and efficient resource use. These scripts streamline workflows, prevent errors, and improve automation.

4.3.4.3. UI Policies

UI policies, as described in the ServiceNow documentation, allow users to control the behavior of information on a form and custom process flows for specific tasks. In simpler terms, they can make certain fields read-only, mandatory, or hidden. UI policies are designed to be user-friendly and do not require any scripting. For more advanced actions, the Run Scripts option can be used (ServiceNow, 2024h).

Despite sharing similar functionalities, UI policies are distinct from Client Scripts. Client Scripts are also used to govern form and form field behavior, but UI policies are more efficient as they have faster load times. In other words, they simplify data entry and enforce business logic, making it easier for users to navigate and use the platform.

A UI policy called *"RS - Make Work notes mandatory when Sent back reason is Other"* was developed because creating just one condition in a low-code way is easier, taking advantage of the out-of-the-box settings. When the *Other* option is chosen, the *Work notes* field becomes mandatory, asking the user to comment on why this development has issues. Figure 4.12 represents the low-code interface of this UI Policy.

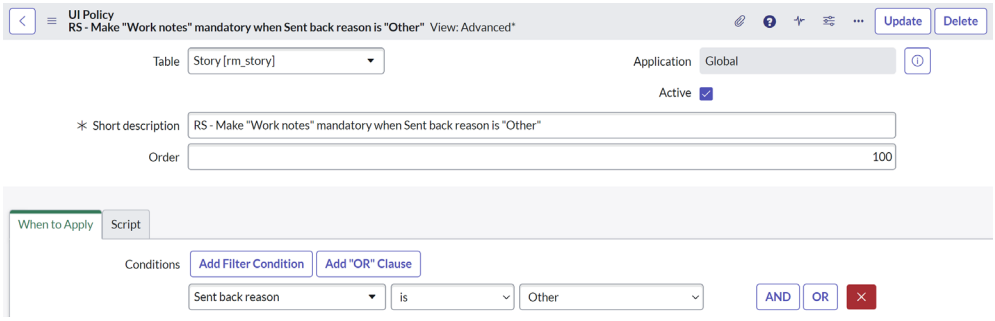


Figure 4.12 – UI Policy low-code interface

Source – ServiceNow PDI (2024)

However, because this UI Policy aims to make a field mandatory and add an info message below the field *Sent back reason*, the advanced Run Script condition was activated, as shown in Figure 4.13.

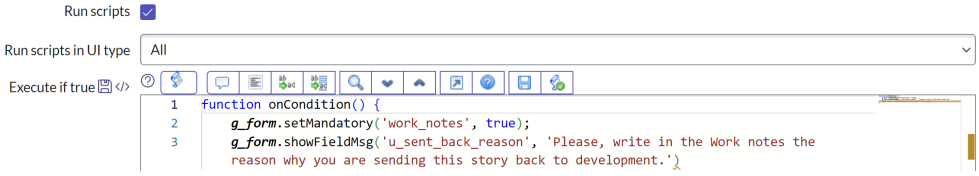


Figure 4.13 – UI Policy advanced scripted interface

Source – ServiceNow PDI (2024)

In conclusion, the new fields related to *Time worked* were given the functions described in Table 4.2, implemented through the automation documented in this subsection.

Table 4.2 – Time worked fields automation description

Field name	Description	Automation
Time worked	Automatic counter with start/stop button	It starts counting whenever the form is loaded.
Time worked (Draft)	The sum of time spent on Draft state	It increments with the time in the Time worked field whenever the state changes to Validation or the form is saved in the current state.
Time worked (Validation)	The sum of time spent on Validation state	It increments with the time in the Time worked field whenever the state changes to Work in progress or the form is saved in the current state.
Time worked (Work in progress)	The sum of time spent on Work in progress state	It increments with the time in the Time worked field whenever the state changes to Testing or the form is saved in the current state.
Time worked (Testing)	The sum of time spent on Testing state	It increments with the time in the Time worked field whenever the state changes to Complete or the form is saved in the current state.
Total time worked	The sum of the times of all the partial fields	It is incremented whenever a partial field is also incremented, thus becoming a sum of all the partial fields to form the total time worked.

Source – Prepared by the author

This concludes this subsection of automation, created using the good practices gathered in the theoretical framework and the guidelines established by the ServiceNow platform documentation, combined with the author's knowledge and technical expertise.

4.3.5. Knowledge base creation

This subsection explains the knowledge articles created on the platform within the knowledge management module. In this context, Knowledge Management (KM) is a work management method that recognizes the significance of intangible assets, such as knowledge. To achieve this, knowledge organization involves designing an infrastructure that caters to specific needs and objectives. In business terms, an organization can be regarded as a large entity that needs to solve three business problems to function effectively: coordination, memory, and learning (Milton & Lambe, 2020).

The primary focus of this KM development is learning. The authors explain that an organization must be capable of responding appropriately to changes in its external environment, adapting its practices accordingly, and internalizing what it learns. The business issues in this aspect include accelerating the learning curve, achieving continuous improvement, accessing business intelligence, and providing decision support (Milton & Lambe, 2020).

The ServiceNow documentation explains in technical terms that the knowledge management application enables the sharing of information through knowledge bases. These knowledge bases consist of articles providing users with different information, including self-help, troubleshooting, and task resolution (ServiceNow, 2024f).

To achieve this, a knowledge base called Agile Development was created to store the customizations and new functionalities added to the Agile Development 2.0 application. It has been integrated into the main knowledge page, enhancing its usability and accessibility.

In addition, six knowledge articles were created to inform users and act as a guideline for filling in the new version of the forms. The script integrated these articles into support messages that users will receive in every state of a story, from Draft to Testing. An example of this integration can be seen in Figure 4.14, where the info message appears at the top of the form.

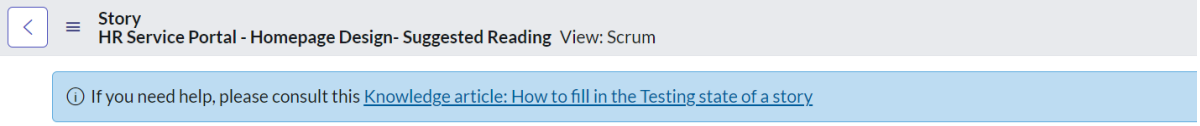


Figure 4.14 – Knowledge article info message in the Story form

Source – ServiceNow PDI (2024)

The six articles were classified into two categories: process documents and user guides and are available to all platform users. This classification makes it easier to filter articles by type of content. It is important to mention that the knowledge articles created are not saved by default in the update set but have been manually added using the “Add to update set” utility.

4.3.6. System notifications

Another module available is Notifications. According to the platform's documentation, this module allows administrators to send email notifications to selected users regarding specific system activities. Among its features, administrators can set when the notifications should be sent, who should receive them, and their content (ServiceNow, 2024c).

In this sense, three notifications have been created and set to be triggered via e-mail. The group assigned to the story will receive relevant updates regarding its process. One of them also has an email script with an if condition that will direct the type of text and the value of the variables used to construct this text.

The first notification was developed to inform the assignment group that a story assigned to it had been sent back for development. It is called *RS - Story sent back to development*; the send condition is when the *Sent back to development* field changes to true. It is important to stress that the condition has been set to “*changes to*” to prevent duplicate notifications from being sent.

Next, the text was constructed dynamically by getting the values of the variables, the name of the sprint, and the link to access both the story and knowledge. The out-of-the-box template was used, and the final notification can be seen in Figure 4.15, where the expressions highlighted in yellow are the dynamic variables that compose the text of the email.

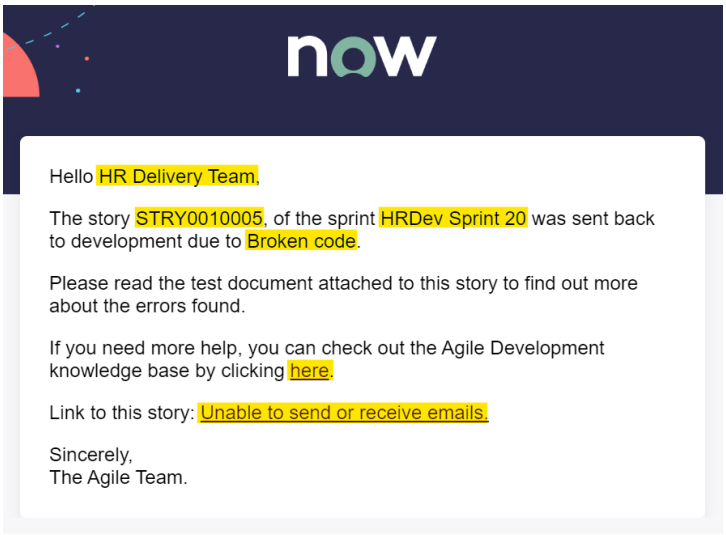


Figure 4.15 – Notification “Sent back to development”

Source – ServiceNow PDI (2024)

A second notification, *RS - Story awaiting validation*, has been created to inform the product owner when a story is pending validation. It triggers when the state changes from Draft to Validation. The final result can be seen in Figure 4.16.

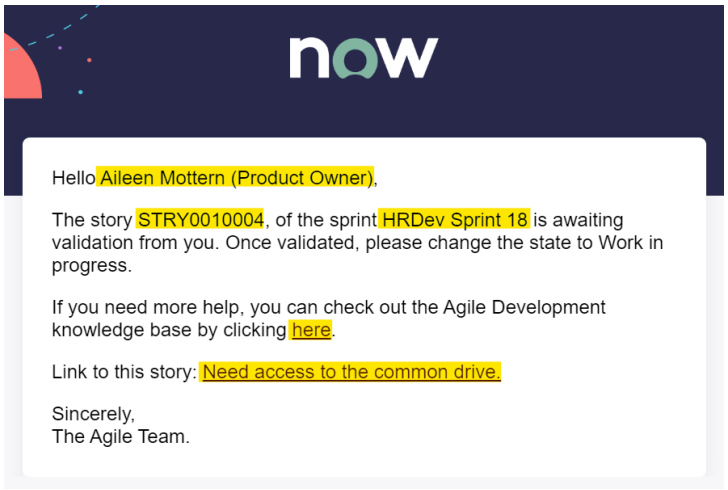


Figure 4.16 – Notification “Awaiting validation”

Source – ServiceNow PDI (2024)

Finally, a third notification was developed to inform the assignment group of the story's completion. It is called *RS - Story is now complete* and the trigger is when the state changes to Complete. The text of this notification includes the most relevant aspects of the story's

closure, including the story's number and sprint, the total number of hours it was worked on, and information on whether this story went backward, how many times, and why.

This notification also includes a mail script, making it possible to create some of the sentences dynamically. If the story was sent back to development, it will show the following sentence: *"This story was sent back to development 5 time(s), due to Not all requirements met"*. If it was not sent back to development the sentence will be: "*This story went well and it was not sent back to development*". The final notification can be seen in Figure 4.17.

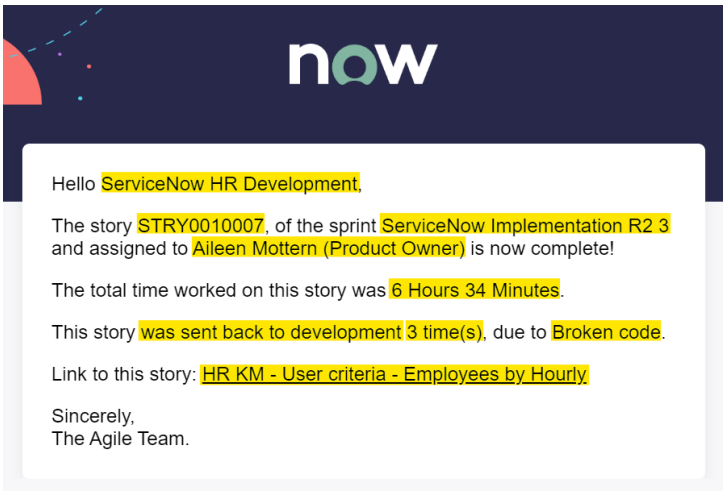


Figure 4.17 – Notification "Story is now complete"

Source – ServiceNow PDI (2024)

To conclude, all notifications are enabled by default within the update set, and when they are installed in a new instance, they automatically start working.

4.3.7. Reports and dashboard

One of the main objectives of this project, described in Chapter 1, is to create an interactive dashboard that will show the performance and quality of the stories, offering a real-time view of the data collected through the new customizations.

In this context, a master dashboard was created to gather relevant aspects of stories, from simple information, such as the sprint they are in, to more in-depth information, such as the average development time of each state, issue tracker, and blocking reasons. Moreover, it will help to compare the estimates given through the story points with the actual development times of a story and identify teams that are under or oversized in terms of workforce.

This subsection presents all the reports created for the master dashboard. Based on this, ServiceNow reports are visualizations that allow users to visualize and analyze current instance data, share it on dashboards, export it to PDF, and email it to users (ServiceNow, 2024j). Moreover, the reports were created based on the data collected by the new customizations combined with data inherent to the stories, such as the sprint and epic they belong to, the assignment group, and the number of story points. This information was

organized in the dashboard following the guidelines provided by Few, described in the Theoretical Framework.

To accomplish this, the report creation process was divided into three sections. The first shows general story data aggregated by status, progress, assignment group, epic, sprint, and creation. The second section covers metrics related to working time, using the fields involved in Time worked customizations. For the third section, reports were created measuring the quality of the developments, using the customized fields *Sent back to development*, *Sent back reason*, and *Number of times sent back*. In this regard, Table 4.3 describes the reports developed and indicates which section of the dashboard they belong to.

Table 4.3 – Reports created to compose the Stories dashboard

Report	Description
Section 1 – General	
Total number of stories	A single score chart showing the total number of stories created.
Total stories reassigned	A single score chart showing the total number of stories that were reassigned, that is, had its assignment group changed.
Stories not yet assigned	A single score chart showing the total number of stories that have not yet been assigned.
Stories not yet released	A single score chart showing the total number of stories that have not yet been released.
Stories by state	Pie chart showing all created stories divided by state.
Stories created by month	Bar chart showing all created stories divided by month.
Stories by progress	Bar chart showing all created stories grouped by progress (epic, state, or assignment group).
Top used themes by story points	Bar chart showing the themes that are most used in a story.
Section 2 – Performance metrics	
Average - Total Time Worked by month	A time series chart showing the average total time worked for all stories per month.
Total number of stories	A single score chart showing the total number of stories created.
Time worked breach stories	A single score chart showing the total number of stories that had their time worked breached.
Max Total Time Worked	A single score chart showing the maximum duration of a story.
Avg Time - Draft	A single score chart showing the average time spent when the story was in the Draft state.
Avg Time - Validation	A single score chart showing the average time spent when the story was in the Validation state.
Avg Time - Work in progress	A single score chart showing the average time spent when the story was in the Work in progress state.
Avg Time - Testing	A single score chart showing the average time spent when the story was in the Testing state.
Average Time Worked - Work in progress by Assignment Group	A bar chart showing the average time spent in the work in progress state per assignment group.
Total Time Worked breached by Assignment Group	A pie chart showing the assignment groups that most exceeded a story's total duration.

Section 3 – Quality metrics	
Total stories sent back	A single score chart showing the total number of stories that was sent back to development.
Avg times sent back	A single score chart showing the average number of times a story was sent back to development.
Stories by sent back reason	A pie chart showing the reasons why a story went backward.
Stories sent back over time	A time series chart showing the number of stories that have been sent back for development over the months.
Sent back to development by assignment group	A bar chart showing the number of stories sent back to development per assignment group.
Sent back to development by story points	A pie chart showing the number of stories sent back to development per story point.

Source – prepared by the author

The main dashboard comprises twenty reports. Six interactive filters, listed in Table 4.4, were created to improve the functionality of these reports in terms of data analysis and make their filtering more dynamic.

Table 4.4 – Interactive filters created for Stories dashboard

Interactive filter	Functionality	Dashboard Tab
Select Time Range	Filters the creation date of records based on time periods.	General, Performance and Quality
Select Story Points	Filters the records according to the number of story points assigned to them.	General and Performance
Select State	Filters the records according to their status.	Performance
Select Epic	Filters the records according to the epic they are in.	Performance and Quality
Select Sprint	Filters the records according to the sprint they are in.	Performance and Quality
Select Assignment Group	Filters the records according to their assignment group.	Performance and Quality
Has the story been blocked?	Filters the records by checking whether the story is blocked or not.	Quality

Source – Prepared by the author

Creating these filters allows for a more dynamic use of the dashboard. Specific filters can be applied to show only the needed data, making it easier to draw conclusions, make comparisons, and even extract the data to an Excel file.

In addition, creating a dashboard on the ServiceNow platform is straightforward. The reports and interactive filters are dragged and dropped into the dashboard tab, where they can be resized, organized, and aligned.

The master dashboard created was entitled Stories. To comply with the project's organization, three tabs were created on the dashboard with the same names as the report sections: General, Performance, and Quality. Figure 4.18 shows the dashboard home page using the General tab, displaying all the reports and filters that compose this tab.

Figure 4.18 – Stories master dashboard – General tab

Source – ServiceNow PDI (2024)

In this dashboard section, the user can reach high-level conclusions about the stories, including their division by state, creation by month, progress, etc., and filtering one report from the other is possible. It is worth remembering that all the reports are dynamic, so clicking on one of the chart bars, for example, will automatically open a list with all the filtered records for the selected column.

Next, the second tab, Performance, shows reports and interactive filters that allow users to draw conclusions about the time spent on stories. Five interactive filters were added to this tab: Select Story Points, Select State, Select Epic, Select Sprint, and Select Assignment Group. Figure 4.19 shows the Performance tab with its reports and interactive filters.

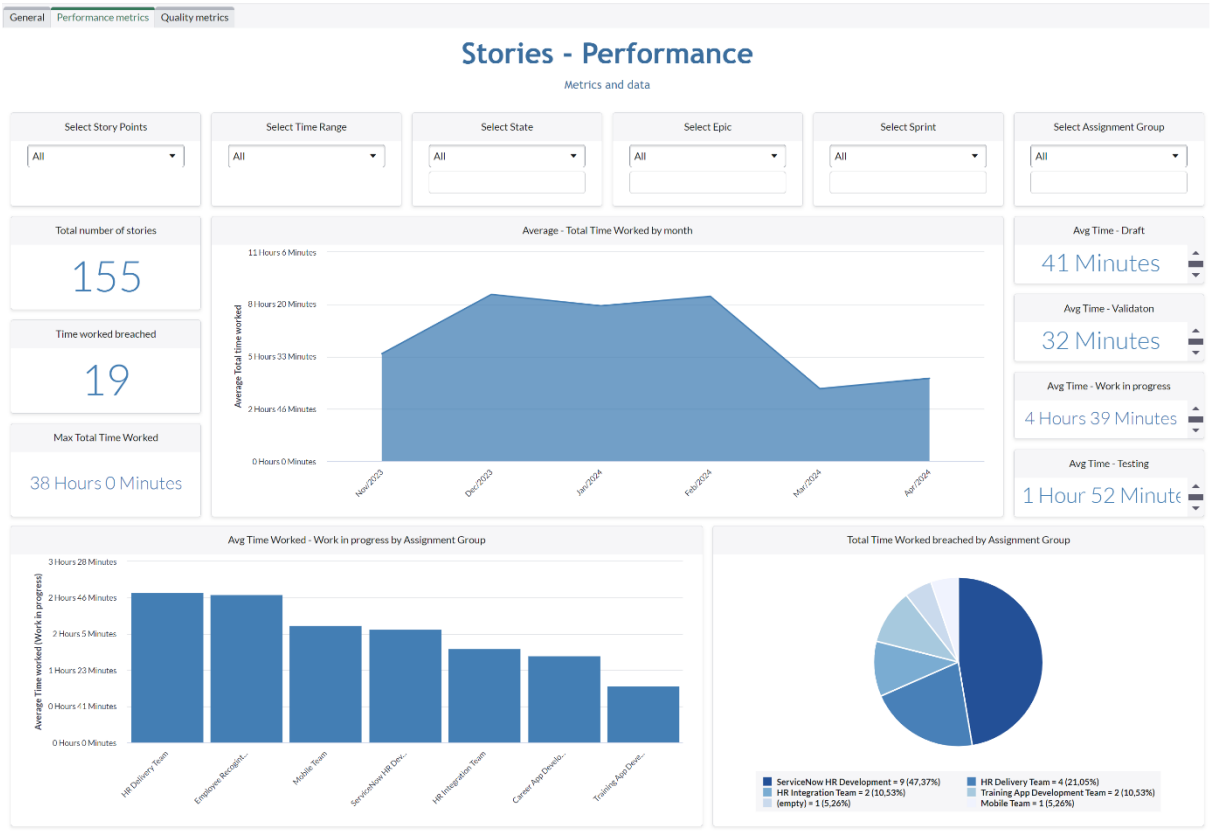


Figure 4.19 – Stories master dashboard – Performance tab

Source – ServiceNow PDI (2024)

In this dashboard section, the user can gain insights into the time spent working on stories. This is made possible through the average total time worked reports, which show the total working time of a story and its breakdown by state.

Furthermore, comparing the total number of stories versus those that have exceeded the total working time is also possible. The *Time worked breached* report has dynamic filters that identify the maximum working time of a story, showing the stories that have exceeded the maximum expected time. For example, for a 5-point story, the estimated working time is between 3 and 8 hours but should be more than 8 hours. In this case, by selecting Story Points = 5 in the interactive filter, this report dynamically adapts to show the stories with a total time of more than 8 hours.

This result fulfills one of the objectives proposed at the start of this project, as it identifies which stories have exceeded their time. It also allows for filtering by sprint, epic, and even assignment group, allowing the identification of teams that performed best during a story development process.

The third and final tab of the master dashboard, Quality, presents reports on the quality metrics used in story development, including blocked stories and stories sent back to be fixed by the development team. Three interactive filters were applied to this dashboard tab: Select Assignment Group, Select Sprint, and Select Epic, as shown in Figure 4.20.

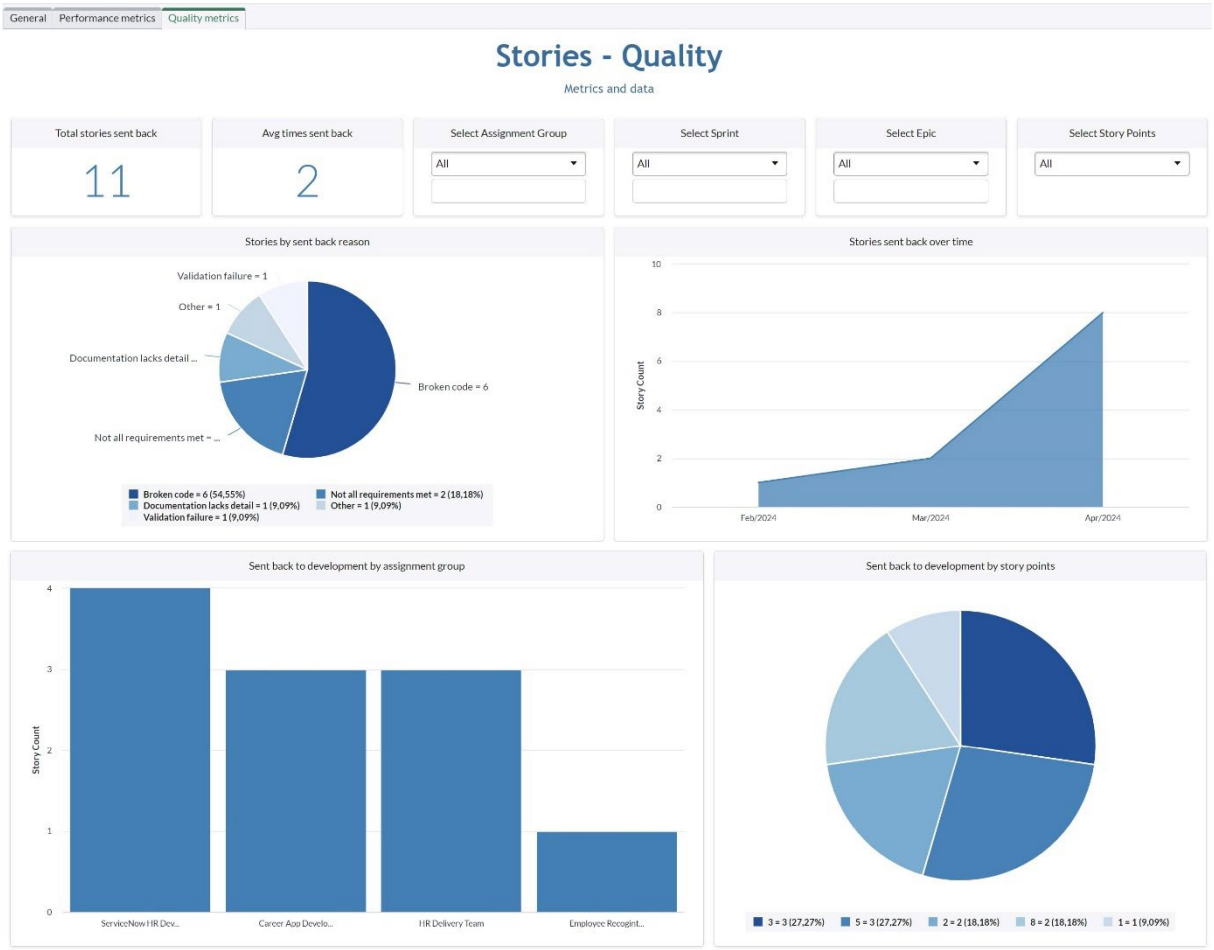


Figure 4.20 – Stories master dashboard – Quality tab

Source – ServiceNow PDI (2024)

This part of the dashboard focuses on the quality of the developments undertaken as part of a story. It can also be filtered by sprint, epic, and story points. It allows the user to conclude the number of stories that have gone backward, the most common reasons for this, and compare it with the average number of times a story goes back to development.

In addition, users can see the variations in the number of stories sent back over time and the breakdown of these stories by assignment group and story points. This also helps to conclude which teams work most efficiently and with the fewest errors.

Managers can also compare results between dashboard tabs. For example, they can draw conclusions about which story points had the most stories with breached time (Performance tab) and compare whether these stories also had any quality problems (Quality tab), such as being sent back for development. It is also possible to compare the story points of the stories sent back for development, leading to other conclusions such as teams with too much work and too few members or teams with insufficient deliveries and errors.

To conclude, reports were developed considering the relevant information to be shown regarding a story, from its conception (Draft state) until its conclusion (Complete state). This

information was combined with the data collected from the new customizations and displayed following the Gestalt principle of proximity, detailed in the Theoretical Framework. It made it possible to compare results and draw conclusions about team performance, quality of work and deliveries, assertiveness in developments, and even reassigned stories.

4.4. PHASE 4: TESTING AND ASSESSMENT

After implementation, the testing phase began. This phase is equally important, as it ensures the integrity of what has been developed by integrating the customizations into the platform. To do this, several manual tests were performed like listing and checking the most critical codes and updates, installing the customizations in a clean instance, and creating more complex Automated Tests Framework (ATFs) that ensure end-to-end functionality.

4.4.1. Manual tests and verification

Despite the existence of ATFs, it is important to remember that during all the development stages, from creating the fields in the table to the automation scripts, manual tests were executed to validate different scenarios. A basic example of a manual test that has been done is to save and try out the codes created until the expected result is obtained without errors.

These tests were necessary to guarantee the integrity and accuracy of the customizations. Usually, one code triggers another, or an action depends on execution until the previous action is completed. One example is a Business Rule that is important for making a Client Script work and populates a variable that will trigger an email notification. This, again, highlights the importance of integrity testing at the time of development.

A checklist, represented in Table 4.5, was created to list the main manual tests to be conducted and evaluate their success rate. This checklist was used and updated during the implementation phase when most of the manual tests were carried out, as it was necessary to validate the code before proceeding to the next stages.

Table 4.5 – Manual testing checklist

Check	Number	Activity	Success
☒	1	New fields visibility, behaviors, auto-filling and automation	Passed
☒	2	Business Rules: conditions and code validations	Passed
☒	3	Client Scripts: conditions and code validations	Passed
☒	4	UI Policies: conditions and code validations	Passed
☒	5	Knowledge base integration and scripted info message	Passed
☒	6	Dashboard functioning when interactive filters are applied	Passed

Source – Prepared by the author

To sum up, manual testing helped to ensure quality and reliability and that customizations meet requirements standards. They also allowed exploring functionalities, ensuring seamless integration of the new customizations into the platform.

4.4.2. Update sets and installation on another instance

The main test was done manually to see if everything that had been done was included by collecting all the partial update sets that had been created during the development of this project.

After checking all the update sets, a batch was created. According to the ServiceNow documentation, batching update sets lets group multiple updates to be previewed and committed together. This helps avoid problems when dealing with multiple update sets, such as committing them in the wrong order or leaving some out (ServiceNow, 2024i).

The batch that gathered all the customizations was called *RS - Agile - Batch total,* and it was used to test all the custom features on a clean instance. After creating the batch, another PDI was requested to retrieve the update sets that were part of it. The installation was successful, the customizations were checked and worked.

4.4.3. Testing through ATFs

After installing the batch update set on the new instance, a series of automated tests, known as Automated Test Frameworks (ATFs), were created to enhance and complement the existing manual and automated testing processes. This ensured that the customizations met expected performance and behavior standards.

Consequently, the initial testing phase involved customizing these ATFs and adapting them to the new features and behaviors. To achieve this, the ATFs focused on the following dimensions: users, groups, platform functionality, and back-office operations. The tests focused on different scenarios and use cases, reflecting how the platform is used, and each ATF was designed to simulate real-world interactions and workflows. The successful execution of all ATFs showed that the platform's new behaviors were working as expected, thus confirming the functionality of the customizations created.

4.4.4. Application in a pilot project

The customizations were presented to the managers of the ServiceNow department of a Portuguese IT consultancy. As a result, approval was given to install these customizations in an internal company project, developed by professionals certified in the ServiceNow platform and guided by experienced managers.

The customizations were applied in this pilot project following the same installation flow performed in the manual tests described in subsection 4.4.2. To this date, the preliminary results of this installation have been very positive.

5. EVALUATION AND DISCUSSION

This phase marks the end of the project and will occur after all the developments have been implemented and thoroughly tested, thus covering Phase 5: Evaluation and Discussion. The purpose is to assess the project holistically, helping to compare expected and actual outcomes and identifying areas for future improvement.

5.1. SURVEY: NEW AGILE FEATURES IN SERVICENOW

A survey, available in Annex A, was conducted to assess the usefulness and impact of the customizations developed in this project, and only certified users of the ServiceNow platform were targeted. This means that all the respondents have at least one or more ServiceNow certifications and work actively with the platform, which makes the data collected more reliable as it is feedback from people who experience the platform.

It first started with a meeting held with the ServiceNow team of a Portuguese IT consultancy. This meeting consisted of a live demonstration of the customizations created, and the audience was then asked to complete the survey.

The survey consisted of eight questions and received 37 responses. The first question asked whether the person had previously used ServiceNow's Agile Development application. The second asked about the person's role as a ServiceNow consultant. These questions were necessary to identify the individual's profile and compare the feedback distributed between the different professional roles.

Preliminary results show that the vast majority of respondents, 31, have already used ServiceNow's Agile Development application at some point. Among the individuals who have already used the Agile Development application, the majority are Developers (18 people), followed by Business Analysts (8), Project Managers (3) and Testers (2). Figure 5.1 illustrates this distribution, where “Q02 – What is your role?” is filtered by “Q01 – Have you ever used ServiceNow’s Agile Development application?” yes results.

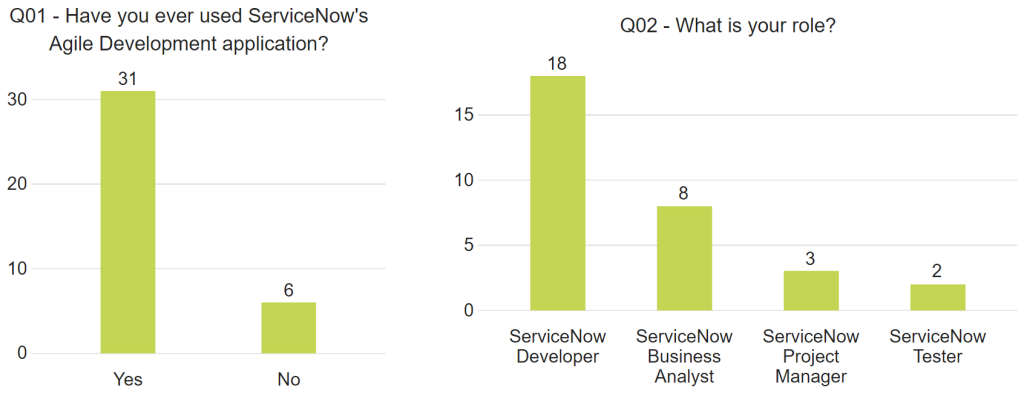


Figure 5.1 – Survey: Respondents role distribution

Source – Prepared by the author

The next questions in the survey relate to the usefulness of the customizations developed. The first of these, Q03, is about the customizations that the individual found most valuable. Here, it can be concluded that the fields related to the time worked on a story (26) added the most value, followed by the fields related to the quality of developments (25). Figure 5.2 compares the distribution of answers to this question.

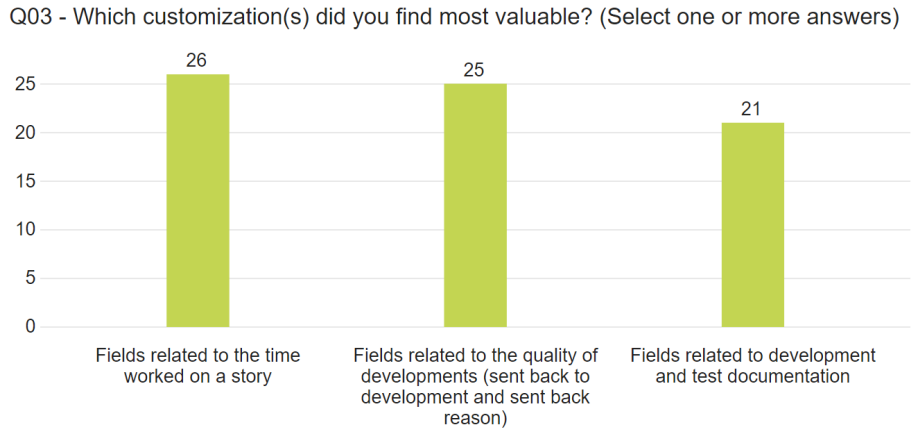


Figure 5.2 – Survey: Most valuable customizations

Source – Prepared by the author

The next question evaluates the most positive aspects of the customization. The answer options were: (i) The dashboard that allows the aggregation of metrics related to stories; (ii) Understanding the most common problems in story development; (iii) Knowing which stories failed and were sent back to development; (iv) Realizing how long it takes to carry out the stories activities and (v) No missing documentation due to the new fields created being mandatory. Once again, the answers are very similarly distributed, which leads to the conclusion that all five aspects were positive and, in some way, added value to the platform. This feedback helps to confirm that the objectives of this project were achieved, illustrated in Figure 5.3.

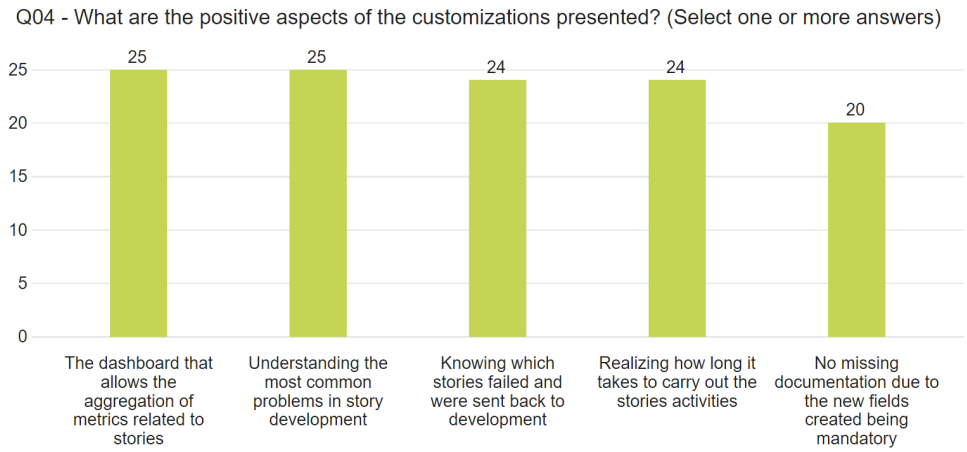


Figure 5.3 – Survey: Most positive aspects of the customizations

Source – Prepared by the author

The next question assesses the usefulness of the new functionalities resulting from customizations. Here, an even distribution of answers was obtained, as shown in Figure 5.4.

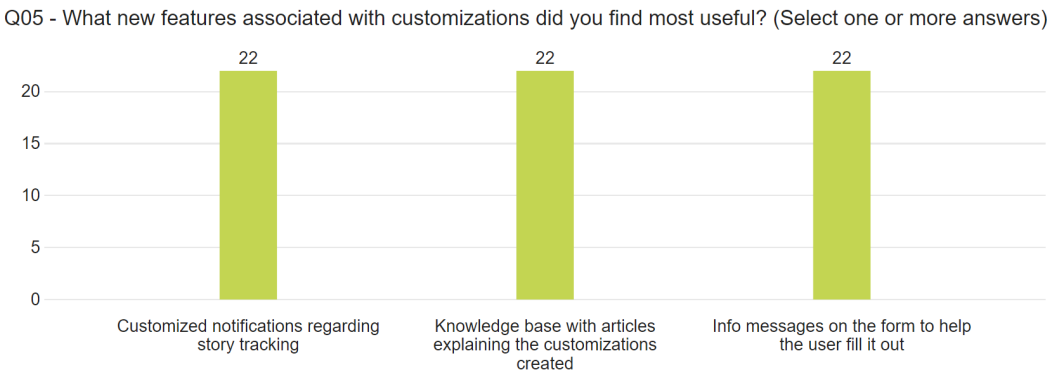


Figure 5.4 – Survey: Most useful associated features

Source – Prepared by the author

This demonstrates that attention to detail benefited in using and aggregating the various functionalities of the ServiceNow platform, and the three main features had an equal weight in the responses.

The next question focused exclusively on the dashboard, the main outcome of this project. It assesses which tabs the respondent found most helpful. The answers obtained, displayed in Figure 5.5, show that most respondents consider the three dashboard tabs equally important, complementing the data between them to gain insights.

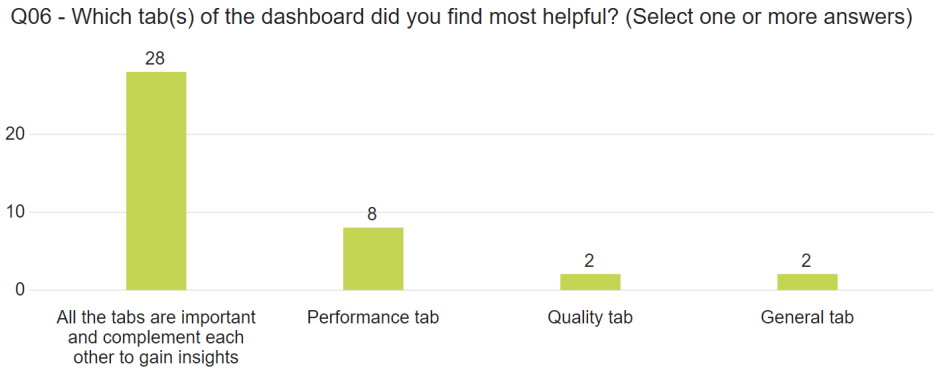


Figure 5.5 – Survey: Most helpful dashboard tabs

Source – Prepared by the author

This result is very significant for several reasons. It shows that the dashboard has been well constructed with the necessary information and filtering. It also verifies that, in line with the suggestions proposed by Eckerson (2012), the information should be on just one page without forcing the user to scroll down. With this in mind, the dashboard was built with three tabs rather than just one main page with all the content.

The next question asks whether the respondent believes the customizations made in this project will improve data-based decision-making and increase project results. Almost all respondents answered yes, representing 96.7%, as demonstrated in Figure 5.6. This feedback is also essential, proving once again that the objectives of the project have been met.

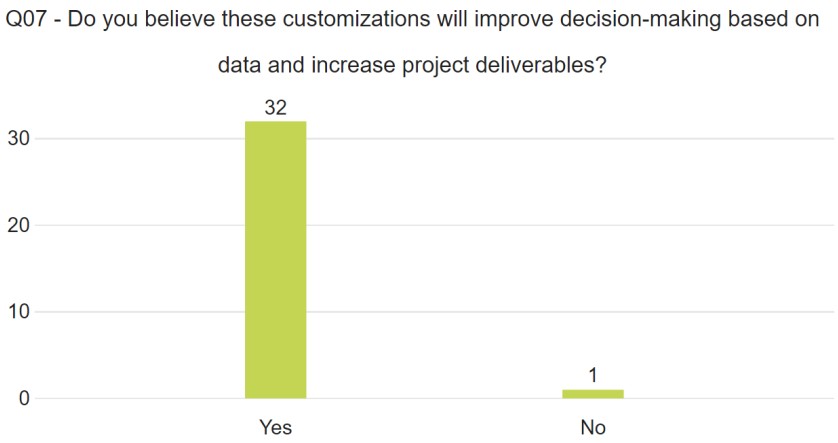


Figure 5.6 – Survey: Improved decision-making based on data

Source – Prepared by the author

The last question refers to any suggestions the respondent might have for improving the developments that have been made. In this case, 4 of the 32 respondents left feedback on process improvements, as shown in Figure 5.7.

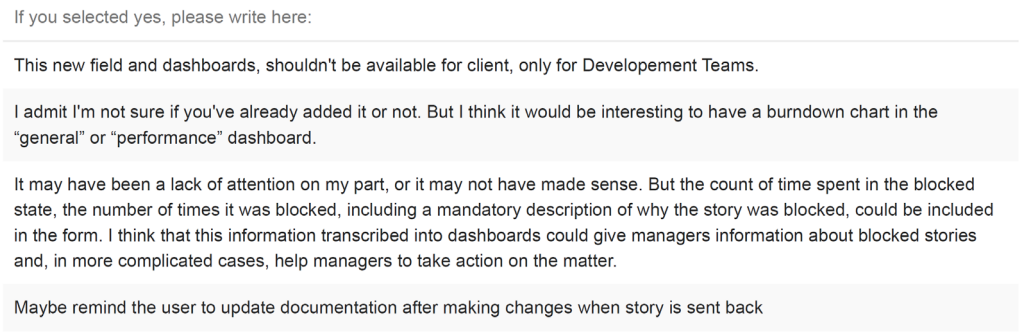


Figure 5.7 – Survey: Answerers feedback

Source – Prepared by the author

Two of the four comments are useful as opportunities for improvement. The first concerns the burndown chart, which should be included in the General or Performance tab. The other reminds the user to update the documentation after making the necessary changes to a story sent back for development.

On the other hand, the other two comments are behaviors that already happen. The first is related to the new fields and dashboards being available only to development teams. This is true because the focus of the customizations carried out has always been on the internal and management teams, where users and managers can gain insights into performance and

quality based on the data. The other is related to a lack of attention on the part of the responder during the project presentation. The findings written there already exist and work.

In sum, the results obtained through this survey show that the customizations presented are all equally important, helping data-based decision-making. In addition, customizations bundled with other ServiceNow functionalities were also considered an aggregated value in this project.

5.2. PROJECT FUNCTIONAL BENEFITS

The manager responsible for the company's internal project was interested in implementing the customizations developed as part of this project. As a result, special feedback was gathered from this manager, looking at points such as functionality, automation, usability, and areas for improvement.

The functional benefits of the project are:

a) There is a possibility of tracking the time spent working on each state of a story. This would allow for understanding the struggles related to each phase and collecting feedback from the teams on the time spent.
b) Better quality control of developments that fail at the testing stage. Better understanding of why these developments failed and the opportunities for improvement.
c) The new mandatory fields for attaching development and testing documentation are useful for tracking what was developed, how it was developed, and the impacts these developments had.
d) The dashboard provides insights and makes it possible to compare the performance of teams in terms of work time and the quality of this work. It also provides an overview of the status of stories.

6. CONCLUSIONS

6.1. SUMMARY OF THE WORK

This project began by creating an acceleration package for the Agile Development 2.0 application on the ServiceNow platform through automation and customization. To this end, research questions were created, and based on these, objectives and milestones were defined to achieve the defined purpose. These milestones provided a path, later divided into phases, to guide the achievement of the main objective during the development of the project.

The first part was to start the project and set the stage. The Agile Development application, including its functionalities and tables, was analyzed in this phase. After that, update sets were created to save the work. These update sets were used to save the developments created and transport them to another instance for testing purposes.

A comprehensive review of existing literature was necessary to align with this project's commitment to best practices, both in theoretical and technical realms. This review embraced the methodologies applied, namely Agile, Scrum, the ITIL framework, and the technical documentation of the ServiceNow platform. Referred to as the Theoretical Framework and detailed in Chapter 3, this literature review formed the foundation of this project.

The next phase details the implementation and development of the project. This phase lists the tables and data validation, the columns and fields created, the new layout of the main table form, the code created in three different ways to automate the behavior of the new fields and features, the creation of a knowledge base to help the user understand and use the new features, the notifications customized to inform the teams of the key stages in the progress of a story, and finally the reports created for each field and the set of them dynamically in the master dashboard to enable data analysis and data-driven decision making.

After this, the tests began. It is worth remembering that most of the manual testing was done during the development phase, as it is important to ensure that the previous customization works before creating a new one. In addition, automatic tests were also used in the ServiceNow testing framework, performing a step-by-step completion of the forms and acting as a double-check on the customizations created.

Implementing the customizations described and tested in Chapter 4, driven by the success of the initially established work plan stages, achieved the objectives of the project. Additionally, a survey dedicated to certified ServiceNow users confirmed the effectiveness and usability of the customized Agile Development application.

Finally, the project resulted in a paper accepted for publication at the 16th Mediterranean Conference on Information Systems (MCIS 2024) and the 24th Conference of the Portuguese Association for Information Systems (CAPSI 2024), further enhancing its success.

6.2. LIMITATIONS

The advantages and benefits that the customizations described in this project represent for users were designed for companies with an active ServiceNow license and the Agile Development 2.0 application. In addition, the customizations have been developed to be easily installed in other ServiceNow instances.

Another limitation of this project was working on a Personal Development Instance (PDI). This instance only includes demo data, which was used to assess the integrity and compliance of the data in the customized fields. Despite not working with real data initially, the customizations were installed in an IT consultancy's pilot project to overcome this barrier. With this, it was possible to start analyzing the dashboards as real data began to be collected.

Another constraint was the lack of time to evaluate the results from the time-over-time dashboard, as there was not enough data to simulate it, and the data from the pilot project was still very early. To overcome this, the satisfaction survey was used to collect qualitative data on the satisfaction and usability of the new customizations.

Moreover, this project customizes software through a prototype built in a development environment and then passed on for testing with demonstration data. In the future, the idea is to implement it in a company's production environment to collect live data and feedback on the customizations created.

In addition, some aspects and integrations with other modules of the ServiceNow platform had to be left out of the scope of this project due to the need to finish it in the stipulated time.

6.3. FUTURE WORK

The Agile methodology is constantly being used and improved, which opens up space to increase the customizations developed in this project. In addition, ServiceNow has other modules and functionalities that could not be explored due to a lack of time and resources. These include:

- Integrate the Agile Development application with the Strategic Portfolio Management (SPM) module.
- Create a schedule job to automatically stop the counter when the user is out of the office in their time zone. Assess whether a new table is needed or should be integrated with the information already provided by the out-of-the-box functions and fields.

Finally, a suggestion obtained from the survey was to include a burndown chart in the General tab to show the amount of work completed in an epic or sprint.

REFERENCES

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J. (2017). *Agile Software Development Methods: Review and Analysis* (arXiv:1709.08439). arXiv. https://doi.org/10.48550/arXiv.1709.08439

Adyrbai, D. (2021, July 1). *ITIL Practices in 2000 words: Incident management, service desk and service request management*. https://www.axelos.com/resource-hub/white-paper/itil-2000-words-incident-service-desk-request-management

Akbar, R., Silvana, M., Hersyah, M. H., & Jannah, M. (2020). Implementation of Business Intelligence for Sales Data Management Using Interactive Dashboard Visualization in XYZ Stores. *2020 International Conference on Information Technology Systems and Innovation (ICITSI)*, 242–249. https://doi.org/10.1109/ICITSI50517.2020.9264984

Alqudah, M., & Razali, R. (2016). A Review of Scaling Agile Methods in Large Software Development. *International Journal on Advanced Science, Engineering and Information Technology*, *6*, 828. https://doi.org/10.18517/ijaseit.6.6.1374

Al-saqqa, S., Sawalha, S., & Abdel-Nabi, H. (2020). Agile Software Development: Methodologies and Trends. *International Journal of Interactive Mobile Technologies (iJIM)*, *14*(11), 246. https://doi.org/10.3991/ijim.v14i11.13269

Antonio, A. M. (2023, October 17). O que é ITIL 4? O Guia Definitivo da ITIL 4. *PMG Academy*. https://www.pmgacademy.com/blog/o-que-e-itil-4-o-guia-definitivo-da-itil-4/

Anwer, F., Aftab, S., Waheed, U., & Muhammad, S. (2017). Agile Software Development Models TDD, FDD, DSDM, and Crystal Methods: A Survey. *INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY SCIENCES AND ENGINEERING*, *8*, 1–10.

Beck, K., Mike, B., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., C. Martin, R., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifesto for Agile Software Development*. https://agilemanifesto.org/

Bernd. (2021, June 18). Stop [time_worked] if the task is resolved, closed or canceled. *Try. Learn. Grow. Repeat.* https://www.trylearngrowrepeat.com/stop-time_worked-if-the-task-is-resolved-closed-or-canceled/

Boothpoor, M. (2023, August 3). *Components installed with Agile Development 2.0*. https://docs.servicenow.com/bundle/washingtondc-it-business-management/page/product/agile-development/reference/components-installed-with-agile-2.0.html

Chauhan, C., Dakoliya, M., & Sharma, R. K. (2023). Use of Fibonacci Sequence in Project Estimation. *Indian Journal Of Science And Technology*, *16*(33), 2649–2652. https://doi.org/10.17485/IJST/v16i33.1534

Choudhary, B., & Rakesh, S. K. (2016). An approach using agile method for software development. *2016 International Conference on Innovation and Challenges in Cyber Security (ICICCS-INBUSH)*, 155–158. https://doi.org/10.1109/ICICCS.2016.7542304

Eckerson, W. W. (2012). *Performance Dashboards: Measuring, Monitoring, and Managing Your Business* (1st ed.). Wiley. https://doi.org/10.1002/9781119199984

Elbanna, A., & Sarker, S. (2016). The Risks of Agile Software Development: Learning from Adopters. *IEEE Software*, *33*(5), 72–79. https://doi.org/10.1109/MS.2015.150

Few, S. (2013). *Information Dashboard Design* (Second Edition).

Giuffrida, R., & Dittrich, Y. (2015). A conceptual framework to study the role of communication through social software for coordination in globally-distributed software teams.

Information and Software Technology, *63*, 11–30. https://doi.org/10.1016/j.infsof.2015.02.013

Gouryraj, S., Kataria, S., & Swvigaradoss, J. (2021). *Service Level Agreement Breach Prediction in ServiceNow*. 689–698. Scopus. https://doi.org/10.1109/ICIRCA51532.2021.9544916

Hasibović, A. Ć., Tanović, A., & Granulo, A. (2023). The importance of ITIL4 adoption for IT service management in insurance companies. *2023 46th MIPRO ICT and Electronics Convention (MIPRO)*, 1341–1346. https://doi.org/10.23919/MIPRO57284.2023.10159950

Hinderks, A., Domínguez Mayo, F. J., Thomaschewski, J., & Escalona, M. J. (2022). Approaches to manage the user experience process in Agile software development: A systematic literature review. *Information and Software Technology*, *150*, 106957. https://doi.org/10.1016/j.infsof.2022.106957

Hollifield, B. (2024, January). *Add to Update Set Utility*. Add to Update Set Utility. https://developer.servicenow.com/connect.do#!/share/contents/9824957_add_to_update_set_utility?t=PRODUCT_DETAILS

Iden, J., Eikebrokk, T. R., & Marrone, M. (2020). Process reference frameworks as institutional arrangements for digital service innovation. *International Journal of Information Management*, *54*, 102150. https://doi.org/10.1016/j.ijinfomgt.2020.102150

ISO. (2016, May 23). *ISO/IEC 20000-1:2011*. ISO. https://www.iso.org/standard/51986.html

Itzik, D., & Roy, G. (2023). Does agile methodology fit all characteristics of software projects? Review and analysis. *Empirical Software Engineering*, *28*(4), 105. https://doi.org/10.1007/s10664-023-10334-7

Jammalamadaka, K., & Krishna, V. R. (2013). *Agile Software Development and Challenges*.

Kashishku. (2023, August 28). *Fibonacci Series*. GeeksforGeeks. https://www.geeksforgeeks.org/fibonacci-series/

Kent, B., & Andres, C. (2005). *Extreme programming explained: Embrace change* (Second Edition).

Klentsova, A., Skrynnik, O., & Jouravlev, R. (2020, April 17). *ITIL Guiding Principles for Continual Improvement*. https://www.axelos.com/resource-hub/white-paper/itil-guiding-principles-for-continual-improvement

Kodmelwar, D. M. K., Futane, D. P. R., Pawar, P. S. D., Lokhande, P. S. A., & Dhanure, P. S. P. (2022). A Comparative Study of Software Development Waterfall, Spiral and Agile Methodology. *Journal of Positive School Psychology*, *6*(3), Article 3.

Larman, C., & Vodde, B. (2013). *Scaling agile development*. *26*, 8–12.

Marrone, M., & Kolbe, L. M. (2011). Impact of IT Service Management Frameworks on the IT Organization. *Business & Information Systems Engineering*, *3*(1), 5–18. https://doi.org/10.1007/s12599-010-0141-5

Mcloughlin, S., Scheepers, H., & Wijesinghe, R. (2014, December 8). *Benefit Planning Management for ITSM: Evaluating Benefit Realization Frameworks*. https://doi.org/10.13140/2.1.2844.9604

Melendez, K., Dávila, A., & Pessoa, M. (2016). Information technology service management models applied to medium and small organizations: A systematic literature review. *Computer Standards & Interfaces*, *47*, 120–127. https://doi.org/10.1016/j.csi.2015.10.001

Micreiros. (2013, September 30). *O que é o ITIL (Information Technology Infrastructure Library)? – Micreiros.com*. https://micreiros.com/o-que-e-o-itil-information-technology-infrastructure-library/

Milton, N., & Lambe, P. (2020). *The Knowledge Manager's Handbook: A step-by-step guide to embedding effective knowledge management in your organization* (2nd ed.). KoganPage.

Mishra, A., & Alzoubi, Y. I. (2023). Structured software development versus agile software development: A comparative analysis. *International Journal of System Assurance Engineering and Management*, *14*(4), 1504–1522. https://doi.org/10.1007/s13198-023-01958-5

Orlovskyi, D., & Kopp, A. (2020). *A Business Intelligence Dashboard Design Approach to Improve Data Analytics and Decision Making*.

Pang, A. (2022, September 26). Top 10 ITSM Software Vendors, Market Size and Market Forecast 2021-2026. *Apps Run The World*. https://www.appsruntheworld.com/top-10-it-service-management-software-vendors-and-market-forecast/

Pappas, L., & Whitman, L. (2011). Riding the Technology Wave: Effective Dashboard Data Visualization. In M. J. Smith & G. Salvendy (Eds.), *Human Interface and the Management of Information. Interacting with Information* (Vol. 6771, pp. 249–258). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-21793-7_29

Rodríguez, P., Mäntylä, M., Oivo, M., Lwakatare, L. E., Seppänen, P., & Kuvaja, P. (2019). Advances in Using Agile and Lean Processes for Software Development. In *Advances in Computers* (Vol. 113, pp. 135–224). Elsevier. https://doi.org/10.1016/bs.adcom.2018.03.014

Romero, V. (2024, February 1). *Using the Task table*. https://docs.servicenow.com/bundle/washingtondc-application-development/page/administer/task-table/concept/c_TaskTable.html

Ruiz, M., Moreno, J., Dorronsoro, B., & Rodriguez, D. (2018). Using simulation-based optimization in the context of IT service management change process. *Decision Support Systems*, *112*, 35–47. https://doi.org/10.1016/j.dss.2018.06.004

Saarikallio, M., & Tyrväinen, P. (2022). Quality culture boosts agile transformation—Action research in a business-to-business software business. *Journal of Software: Evolution and Process*, *35*. https://doi.org/10.1002/smr.2504

Sahid, A., Maleh, Y., Belaissaoui, M., Sahid, A., Maleh, Y., & Belaissaoui, M. (2020). Strategic Agility for IT Service Management: A Case Study. In *Strategic Information System Agility: From Theory to Practices* (pp. 93–116). Emerald Publishing Limited. https://doi.org/10.1108/978-1-80043-810-120211006

Sandeep, R. C., Sánchez-Gordón, M., Colomo-Palacios, R., & Kristiansen, M. (2022). Effort Estimation in Agile Software Development: A Exploratory Study of Practitioners' Perspective. In A. Przybyłek, A. Jarzębowicz, I. Luković, & Y. Y. Ng (Eds.), *Lean and Agile Software Development* (pp. 136–149). Springer International Publishing. https://doi.org/10.1007/978-3-030-94238-0_8

Schmidt, C. (2016). *Agile Software Development Teams*. Springer International Publishing. https://doi.org/10.1007/978-3-319-26057-0

Schwaber, K., & Beedle, M. (2002). *Agile Software Development with Scrum*. Prentice Hall.

Serebryantseva, V. (2022, June 30). *Agile vs Waterfall: Which Development Approach is Better?* PixelPlex. https://pixelplex.io/blog/agile-waterfall-software-development/

Serrano, J., Faustino, J., Adriano, D., Pereira, R., & da Silva, M. M. (2021). An it service management literature review: Challenges, benefits, opportunities and implementation practices. *Information (Switzerland)*, *12*(3). Scopus. https://doi.org/10.3390/info12030111

ServiceNow. (2022a). *Automated Test Framework (ATF)*. https://docs.servicenow.com/bundle/vancouver-application-development/page/administer/auto-test-framework/concept/automated-test-framework.html
ServiceNow. (2022b). *Knowledge Management*. https://docs.servicenow.com/bundle/vancouver-servicenow-platform/page/product/knowledge-management/topic/p_KnowledgeManagment.html
ServiceNow. (2023a). *Agile Development 2.0*. https://docs.servicenow.com/bundle/vancouver-it-business-management/page/product/agile-development/reference/agile-landing-page.html
ServiceNow. (2023b). *IT Service Management*. https://docs.servicenow.com/bundle/vancouver-it-service-management/page/product/it-service-management/reference/r_ITServiceManagement.html
ServiceNow. (2023c). *Personal Developer Instances*. https://developer.servicenow.com/dev.do#!/learn/learning-plans/utah/new_to_servicenow/app_store_learnv2_buildmyfirstapp_utah_personal_developer_instances
ServiceNow. (2023d). *System update sets*. https://docs.servicenow.com/bundle/vancouver-application-development/page/build/system-update-sets/concept/system-update-sets.html
ServiceNow. (2023e). *What is ServiceNow?* ServiceNow. https://www.servicenow.com/what-is-servicenow.html
ServiceNow. (2024a). *Business Rules*. https://developer.servicenow.com/dev.do#!/learn/courses/vancouver/app_store_learnv2_scripting_vancouver_scripting_in_servicenow/app_store_learnv2_scripting_vancouver_server_side_scripting/app_store_learnv2_scripting_vancouver_business_rules
ServiceNow. (2024b). *Client scripts*. https://docs.servicenow.com/bundle/washingtondc-application-development/page/script/client-scripts/concept/client-scripts.html
ServiceNow. (2024c). *Email and SMS notifications*. https://docs.servicenow.com/bundle/washingtondc-platform-administration/page/administer/notification/concept/c_EmailNotifications.html
ServiceNow. (2024d). *GlideDuration*. https://developer.servicenow.com/dev.do#!/reference/api/vancouver/server/c_GlideDurationScopedAPI
ServiceNow. (2024e). *GlideRecord*. https://developer.servicenow.com/dev.do#!/learn/courses/vancouver/app_store_learnv2_scripting_vancouver_scripting_in_servicenow/app_store_learnv2_scripting_vancouver_server_side_scripting/app_store_learnv2_scripting_vancouver_gliderecord
ServiceNow. (2024f). *Knowledge Management*. https://docs.servicenow.com/bundle/washingtondc-servicenow-platform/page/product/knowledge-management/concept/knowledge-management.html
ServiceNow. (2024g). *Time worked fields*. https://docs.servicenow.com/bundle/vancouver-platform-administration/page/administer/time/reference/r_TimeWorkedFields.html
ServiceNow. (2024h). *UI policies*. https://docs.servicenow.com/bundle/washingtondc-platform-administration/page/administer/form-administration/task/t_CreateAUIPolicy.html
ServiceNow. (2024i). *Update set batching*. https://docs.servicenow.com/bundle/washingtondc-application-development/page/build/system-update-sets/hier-update-sets/concept/us-hier-overview.html

ServiceNow. (2024j). *Using reporting*. https://docs.servicenow.com/bundle/washingtondc-now-intelligence/page/use/reporting/concept/c_GenerateReports.html

Simpson, J., Wilkin, C., Campbell, J., Keating, B., & Moore, S. (2016). Best practices or improvisation in system change? An exploratory study. *Pacific Asia Conference on Information Systems (PACIS)*, *27*. https://www.academia.edu/download/62923674/Best_Practices_or_Improvisation_in_System_Change_An_Exploratory_Study20200412-68991-pvyrup.pdf

Srivastava, A., Mehrotra, D., Kapur, P. K., & Aggarwal, A. G. (2020). Analytical evaluation of agile success factors influencing quality in software industry. *International Journal of System Assurance Engineering and Management*, *11*(2), 247–257. https://doi.org/10.1007/s13198-020-00966-z

Toro, Á. de. (2022, August 18). *¿Qué es Scrum? Conoce el Framework que agiliza el Trabajo en Equipo*. https://www.escueladenegociosydireccion.com/revista/business/scrum-framework-agiliza-trabajo-equipo/

Toulson, T. (2019, February 23). *What is a ServiceNow Instance?* CodeCreative | A ServiceNow Blog. https://codecreative.io/guides/getting-started-on-servicenow/what-is-a-service now-instance/

Tran-Ngoc, H., Le-Xuan, T., Khatir, S., De Roeck, G., Bui-Tien, T., & Abdel Wahab, M. (2023). A promising approach using Fibonacci sequence-based optimization algorithms and advanced computing. *Scientific Reports*, *13*(1), 3405. https://doi.org/10.1038/s41598-023-28367-9

Widianto, A., & Subriadi, A. P. (2022). IT service management evaluation method based on content, context, and process approach: A literature review. *Procedia Computer Science*, *197*, 410–419. https://doi.org/10.1016/j.procs.2021.12.157

ANNEXES

A. Survey: New Agile Development application features in ServiceNow

Dear participant,

This survey, part of my Master's Thesis in Information Management with a specialization in Business Intelligence at NOVA IMS, has a significant objective. It aims to create an acceleration package through customizations in ServiceNow's Agile Development application. This will not only enable users and managers to see the time taken to develop a user story and the quality and compliance of their developments but also provide crucial data for making data-driven decisions at the management level.

Your participation on this study is highly appreciated. If you have additional questions about this survey, please feel free to email me at 20220399@novaims.unl.pt.

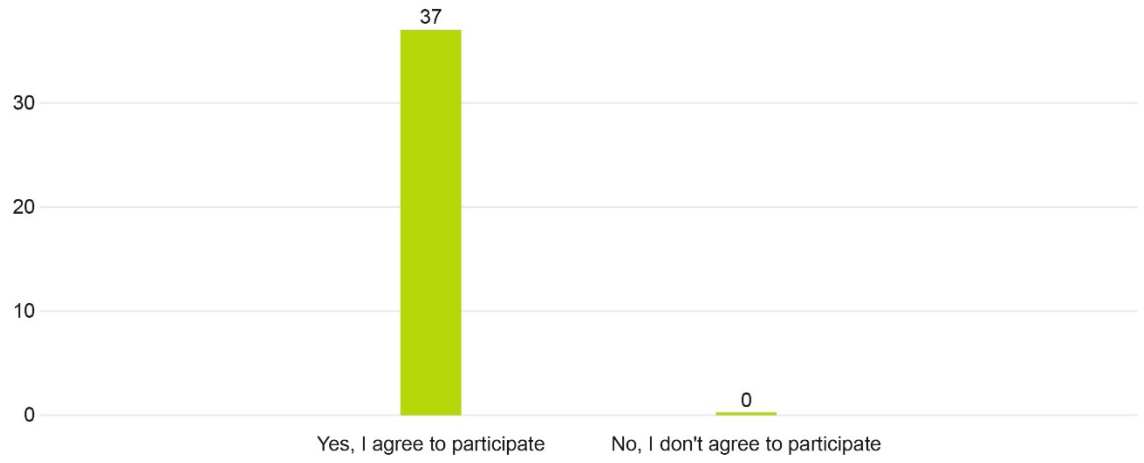
Thank you in advance for your attention and collaboration,

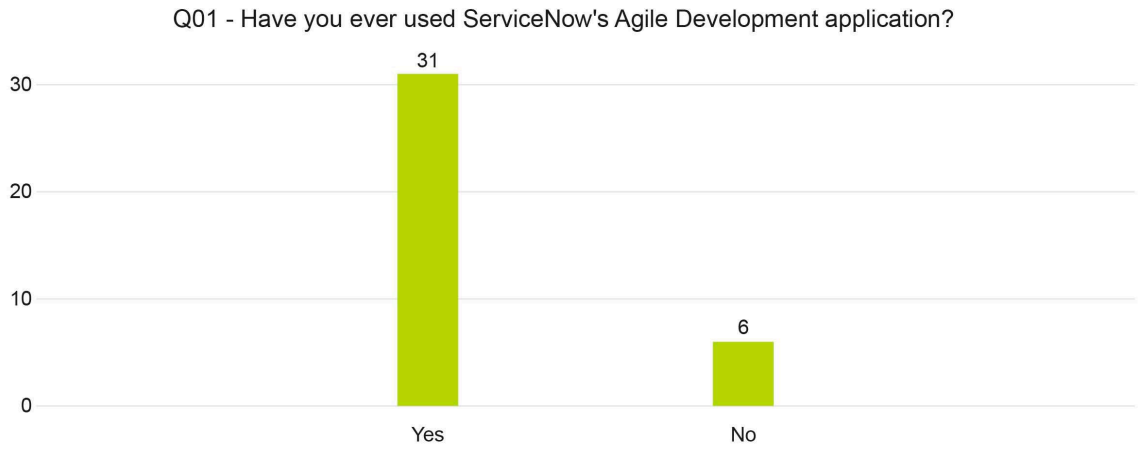
Ricardo Raab Saenger

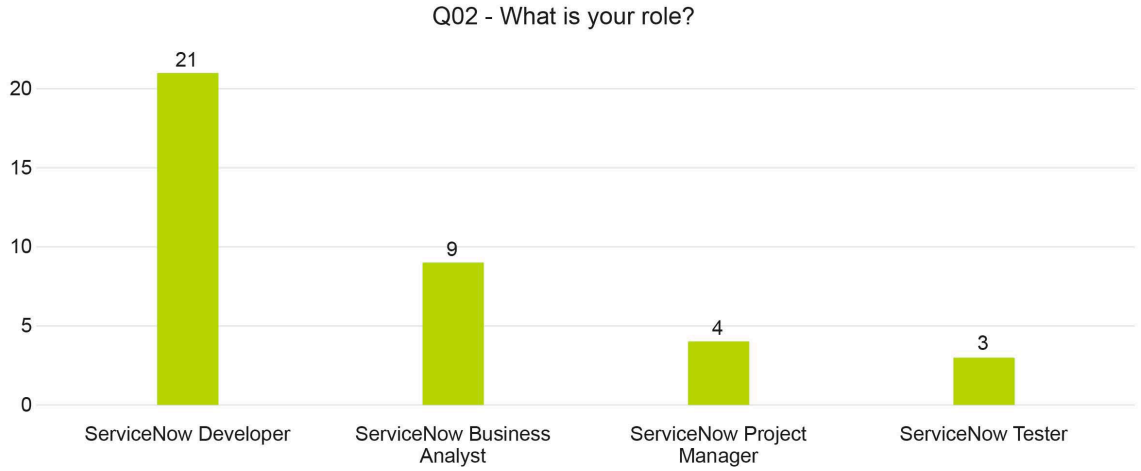
Informed Consent Form:

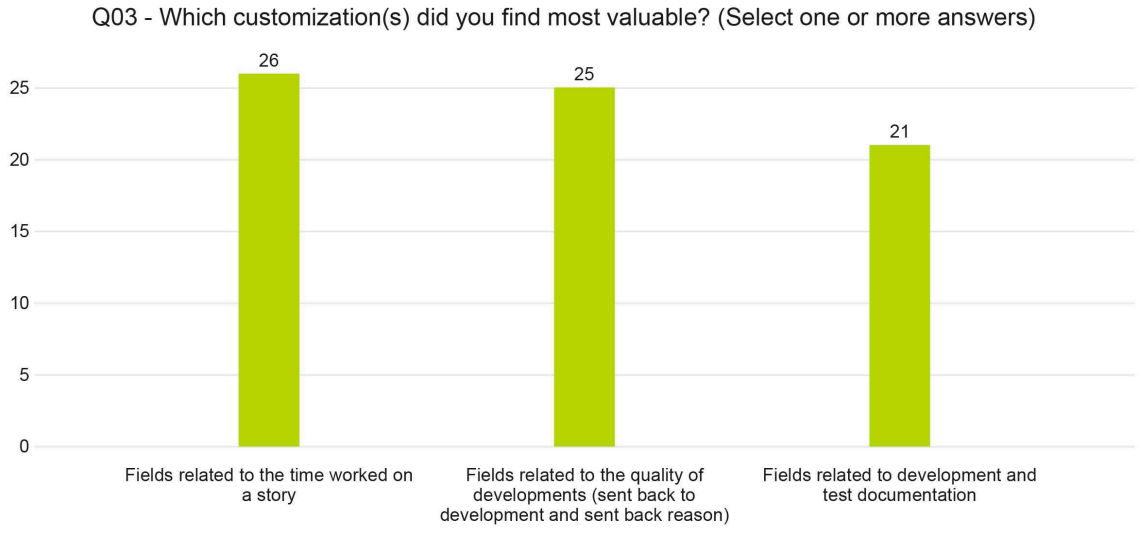
By consenting to participate in this study, you declare the following:

- You are 18 years of age or older.
- Your participation is voluntary and for academic purposes only.
- You are free to stop participating at any time without penalty.
- All data is anonymous and confidential.

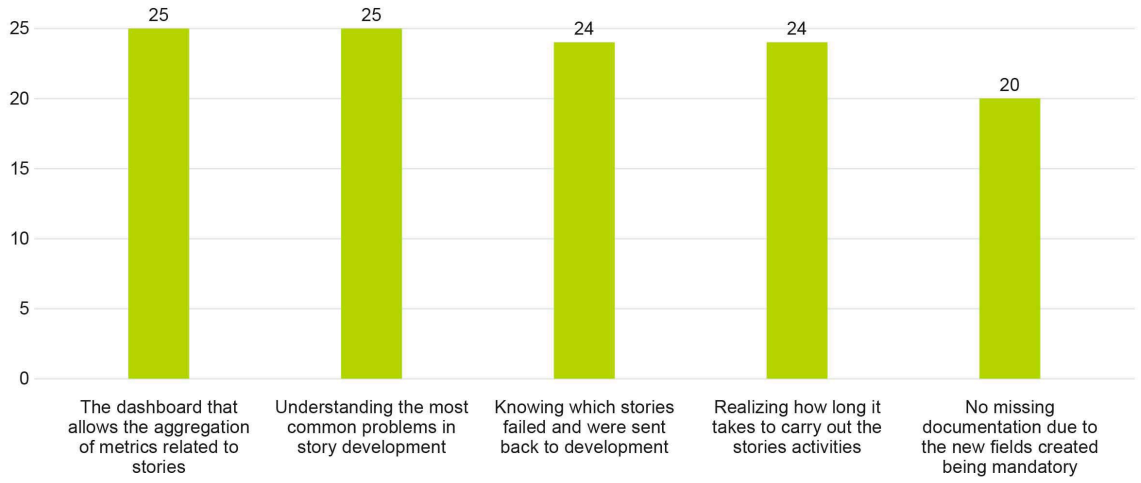



Q01 - Have you ever used ServiceNow's Agile Development application?
30
20
10
0
31
6
Yes
No

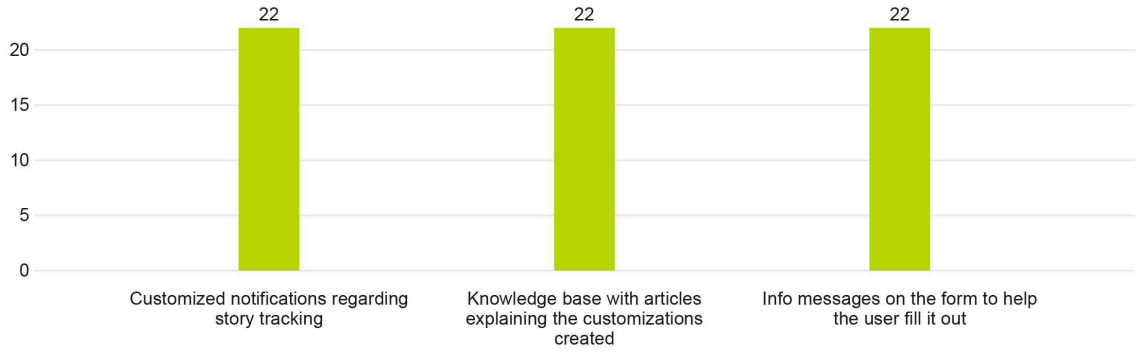

Q02 - What is your role?
20
15
10
5
0
21
9
4
3
ServiceNow Developer
ServiceNow Business Analyst
ServiceNow Project Manager
ServiceNow Tester


Q03 - Which customization(s) did you find most valuable? (Select one or more answers)
25
20
15
10
5
0
26
25
21
Fields related to the time worked on a story
Fields related to the quality of developments (sent back to development and sent back reason)
Fields related to development and test documentation

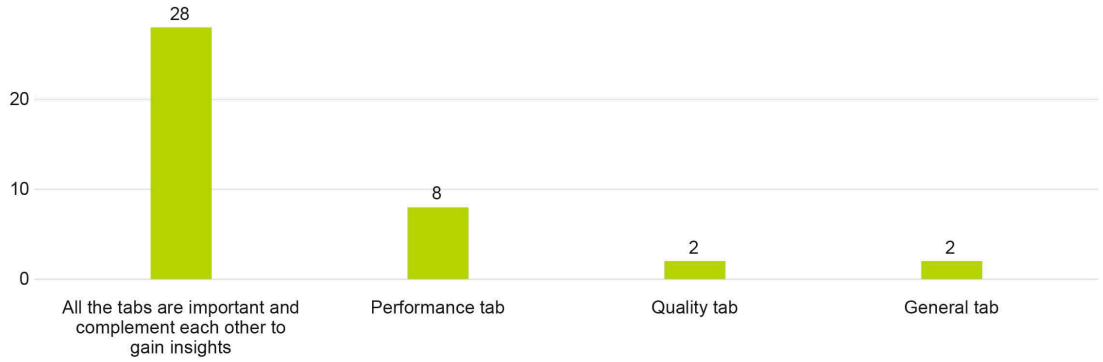
Q04 - What are the positive aspects of the customizations presented? (Select one or more answers)



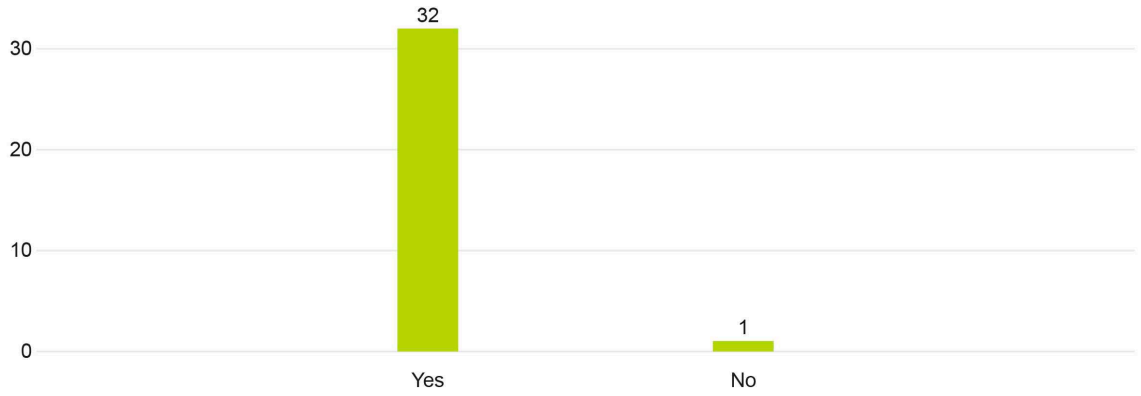
Q05 - What new features associated with customizations did you find most useful? (Select one or more answers)



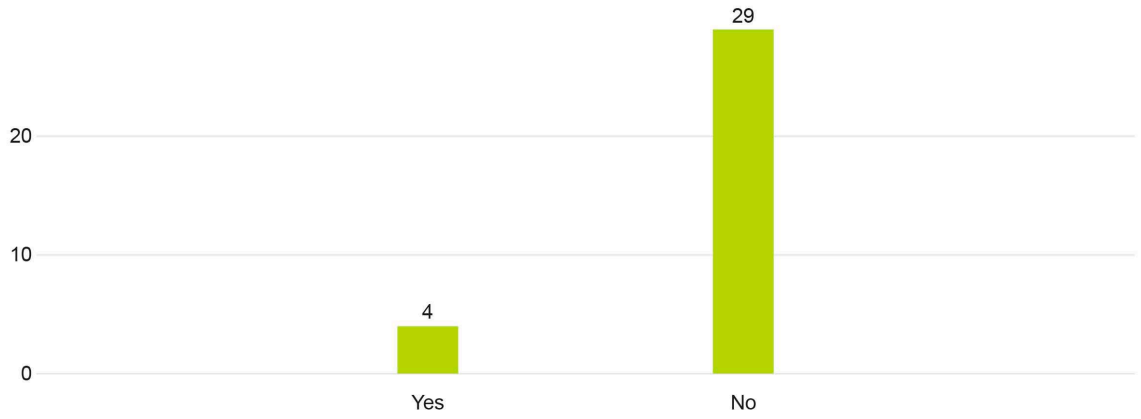
Q06 - Which tab(s) of the dashboard did you find most helpful? (Select one or more answers)



Q07 - Do you believe these customizations will improve decision-making based on data and increase project deliverables?



Q08 - Do you have any suggestions for improving the processes presented?



If you selected yes, please write here:

This new field and dashboards, shouldn't be available for client, only for Developement Teams.

Admito que não tenho a certeza se já adicionaste ou não. Mas acho que seria interessante teres um burndown chart no dashboard "general" ou "performance".

It may have been a lack of attention on my part, or it may not have made sense. But the count of time spent in the blocked state, the number of times it was blocked, including a mandatory description of why the story was blocked, could be included in the form. I think that this information transcribed into dashboards could give managers information about blocked stories and, in more complicated cases, help managers to take action on the matter.

Maybe remind the user to update documentation after making changes when story is sent back

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa