Saint-Petersburg State University
Faculty of Applied Mathematics and Control Processes

Development of intelligent tools for evaluating software development tasks within the SCRUM framework

Gerasimets Bogdana
student of the 2nd year of master's degree
in the field of Game Theory and
operations research, group 19. M09-PU

Scientific supervisor:

Dr. Sc. (Phys.-Math.) Krylatov Alexander

Saint-Petersburg, 2021

Table of contents

Introduction

Flexible development methodologies were originally created for efficient work organization in the field of software development, but in recent years they have been increasingly used in other fields, which previously had other approaches. The activation of the use of such methodologies forms new conditions for their functioning and is an exceptional interest for an in-depth study of these methodologies in order to optimize them for use in different professional areas. Our research work aims to study one of the mechanisms of functioning of flexible methodologies and to optimize the work of such a mechanism through the development of intelligent tools for decision-making.

In our research, we will consider one of the key mechanisms of agile development methodologies - task evaluation of software development.

Task estimation in software development in Agile methodology [33] is an estimation of the conditional amount and time of work on a particular task, the amount of which is determined by the development team, based on the experience of similar tasks, as well as on the preliminary action plan for performing such task.

Task evaluation is an important part of any software development. The specific problem chosen is the lack of scientifically based methods for optimizing task evaluation. Tasks are estimated intuitively or experientially, there is always a margin of error, deadlines are violated, and as consequence budgets are not planned correctly, which affects operational metrics. In our research, we will identify best practices of task estimation, which will help to optimize the planning and forecasting process, which in turn will improve operational forecasts, and, accordingly, budget planning. Such practices will make the development process more transparent, which will help optimize the operational part of management.

The relevance of this research work is the need to develop a scientific approach to the evaluation of development tasks by developing new methods to evaluate software development tasks.

The main goal of our research is to create an IT product, the main functionality of which will be an automatic evaluation of tasks for software development. Such a product can be applied to tracking systems that require task evaluation. For the basis of such a product, we will take empirical data obtained during an experiment with a test team and an algorithm that is developed in the framework of the method of fractal approximation. This IT product will be a program that can learn by loading new development tasks and it will learn to evaluate them in Fibonacci numbers.

To achieve the goal of our research it is necessary to perform the following tasks:

1. Investigate the need to automate the evaluation of development tasks and current implementations.
2. Investigate methods for estimating software development tasks.
3. Analyze empirical data from the study of the test team's performance.
4. Prepare a data set for its use in program training.
5. Prepare a technological base for the creation of an IT product.
6. Develop an IT product that meets the purpose of the research.

The value of the research product is considered in two ways:

- Value to the development team - reduced time to evaluate tasks, and, as a result, increased productivity of the development team.
- Value for business owners - the work of development teams will become predictable and transparent, allowing business owners to make accurate forecasts about the implementation of products.

Implementation of all the tasks planned in the framework of this research will contribute to the development of methods for estimating tasks for software development by identifying typical development tasks for estimation, optimization of methods for estimating such tasks by Fibonacci numbers, approximation of numerical characteristics, and qualitative properties of the estimation of development tasks. The result of our work will be an IT product that will offer the most optimal estimation of tasks with the help of the given model.

The basic methodology for creating the product will be a class of artificial intelligence methods - Machine Learning (hereinafter-ML). This product will be written in the programming language Python. The choice of this language is due to the possibilities of machine learning, inherent in this language - packages Keras and NumPy recognized by the world scientific community.

In the first chapter of our work, we will analyze the basic concepts of the Agile method, the concept of Scrum framework, and the Delphi method in poker planning. For a detailed analysis of Agile [33], we will take the Agile manifesto, otherwise known as the agile software development manifesto, created by Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Ken Schwaber, Jeff Sutherland, Dave Thomas.

To parse the Scrum framework, we need to understand why Agile is used in software development, and we turn to the logic of the Cynefin framework, which was created and described by David Snowden as part of his work "A Leader's Framework for Decision Making" [24]. We turn to the work of "Cynefin, statistics and decision analysis" [9] by S. French from the University of Warwick, Coventry (UK), who analyzed the Cynefin framework through the prism of statistics and decision analysis. In this study, the logic of how software development is carried out within agile methodologies is well traced. Having studied the basic principles of the Scrum framework, we turn to the obligatory activity within this framework - poker planning, as this activity is the key to evaluate the software development tasks. The Delphi method [12] was taken as the basis for poker planning, it has been improved and today most development teams evaluate their tasks in story points, and in this research, we will address the use of Fibonacci numbers in story points. We investigate the Delphi method developed by the RAND Corporation and presented in Dalkey N., Helmer O. An Experimental Application of the Delphi [6]. We will also consider its improved version by Barry W. Boehm and take examples from his book Software Engineering Economics [2].

In the second chapter, we will look at how the performance of a team is estimated theoretically through approximation by fractal functions [20] and consider an empirical example of a test team.

The first section of the second chapter focuses on fractal approximation in estimating development tasks. We will look at how a sequence of Fibonacci numbers is used to estimate tasks by approximating the resources needed for development as a function of task complexity, representing a relative score, and break down the advantage of such an estimate. We will explore how Fibonacci numbers are used in different domains and identify the value of using such numbers in assessing development tasks. We will look at works such as "Promoting convergence: The Phi spiral in the abduction of mouse corneal behaviors" [19] to understand how Fibonacci numbers are used in architecture. We turn to the work of Viswanath D. "Random Fibonacci sequences and the number" [25] and explore how Fibonacci numbers are used in the generation of pseudorandom numbers. Consider how Fibonacci numbers are used in coding theory referring to the work of Knuth D.E. - "Fibonacci multiplication" [16]. Also, we will consider the actual and popular use of Fibonacci numbers for searching of the golden ratio in charts of stock exchange quotations and consider the work of the ideologist of this direction Elliott R.N. "Nature's law - the secret of the universe" [8] and we will analyze the main principles proceeding from the work of Frost A.J., Prechter R.R. "Elliott Wave Principle: Key To Market Behavior" [10]. At the end of this section, we will show the main advantages of using Fibonacci numbers in software development on the basis of those works which we have studied in this section.

The second section of the second chapter will present analytics on the performance of the sample team (all analytics will be presented from December 2019 to January 2021). The analytics in the form of the Burndown report are based on George Dinwiddie's "Feel The Burn, Getting the Most out of Burn Charts - Better Software" [7] and Gantt charts [11]. An analytical control chart is constructed through moving average [13], cluster analysis [1], standard deviation [3], and mean

calculation. An analytical performance chart called Velocity chart with the exponential trend and an analytical total flow chart will also be presented. In this section, we will trace how the exponential scale is formed using a real-world example. We investigate the functioning of task estimation in Fibonacci numbers within the framework of C. Shannon's "Mathematical theory of communication" [22,23] and turn to the example of entropy [21] in task estimation, showing this experimentally. Applying C. Shannon's formula, we will consider how information growth changes using the example of logarithms [2].

The third section of the second chapter will be devoted to the method of avoiding outliers in predicting specific story points. At the moment, development teams find the ideal estimate in Fibonacci numbers empirically, that is, they adjust the information curve (which is close to linear) by experience. The goal of our study automates this process.

In the third chapter of our research, we describe the process of IT product development. We will present the rationale for the chosen program development methods, present the technological base, and the program that will evaluate the development tasks based on our input data. The result of program development will be the correct placement of weights between the layers of neurons, which will help to most accurately assess the development tasks. The database for training the program is real, collected during the experiment, and is suitable for senior-level development teams, while it can learn new tasks and evaluations, reorienting to specific teams.

The programming language for our program is Python. The program is designed based on 743 software development tasks. Our program presents two ways to build it - a string as a list of indexes of letters in the alphabet, and a much more interesting way - a string as a list of unique word hashes. Both methods work, but the second solution seems to be more interesting and accurate.

In our program we use 2 libraries - Keras [31], NumPy [36]. Keras is an open neural network library written in Python. It is aimed at operational work with deep learning networks, is designed to be compact, modular, and extensible. It was created

as part of the research efforts of the ONEIROS project, and its main author and maintainer is F. Schollet, a Google engineer. This library contains numerous implementations of commonly used building blocks of neural networks, such as layers, target, and transfer functions, optimizers, and many tools to simplify working with images and text [31]. The second library is NumPy, an open-source library for the Python programming language. In early 2005, programmer Travis Oliphant wanted to unite the community around one project and created the NumPy library to replace the Numeric and NumArray libraries. NumPy was originally part of the SciPy library. The source code of NumPy is in the public domain [36].

For our program, we use two modules - CSV (to read our data set) and OS (to work with the operating system).

In artificial neural networks, the activation function of a neuron determines the output signal, which is determined by an input signal or a set of input signals. We use 2 activation functions - the input Relu and on the output we use Softmax. Relu (Rectified Linear Unit) is the most commonly used activation function in deep learning. This function returns 0 if it accepts a negative argument, while in the case of a positive argument, the function returns the number itself. The ReLU function has several advantages over sigmoid and hyperbolic tangent:

- It is very quick and easy to calculate the derivative.
- It is 0 for negative values and 1 for positive values.
- The sparsity of activation.

The Softmax function is used in machine learning for classification problems when the number of possible classes is more than two (a logistic function is used for two classes). Softmax is used for the last layer of deep neural networks for classification tasks [30].

To train our program, based on the use of activation functions above, crossover entropy (or otherwise, category cross-entropy) is used as a loss function. In information theory, cross-entropy between two probability distributions measures the average number of bits needed to recognize an event from a set of possibilities if the

coding scheme used is based on a given probability distribution, instead of the "true" distribution [22,23].

To compile our model, we use not only cross-entropy but also Adam gradient descent. Adam (Adaptive Moment Estimation) is an update of the RMSProp optimizer. This optimization algorithm uses moving averages for both gradients and second moments of gradients. It can be regarded as a stochastic approximation of gradient descent optimization because it replaces the real gradient (computed from the full dataset with its estimate (computed from a randomly chosen subset of the data) [32].

In the third chapter, we present the code of our program with comments on the process of its creation and this chapter shows that the goal of our research has been achieved and the objectives have been achieved.

It should be noted that the topic of our research is not investigated in the scientific and practical environment and has great potential in the scientific community.

Chapter I. Tasks assessment in software development

Software development is always subject to many risks, and the main is the inability to schedule the work with exact deadlines and estimates of the work to be done. In the first chapter of our study, we will dissect the theoretical basis for estimating tasks in Fibonacci numbers. In order to begin the analysis of estimations of development tasks, it is necessary to understand why tasks in the SCRUM framework are estimated in Fibonacci numbers and what is the context of such estimations.

Cynefin framework systems

In 1999, D. Snowden wrote in great detail about the software development process in his work "A Leader's Framework for Decision Making" [24], where he introduced the Cynefin framework. In his framework, D. Snowden describes 4 possible systems of work. The basic notion of the Cynefin framework is that all activities and processes correspond to a particular type of system. By system is meant a kind of coherence - doing *x* and getting results - *y*.

The first system is ordered, simple, obvious. In this system an action *x* is done and a result is linearly obtained - *y*. This system is characterized by taking input data, classifying it and reacting in an optimal way. Everything for which there are precise and simple instructions works in this system (for example, to assemble a table according to the instructions). Best practices work in this system.

The second system - ordered, complicated. Similarly as in the first system - an action is taken *x* and we get a result *y*, but with the only difference - now between the action and the result there is a process which has analogs, i.e. there is no single option for *x* of getting *y*. This system is characterized by taking input data, analyzing it and reacting to that data according to an expert point of view or analysis result. Everything that can be analyzed works in this system (for example, to determine the appropriate trajectory for a spacecraft). Good practices work in this system.

The third system is complex. In this system *x* and *y* connected, if we do *x*, we get *y*, which in turn will affect *x*, and so on. It is a system of complex non-linear systems - learning configurations, the interaction of a large number of agents. Such configurations can only be explained in hindsight - they can be explained, but almost impossible to predict. Configurations can recur, but will not be patterned - they can disappear or change at any moment. This domain is characterized by testing or probing (to detect configurations), observation, experimentation. Anything where the consequences are ambiguous, that can be explained but not foreseen - works in a complex system. It is development by agile methodologies (and specifically with the SCRUM framework) that works in a complex system and it is important for us to consider this to create our IT product.

The fourth system is chaotic. For this system, communication is difficult (or impossible) to define. All the processes are changing rapidly and unpredictably, the result is impossible to predict. This is a system of crisis and innovation. It is impossible to classify this system into any categories, and it makes no sense to wait for certain configurations. It is characterized by quick actions against uncertainty, observation of reaction and correction of actions. An example of working in such a system is realignment.

This classification gives us an understanding of how work within the SCRUM framework - working in a complex system, development teams have an end goal, but best and good practices will not help to form and evaluate development tasks. It is the work in a complex system that needs scientific understanding, because without it the risks of disruption of deadlines, poor planning, inability to predict and evaluate work affect performance metrics and work results.

Basic concepts of SCRUM framework

97% of companies around the world [27] that develop software or interact with software practice agile development and 54% [27] of them use the SCRUM framework. In order to start creating a task evaluation algorithm, we need to

investigate how such evaluation is used in practice. Let's break down the basic concepts of the SCRUM framework and how it works.

The SCRUM methodology is based on several pillars, one of them is the method of facilitation. The basis for such methods is the facilitation model created by the International Institute for Managerial Development (Lausanne, Switzerland) together with The Lego Group corporation - Lego Serious Play [17][18]. IMD professors Johan Ross and Bart Victor and the founder of the Lego Group Kjeld Kirk Kristiansen took part in its development. Now this method can be used under a Creative Commons license. This method is based on solving business problems using Lego cubes and figures. Thanks to this method, people find unconventional and creative solutions.

Another important component of the Scrum framework is the tactics and strategies of rugby or short distance running. Sometimes the transcription of the word SCRUM is Sprint Continuous Rugby Unified Methodology [18]. The authors of the SCRUM methodology, by analogy with the game of rugby, the authors of the methodology suggest that all team members "engage in a scrum, act in concert and bring the game to the desired result" [18].

Two prerequisites for the SCRUM framework are a team and mandatory activities. The following roles are defined in the SCRUM team:

- Product owner - maximizes product value (representing the user side).
- SCRUM master - oversees all mandatory activities, conducts and facilitates meetings.
- SCRUM team - a cross-functional team of 5 to 9 developers of different profiles, which develops the ordered product.

Obligatory events for software development by SCRUM method:

- Sprint planning meeting [35] - work planning for a limited period of time (from a week to four weeks).

- Poker planning [14] (a fundamental activity for our research work) - a task evaluation technique that is based on common agreements.
- Daily scrum - a daily meeting on plans and results.
- Sprint review meeting - discussion of sprint results.
- Retrospective - discussion of sprint results, identifying what can be improved, what to get rid of.

It is very important for our study to understand the principles of poker planning. It is during it that developers evaluate tasks with Fibonacci numbers. Usually poker planning goes as follows - the development task is considered, all team members secretly vote, then they look at the voting results and discuss the best estimate. The next section deals with the basics of poker planning.

Delphi method in poker planning

The base of poker planning is laid in the method of predicting the impact of future scientific developments on methods of warfare, which is known as Delphi and developed by the RAND Corporation [28]. This method is of interest to us in the vector of quantitative research [6] - it will help us to form our own algorithm for calculating an "exact estimate".

The Delphi method is a structured communication method originally developed as a systematic interactive prediction method based on a group of experts. The main idea of the Delphi method is the analysis and subsequent processing of experts' evaluations of the subject of evaluation, and as a result, an overall valid and reliable evaluation. The Delphi method formula is as follows [12]:

$$\bar{x}_j = \frac{\sum_{i=1}^{m} x_{ij} \cdot k_i}{\sum_{i=1}^{m} k_i}$$

- X_{ij}— the relative importance score given by the i expert to the j element;
- K_i — coefficient of competence of the i expert, taking into account the experience in interaction with a particular element (K_3) and the reasonableness of the answer : (K_a): $\frac{K_i = K_3 + K_a}{2}$;
- $i = 1...m$— expert numbers;
- m — number of experts;
- $j = 1...n$ — numbers of the elements to be studied;
- n — number of items.

Using a series of sequential actions, maximum consensus appears in determining the correct solution. The Delphi method analysis is conducted in several steps, and the results are processed using statistical methods. The basic principle of the Dolphin method is that independent experts (who may not even know about each other) give more accurate estimates and better predict the outcome than a structured group of people or team.

The average value of the answer varies between 1 and 100 points. Than higher $\bar{x}_j$, than higher the priority of the item. The reliability of the result is determined by the dispersion of expert evaluations. The smaller the dispersion of estimates, the more consistent the answers and the more reliable the result. It is believed that during this process the range of answers will decrease and the group will strive for the "right" answer. Finally, the process stops after a predetermined stopping criterion (e.g., number of rounds, consensus achievement, and stability of results) and the average or mean scores of the final rounds determine the results.

This method was reinterpreted by Barry Boehm and John Farquhar in the Wideband Delphi Technique [2]. The name wideaband was given to this method because of the wider and more active interaction between the participants. This one was popularized in the book Software Engineering Economics by Barry W. Boehm

[2] and it is similar to the SCRUM poker planning - same to Delphi method there is a mandatory meeting and a facilitator. The process of using Wideband Delphi is similar to Delphi, but broader. When planning a Wideband Delphi session, the problem is defined and the participants selected. The kickoff meeting gets all estimators focused on the problem. Each participant then individually prepares initial task lists and estimates. During the estimation meeting, several cycles lead to a more comprehensive task list and a revised set of estimates. The information is then consolidated offline, and the team reviews the estimation results. When the exit criteria are satisfied, the session is completed.

Wideband Delphi methodology is based on several reliable evaluation principles. The principle of teamwork makes it possible to take into account the opinions of several experts at once. The resulting range of estimates reflects the share of uncertainty, which is inherent in any estimate. Wideband Delphi's methodology takes time and requires the presence of experienced experts, but allows you to make an objective estimate and avoid extreme values.

These methods are the basis of poker planning, but with the difference that these methods do not use Fibonacci numbers as they do in the SCRUM framework task estimation. In creating our own evaluation model we will take into account the basics of poker planning in Delphi methods, but we need to take into account that in SCRUM framework the evaluation of development tasks is done in Fibonacci numbers through fractal approximation of data and additional data analytics are needed, which is presented in the second chapter of our study.

Chapter II. Determination of future team performance

Fractal approximation for tasks assessment

Fibonacci numbers are widely used to solve various problems in many different areas. A series of Fibonacci numbers tends to a proportional ratio and is a number with an unpredictable and infinite sequence of decimal numbers in the fractional part. Here are some examples of how Fibonacci numbers are used to help us understand how they are used in development.

The Fibonacci sequence [25] is a sequence of numbers where each successive number is an addition of the last two numbers, starting with 0 and 1:

$F_0 = 0$, $F_1 = 1$ and $F_n = F_{n-1} + F_{n-2}$ for $n > 1$.

If we express it in numbers, we get a series of 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 …

The main feature of this series is the ratio of the row member to the previous one, which tends to the number 1.618, which is the golden ratio. This number has been known since ancient times and was first found in Euclid's Elements (about 300 BC), where it was used to construct a regular pentagon [25].

Based on the Fibonacci sequence, we can construct a set of squares with sides equal to the elements of this sequence. Adding each square from this set to the sides of the previous two squares, we will always get a rectangle whose sides are equal to the next two Fibonacci numbers. If we decide to inscribe a quarter circle in each of these squares, we get an approximation of the well-known golden spiral used in architecture [25].

As we know, the golden ratio is an irrational number approximately equal to 1.618. If we divide each subsequent Fibonacci number by the previous one starting from one (1/1 = 1; 2/1 = 2; 3/2 = 1.5; 5/3 ≈ 1.666; 8/5 = 1.6; 13/8 = 1.625), we will get closer to the real value of the golden ratio.

One of the most unusual examples of the use of Fibonacci numbers in modern mathematics and computer science is the generation of pseudorandom numbers. For

researchers in all fields of science, the question of random numbers has recently become very important. A random Fibonacci sequence is a stochastic analogue of a Fibonacci sequence, which is defined by a recurrence formula: $f_n = f_{n-1} \pm f_{n-2}$, where + or - is chosen randomly for each n with equal probability.

According to the theorem of Harry Kesten and Hillel Furstenberg, random recurrence sequences of this kind grow in a certain geometric progression, but it is difficult to calculate their growth rate. In 1999, Diwakar Visvanath showed that the growth rate of a random Fibonacci sequence is equal to 1.1319882487943..., a mathematical constant that was later called the Visvanath constant [25].

These numbers turned out to be most important for specialists in numerical modeling and optimization - it was their experiments that required huge arrays of random numbers in the first place. An indirect example of the importance and necessity of these numbers is the very popular in the 20th century book "A Million Random Digits with 100,000 Normal Deviates" by the American analytical corporation RAND [28], which was published for half a century. Its main content was a million random numbers written with 2,500 numbers per page.

A computer cannot generate random numbers, so an algorithm was created which generates a sequence of numbers whose elements are almost independent of each other and obey a given distribution. The numbers generated by this algorithm are called pseudorandom numbers. Modern computer science widely uses pseudorandom numbers in a variety of applications. In this case, the quality of the used generators directly depends on the quality of the results. This fact is underscored by the famous aphorism of ORNL mathematician Robert Caveat: "the generation of random numbers is too important to leave it to the chance" [28].

Also interesting is the use of Fibonacci numbers in coding theory, the science of the properties of codes and their fitness for a given purpose. Coding within the theory is seen as the process of converting data from a form suitable for direct use into a form suitable for transmission, storage, automatic processing, and preservation against unauthorized access. The main problems of the theory include questions of

mutual uniqueness of coding and complexity of realization of a communication channel under the given conditions. The coding theory proposes stable so-called "Fibonacci codes", where the base of the codes is an irrational number. Such codes are called Fibonacci Numeral System, which is a mixed number system for integers based on Fibonacci numbers. It is based on the Zeckendorf theorem - any non-negative integer is uniquely representable as a sum of some set of pairwise different Fibonacci numbers with indices greater than one, containing no pairs of neighboring Fibonacci numbers [34].

Another use of Fibonacci numbers is to find the golden ratio in stock price charts. Such idea belongs to Ralph Nelson Elliott. He has studied the annual, monthly, weekly, daily, hourly, and half-hourly charts of various stock indices, covering a 75-year history of market behavior. In the course of his research, he noticed that the index movements are subordinated to certain rhythms - waves, in the proportions of which the number 1.618 can be seen. R. Elliott wrote a number of works on this topic, the largest of which was the book Nature's Law - The Secret of the Universe, in which he included all his works on wave theory and the Fibonacci ratio. After Elliott, many traders and market researchers have searched for various applications of Fibonacci numbers in stock trading. The development of computing technology has allowed analysts to go far in this direction [8].

Fibonacci levels are often used in conjunction with Elliot's wave theory. According to this theory, any trend movement in a financial instrument can be decomposed into five waves: three main (impulse) waves along the trend and two correctional waves against the trend. The impulse waves are numbered the first, third, and fifth, while the corrective waves, in turn, are numbered the second and fourth. Any corrective movement can also be decomposed, but only into three waves. All of the internal waves are also decomposed according to the fractal principle. Understanding which wave the price is currently forming makes it possible to predict where it will go next. The third wave is the most interesting for traders. It is

considered the longest and fastest wave. The ideal deal using Elliot's theory is to enter a trade at the end of the second wave and exit at the end of the third wave. [10]

According to the theory, the height of the 3rd wave refers to the 1st wave as 1.618. So, if we see the 1st and 2nd waves already formed, we can calculate the length of the 3rd wave using the Fibonacci levels. The "Fibonacci expansion" tool is specially designed for this purpose in some terminals. It is built based on three points: the beginning of the first wave, the end of the first wave, and the end of the second wave. The Fibonacci levels will appear on the screen, and the level marked 1.618 will mark the estimated end of the third wave.

And finally, let's break down the use of Fibonacci numbers in software development. A sequence of Fibonacci numbers is used in evaluating development tasks by approximating the resources needed for development depending on the complexity of the task, representing a story point - a relative score. Story points are a mapping from a set of task complexities to a set of required resources. Usually the tasks are scored in a sequence from 1 to 13 (from less time and volume to more), and groups of tasks may be scored in 21 or more story points. Usually sprints take tasks from 1 to 8 story points, and if the score is higher, you should consider splitting the task into several tasks.

What is the advantage of using a sequence of Fibonacci numbers in development?

- All scores are given exponentially, and the harder the task, the larger the gap between adjacent scores. This helps detail small tasks and leaves a margin for more difficult tasks. That is, small and clear tasks are usually graded 1, 2, or 3 story points. The more abstract the task, the higher its score (the more uncertainty).
- Whole numbers specify the value.
- Developers have to make a choice "more or less».
- Scheduling tasks is easier - non-linear sequencing reduces analysis time. Allocating tasks can be faster with a 5 and 8 interval than with a 3 4 5 6 7 scale. The more detailed the scale - the more time is spent evaluating and planning.

- There is no way to reduce the task proportionally (e.g., if working in pairs).

The Fibonacci scale is progressive because the values grow faster than linearly. Such a scale can be called exponential. We conducted an experiment on a test team, and made interesting observations - the usefulness of such an exponential scale is first of all that the larger the size of an item of unperformed work, such as *N* points of a particular backlog task, the harder it is to identify the difference between *N* and *N-1*. This is a consequence, but it is not an axiom. Let us assume that an arbitrary task of the backlog (the list of tasks for software development) *X* is not larger than the maximum possible size of the task from the backlog *Y* and yet can be any value from 0 to *Y* with equal odds. Let us also assume that we have an estimation method that allows us to estimate the size of *X* with absolute precision *P*. As we know from the basics of information theory [22], the information of two experiments closed in experiment *A* about to experiment *B* can be expressed as follows:

$I(A,B) = H(B) - H_a(B)$ where $H(B)$ is entropy [23], that is, the level of uncertainty of the experiment *B*; $HA(B)$ - conditional entropy *B* after the experiment *A* took place. We see how clearly entropy from information theory is similar to our software design. Relying on the formula above, we can interpret the amount of information contained in an experiment *A* as the amount of uncertainty it reduces relative to the experiment *B*.

Applying this to the evaluation of development tasks, we can determine that we have two experiments - the first (*B*) is to determine the exact size of the development task (*X*) and the second (*A*) is to apply our methodology to the evaluation of development tasks, that is, to reduce uncertainty. Applying the Shannon formula we get the following formula:

$I(A,B) = log(Y/P)$, where the logarithm is unimportant, it can be any number greater than 1, which is based on a simple property of logarithms, for example [2], logarithm $log_b x$ can be calculated from the logarithms of *x* and *b* with respect to an arbitrary basis *k*, using the following formula: $log_b x = \frac{log_k x}{log_k b}$. Usually

the number 2 is used for the base of the logarithm, since we store information in binary format.

Such an equation makes an interesting observation - the information we get from an estimate grows much slower than the accuracy of the estimate. It grows as fast as the logarithm function. That is, because of this observation, it becomes obvious to us why a low estimate of a problem helps and a high estimate the opposite. It is this approach that makes the use of an exponential (or near-exponential) scale of problem estimation perfectly logical - it adds valuable information faster, any function grows faster, and tends to a perfect linear function where $f(x) = x$.

Burndown curve and current process data

The test team performance analytics are the empirical inputs for our research, without which our research would have no practical relevance. Data collection for the empirical research began in December 2019 and was completed in February 2021.

The composition of the SCRUM test team was 7 people: 1 Quality Assurance Senior level engineer, 1 Quality Assurance Middle level engineer, 2 Frontend Senior level developers and 2 Backend Senior level developers. The test team in this composition started in December 2019, and it is particularly interesting how the task scores regulates as time passed. The task evaluation analysis is based on 29 two-week sprints.

The task manager Jira is taken to maintain the tasks and the reports below are based on it. Task evaluation from the reports is by prior agreement and does not rely on scientific data, which is particularly valuable for our study.

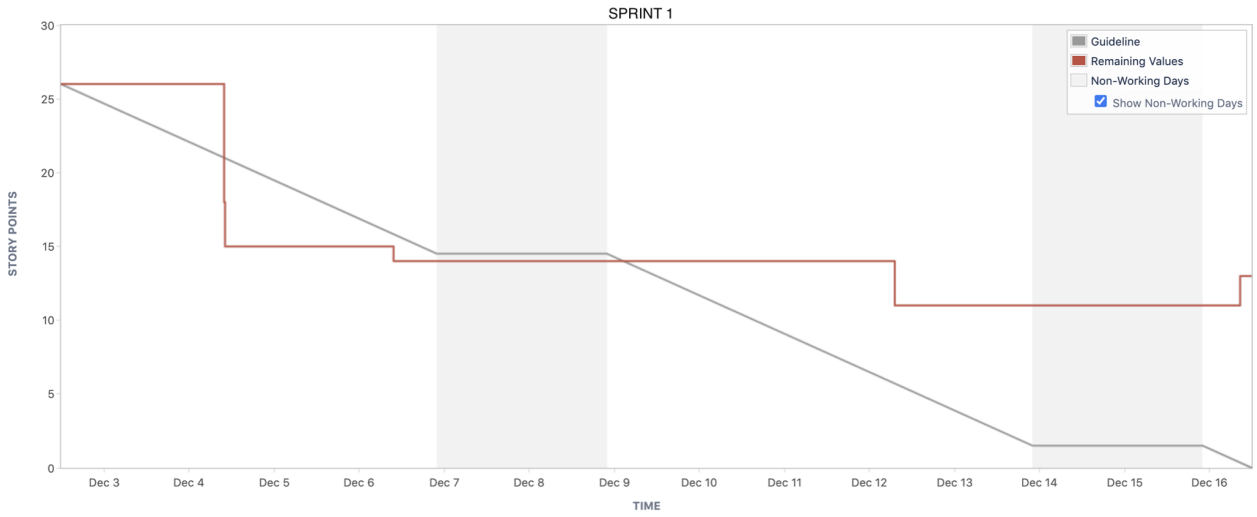
One of the most visible reports that can be presented in our study is the Burndown chart [7]. This chart is the primary means of keeping track of tasks completed in a sprint or in the entire project. Our team's Burndown charts for 29 sprints are provided in the Appendix to our study [Attachment 3].

Let's look at the main components of this chart:

- Guideline - the burndown chart marks the ideal task line to rely on.

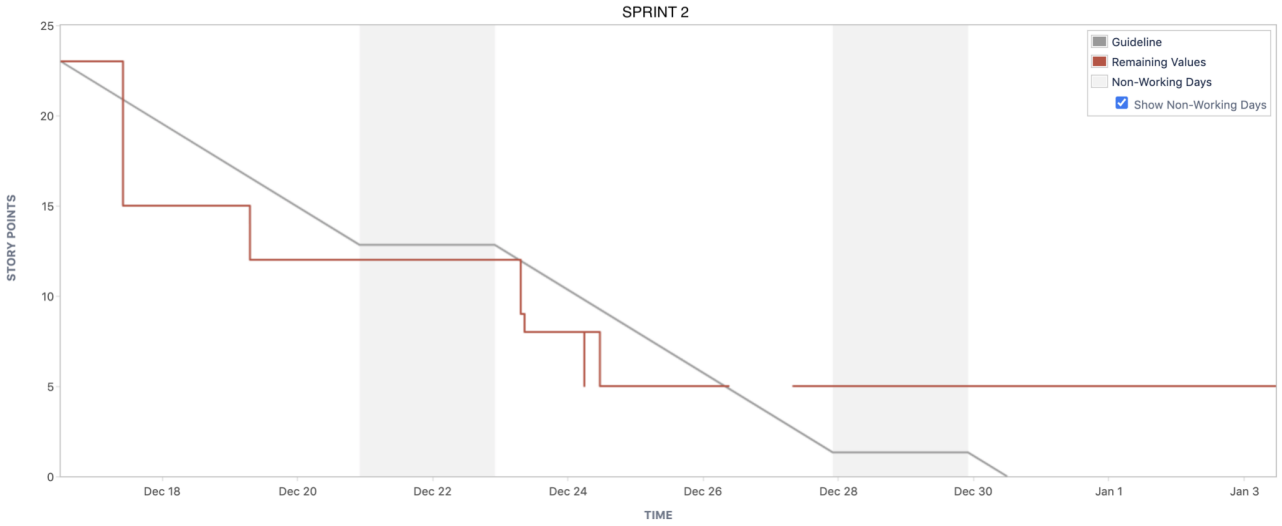
- Remaining Values - the actual history of task completion.
- The Y scale marks the number of planned story points.
- The X scale marks the number of days until the end of the sprint.

Let's start with the first, introductory sprint:

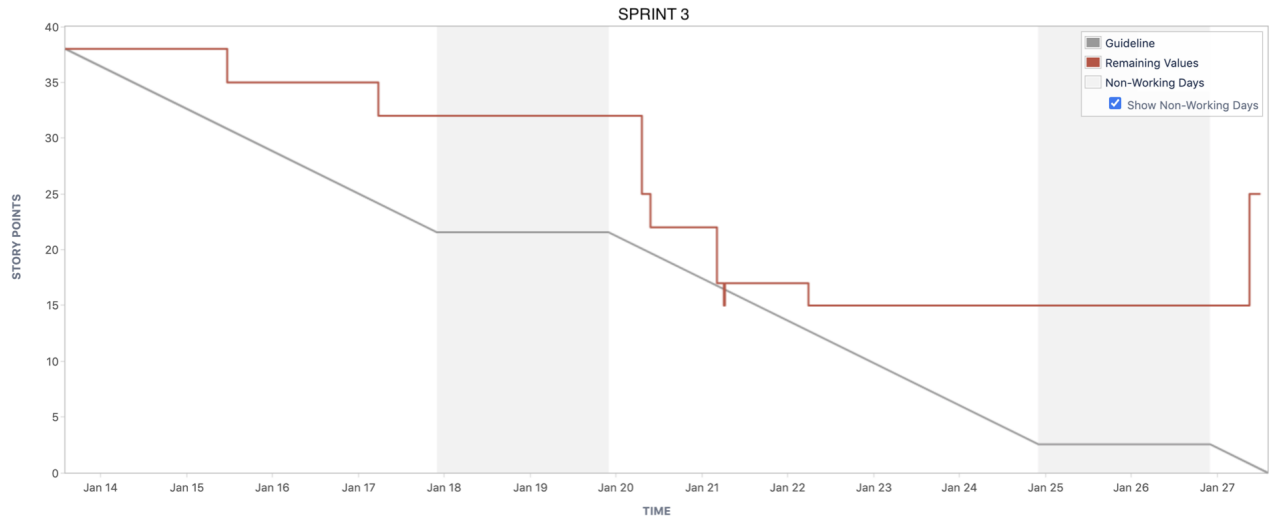


We see that the team evaluated the task scope - 26 story points, followed by a change in scope change to 28 story points. 15 story points are completed and 13 story points are not completed. From the schedule, we can see that the team has not completed all the planned tasks on time, while new tasks have been added, or existing tasks have been re-evaluated. That is, the task assessment was unsuccessful, many tasks were rated too low, and too many tasks were taken into the sprint as a whole, or too many tasks were taken into the sprint.

In the second sprint the scope was smaller (23 story points), and the first part of the sprint went about the same or even lower - the tasks were solved faster than estimated:

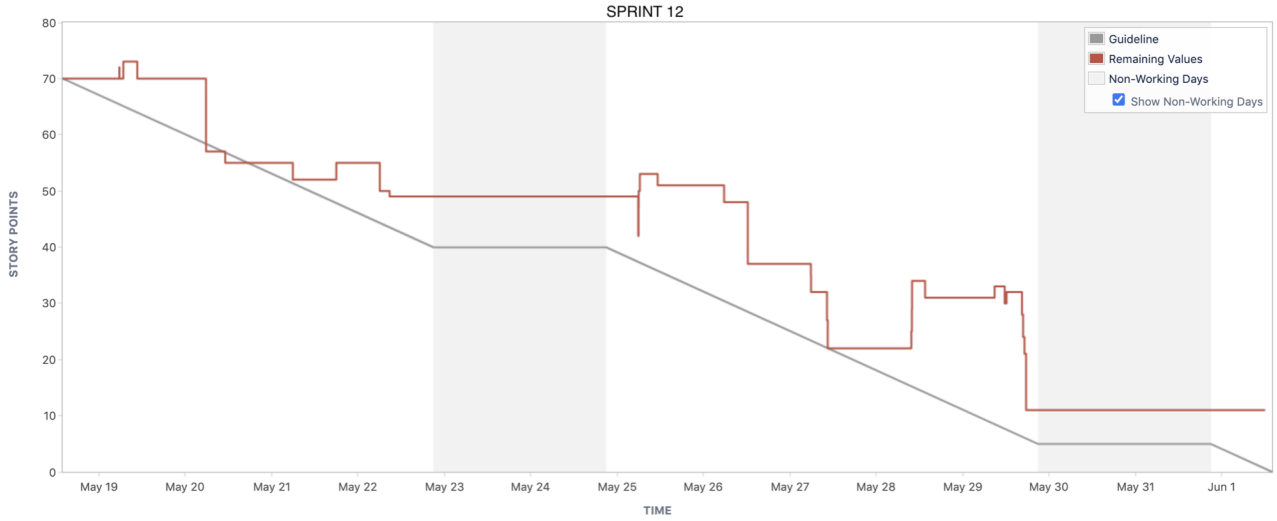


In Sprint 3, we again see changes in the scores, the removal of tasks from the sprint, and as a result, there are uncompleted tasks in the sprint. Once again, the incorrect task scores, which did not lead to the burning of these tasks.



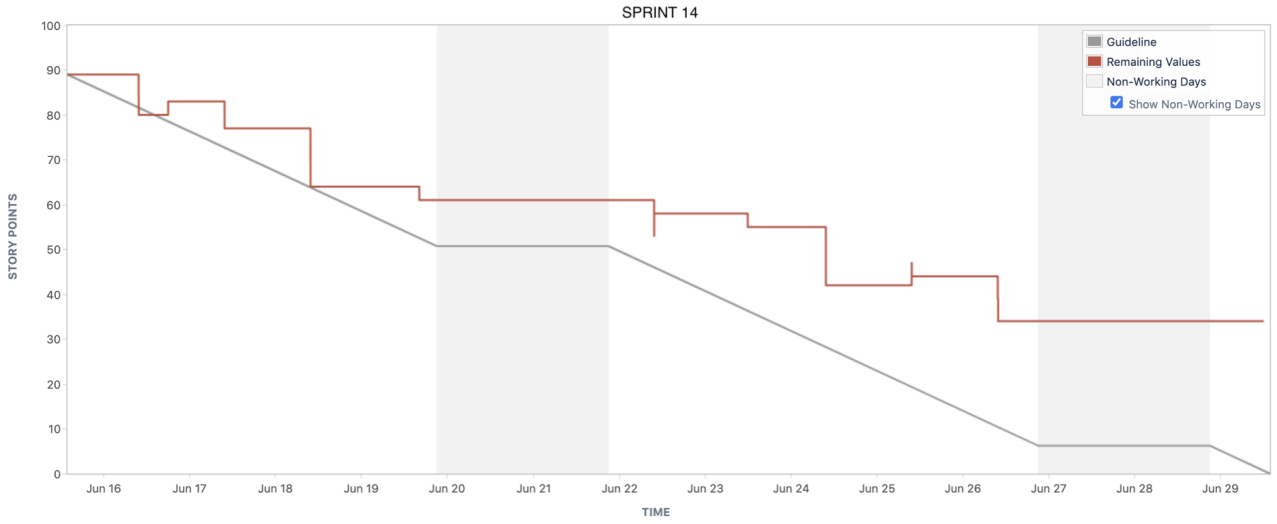
This state of affairs lasted for half a year, while the evaluation of tasks was corresponded to good result. For half a year, the graphs reflected the problems of not knowing how to evaluate tasks and what scope of tasks a team can take in a sprint. In all of the graphs from Sprint 1 through Sprint 12 in Appendix [Attachment - 3], we see that there are a lot of unfulfilled tasks, and some tasks take a very long time to develop. This trend persists for 11 sprints in a row, and only after that we see more accurate estimation of tasks.

On Sprint chart 12, we see that Remaining Values are very close to Guideline in spite of overestimation. A trend is emerging - story points are being burned evenly and consistently. This means that the team begins to understand if they're estimating correctly, and they come up with how many story points they can take into the sprint to achieve the ideal burndown.

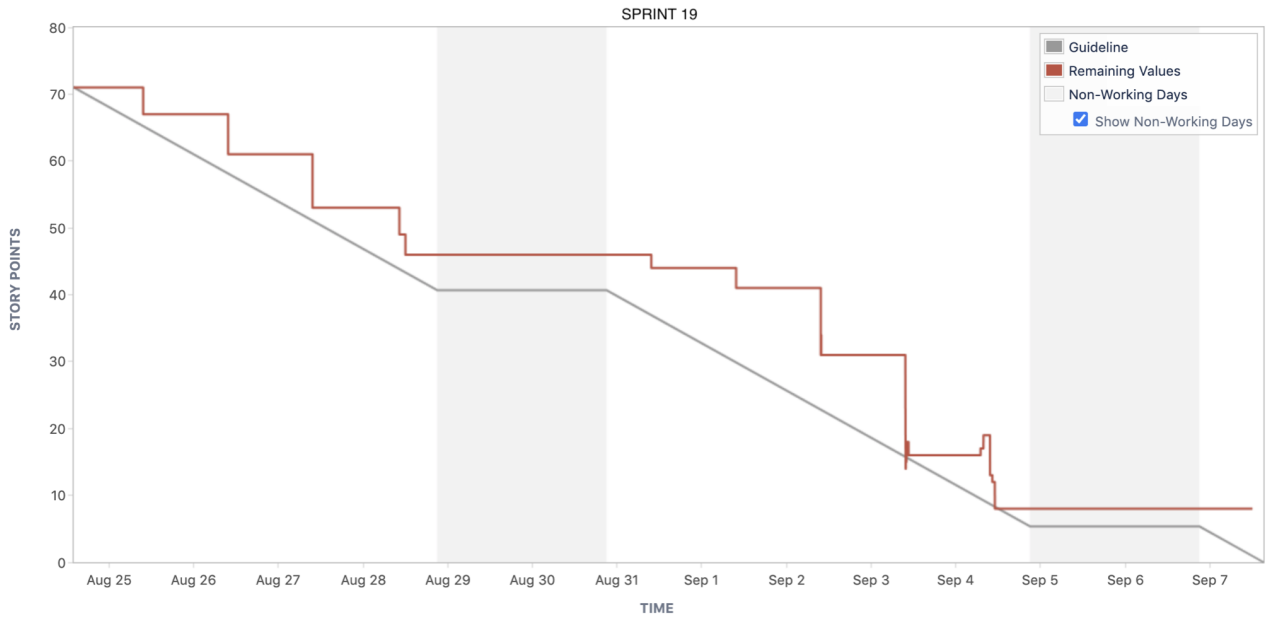


Despite the large scoop of tasks (the team started the sprint with a total scoop of 70 story points) - the tasks performed smoothly, even with new tasks thrown in, and at the end of the sprint almost all story points were burned.

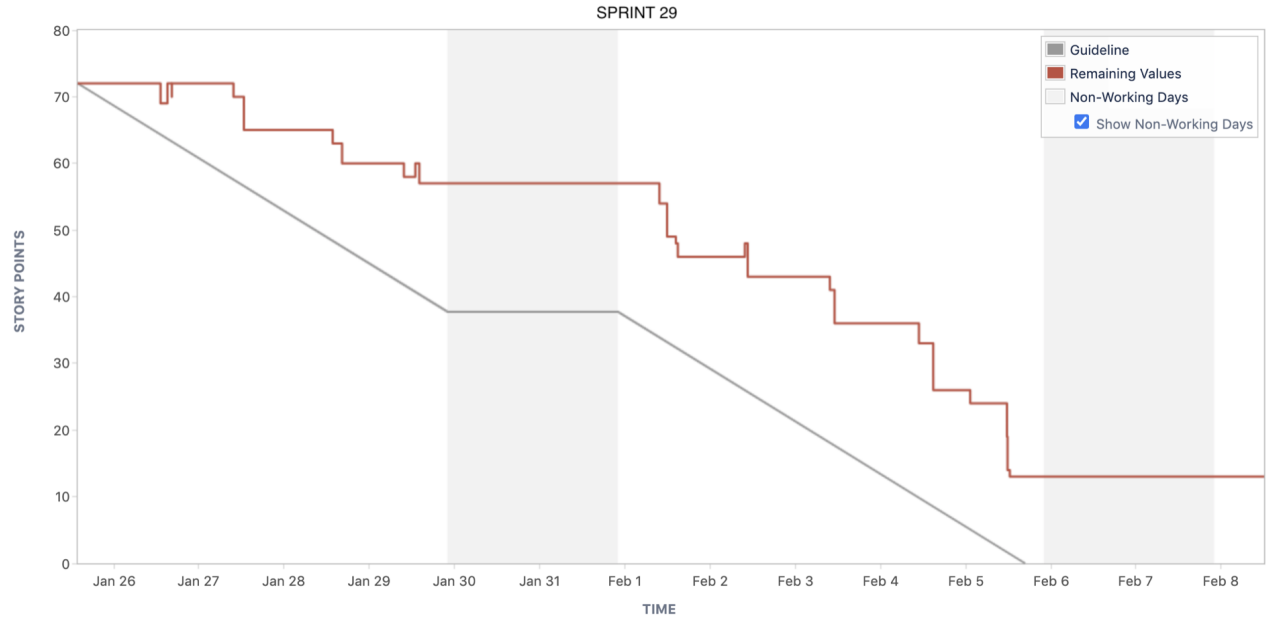
Let's look at some graphs from the sprints that show how the software development task scores have reflected good results. Sprint 14 took quite a few story points - 89. Nevertheless, we see a fairly even burndown, and close to Guideline.



Sprint 19 shows an almost perfect trend - story points are burned daily, scope change is insignificant, and by the end of the sprint almost all tasks are completed. The graph reflects a very clear and well-coordinated work and a sufficiently high-quality assessment of development tasks.



In Sprint 29, the last sprint of our research, we also see a uniform burning of story points with a scope that is close to the average of all 20 sprints. The team understands how many tasks it can take into the sprint and is fairly clear about the tasks, scope change is minimal.



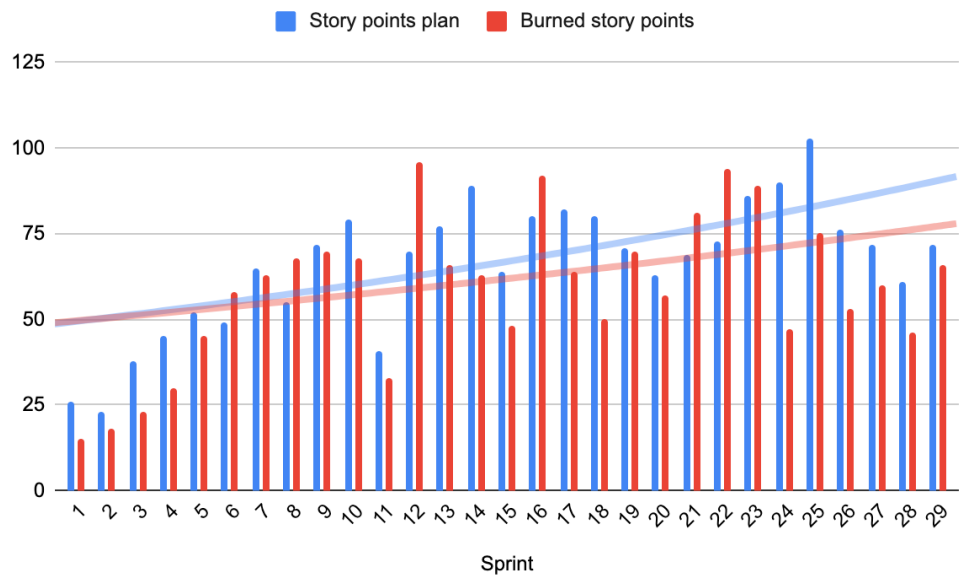
For a more visual study, we made a comparative Table 1, which covers all sprints and shows general trends:

Table 1

Sprint	Story points plan	Story points total	Scope change	Burned story points
1	26	28	2	15
2	23	23	0	18
3	38	48	10	23
4	45	48	3	30
5	52	53	1	45
6	49	70	21	58
7	65	74	9	63
8	55	89	34	68
9	72	101	29	70
10	79	90	11	68
11	41	51	10	33
12	70	107	37	96
13	77	97	20	66
14	89	97	8	63
15	64	89	25	48
16	80	128	48	92
17	82	82	0	64
18	80	85	5	50
19	71	79	8	70
20	63	66	3	57
21	69	86	17	81
22	73	130	57	94
23	86	103	17	89
24	90	92	2	47
25	103	102	-1	75
26	76	92	16	53
27	72	87	15	60
28	61	70	9	46
29	72	79	7	66
Average	**66,3**	**80,8**	**14,5**	**58,8**

We see that the average of 29 sprints burns 58.8 story points against the stated 66.3 story points, an 89% burn rate, which is a good result. The average sprint scope changes by 14.5 story points, and this is an opportunity for the development team to improve its score. The total number of story points the team gets when completing the sprint is 19% greater than the stated scope, which is a poor result and would be an excellent growth area. In Chart 1 (which at the same time is the team's Velocity chart), we see a slow growth exponent, which means that the team's productivity is increasing while the number of tasks the team plans is also increasing. It is based on how Velocity changes that we can draw conclusions about team performance.

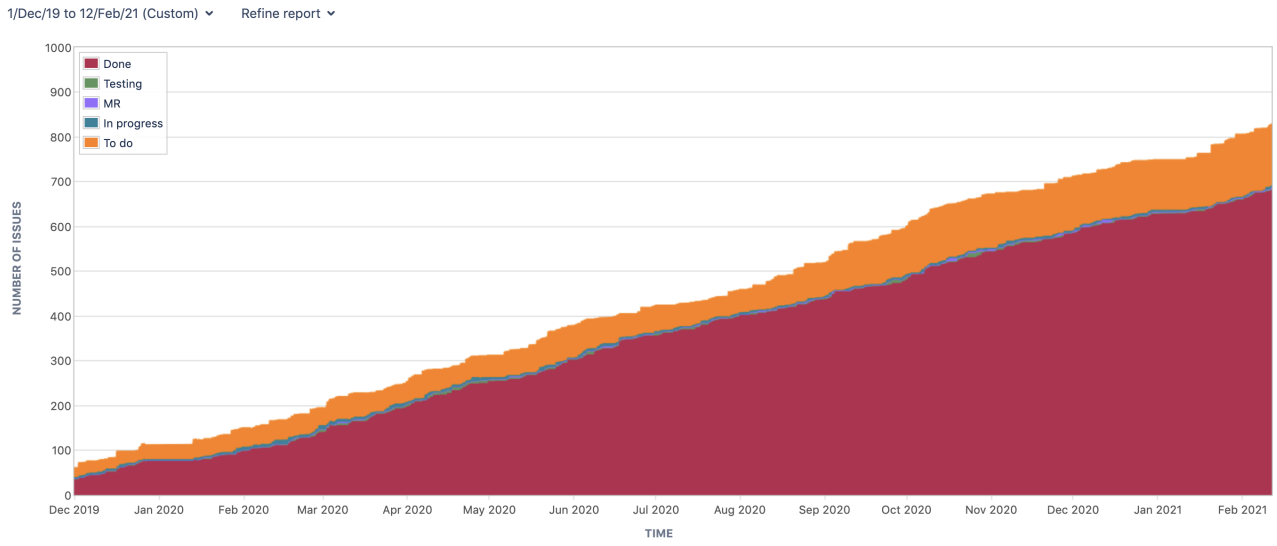
Chart 1



Analyzing the information from the Burndown charts, we see that the test development team has better tasks estimation, and continues to improve their score over the course of their collaboration, and the improvement will occur with each successive sprint.

Consider another chart, the Cumulative Flow Diagram (CFD), which shows the progress of the work. The horizontal X-axis in the CFD indicates time and the

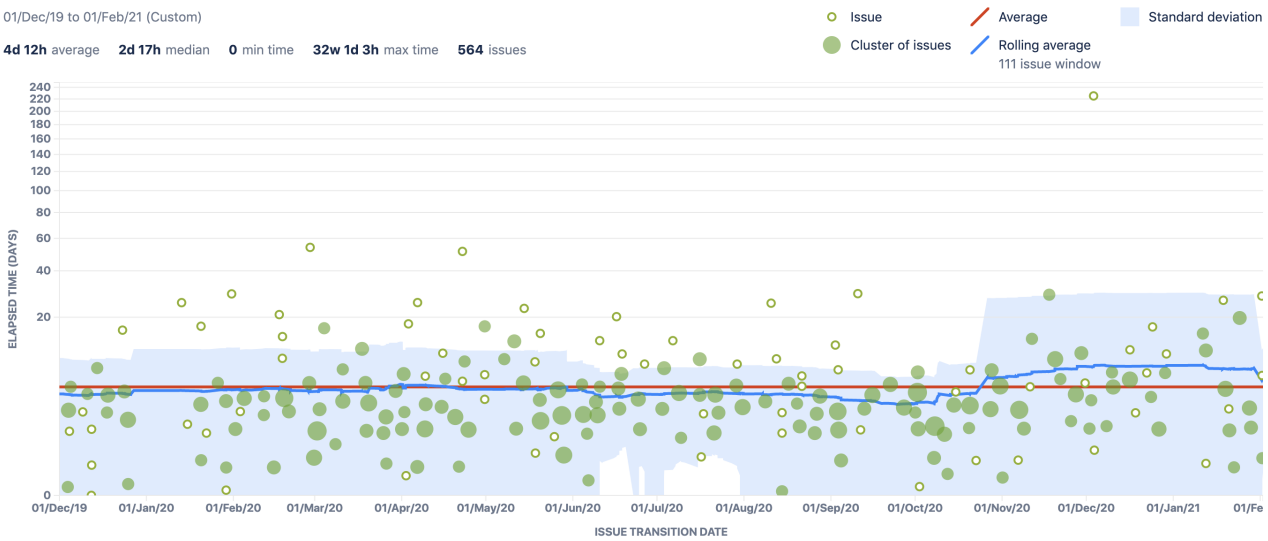
vertical coordinate axis indicates requests. Each colored area of the chart equates to the status of the business process. CFDs are useful for identifying bottlenecks. If the chart contains an area that expands vertically over time, the status of that area will be a bottleneck. This diagram shows the number of tasks in the same status at the same time. Let's look at such a diagram within our test team:



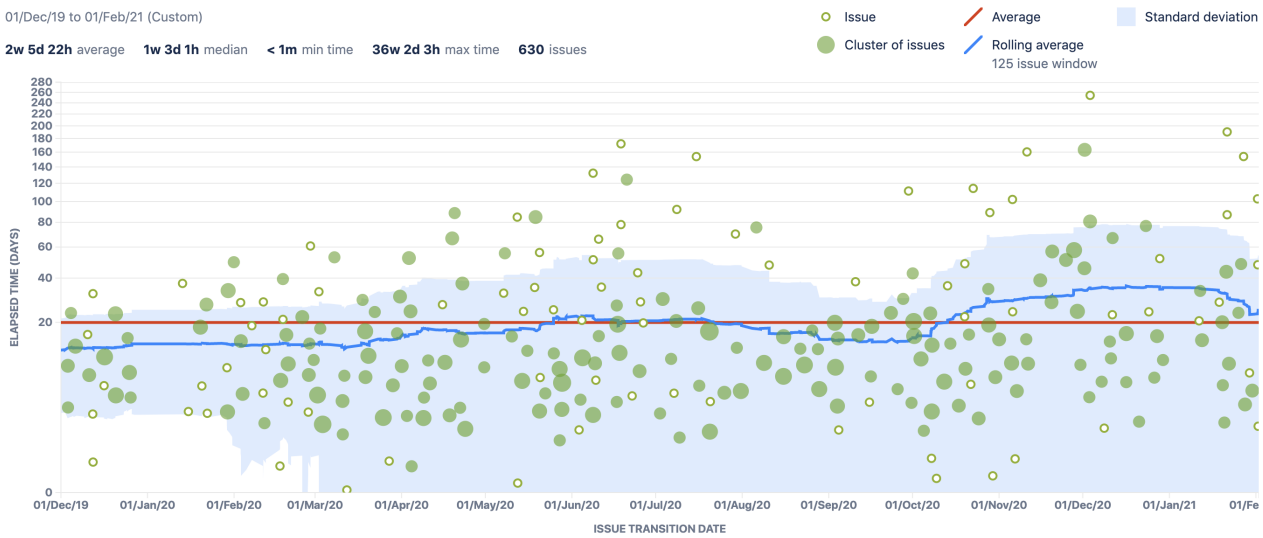
We see that our diagram highlights 5 established processes: to do, in progress, merge request (MR), testing, and done. We see that the to do tasks do not stand out too much against the Done tasks, and the Done grows at the same rate as the scheduled tasks. This means that the team has time to handle the incoming flow of tasks.

Let's consider another diagram - a control chart. Control charts allow us to focus on the cycle time of individual tasks - the time it takes in total to move a task from the "In Process" stage to the "Done" stage.

Small cycle times usually mean high team productivity, and teams characterized by stable cycle times for many tasks are more predictable in terms of delivering work results. Consider such a schedule based on the work of the tested team:



In this control chart, we have added all the stages of task progress as in the cumulative chart. On this chart, we see a very large maximum task time - more than 36 weeks, and most likely this refers to tasks that were often postponed, and it does not give us a complete picture of what is happening. At the same time, we see a fairly large median of more than a week, and an average of more than two weeks. So let's try to remove 2 business processes - planned and done:



By removing the two processes that significantly increase the average or median task completion, we see a pretty excellent graph with real numbers that reflect how much of the "active" task is actually done. We see that the median is 2 days and 17 hours, and the average is 4 days and 12 hours. This is the absolute norm

for developing and completing tasks in two-week sprints. It's also interesting to note the rolling average of this chart - decreasing rolling average indicates process improvements and increased throughput. We can see that our moving average is increasing, and that is a positive result. Thanks to this graph we can predict the work of development teams more accurately, and we can also take into account the median and average data in our program.

Deviation avoiding technique in story points forecasting

Since indeed most of the development teams in the world use a sequence of Fibonacci numbers to evaluate their tasks, can they be guaranteed to say that the correction of their information curve grows fast enough, and more useful than a linear scale? In our study, we have indicated that the logarithmic behavior of such a curve is traceable, but finding its base remains a challenge.

Changing the base of the estimate in the Fibonacci sequence is an unappreciated method. Such a method can bring the team closer or farther away from an effective estimate. There is only one ideal basis for evaluation, and it is important to get close to it. Often teams regulates estimations over time as they work together, that is, they re-scale bases, to the point of the most accurate estimates. Coming to a single Fibonacci score base, development teams typically have 30-60 story points in a two-week sprint. This is a very important observation for our study. Of course, much depends on the size of the team (smaller teams will have fewer story points, more in larger teams, respectively). But it is all about optimizing the basis of evaluation over time to manage the available information.

To make it easier to reflect what we are aiming for, we will replace the Fibonacci sequence with an exponential function.

Indeed, if at some point the teams go for a new task assessment, for example, two old tasks became one new task, then the function $f(x)$ turn into another function $g(t) = f(2x)$, where $t = 2x$. If we replace this expression with x, we get $g(t) = b^t$,

where b is the square root of a $g(t)$will again become an exponential function, but with a different basis.

Thus, in the task estimation prediction framework through the deviation avoiding technique, we understand that the development teams themselves regulates the bases of their estimates over time, and arrive at the optimal estimation capability, that is, they regulate the information about the elements of lag they are estimating. At this point, teams find the exponential function empirically, that is, they adjust the information curve (which is close to linear) by experience. One of our main goals is to automate this process through program development, and we describe this process in chapter three.

Chapter III. Intelligent tool for the decision-making support

Program for tasks estimation in story points

Working with the test team we collected data set [Attachment 1] of 743 development tasks (collected during our empirical research) and evaluations to these tasks. Such a sample is an example and the basis for the construction of our program, for the ideal work of our program about 30-40 thousand development tasks are required. Nevertheless, even on such a relatively small number of tasks we can already say that our hypothesis that task evaluation can be automated by creating a program with a certain logic is confirmed.

Our program is a mathematical model and its software implementation, whose basic elements are neurons. Neurons, in turn, are connected to each other and act in a given way. The value of our program in the evaluation of development tasks is that, unlike a conventional program, the software is able to learn, that is, we can add new tasks, evaluate them and improve the evaluation of such tasks by training the program.

As images for our program we chose text symbols. The main difficulty in building our software was the representation of strings with text symbols in numbers. Our program presents two ways - a string as a list of indexes of letters in the alphabet, and a much more interesting way - a string as a list of unique word hashes. Both methods work, but the second solution seems to be more interesting and accurate.

Our program is written in Python, for several reasons. First, Python is a programming language with low entry threshold - it is not necessary to write code from scratch, you can use already existing libraries. This language is concise and expressive, the time required to use such a language is minimal. It also has a mechanism for interoperability with other languages such as C and C++, which in turn gives access to fast computations. Another advantage of Python is the solution in a small number of lines. Full code is attached in Appendix [Attachment 2].

In our program we use 2 libraries - Keras, NumPy and modules CSV and OS. We started to use Keras not at once, originally we thought we could do without it. But without it our network refused to work with a single layer, which in turn entailed writing a few layers by hand - and this is an extra work and strings, so we decided to use this library. Keras is a high-level neural network API, written in Python and able to work on top of TensorFlow, CNTK or Theano. Such a library helps build fast experiments and allows you to go to the result almost without delay, which is a huge advantage in any research work. With such a library, you can quickly prototype, maintain repetitive networks, and work on the CPU and GPU [31].

The second library we use is NumPy. This library is open source and has great features - it supports multidimensional arrays, matrices, and it also supports high-level mathematical functions. The NumPy library provides implementations of computational algorithms optimized for working with multidimensional arrays. As a result, any algorithm that can be expressed as a sequence of array operations and implemented using NumPy runs as fast as equivalent code executed in METLAB [36].

The CSV (comma-separated-value) module we use in creating the program is a tabular data representation format. It is in CSV that we load our development tasks and their evaluation into story points. In this format of data representation - each line of file is a table row.

The OS module - provides many functions to work with the operating system, and their behavior, as a rule, does not depend on the module, so the programs remain portable.

With this data we start our program:

```
# pip install keras
from keras.models import Sequential, load_model
from keras.layers import Dense, Flatten
from keras.preprocessing import text

# pip install numpy
import numpy
```

```
8 import os
9 import csv

11 sheet_path = ‘Tasks with estimation.csv' # full name of file
with table (no path  if gile lies in one directory with program)
12 text_for_check = 'codecodecode' # Text for the check
```

Next we present our two solutions for this program. The first is the transformation of a string into a list of letter indexes:

```
15 # def prepr(txt):
16 #     letters =
'абвгдеёжзийклмнопрстуфхцчшщьыъэюяabcdefghijklmnopqrstuvwxyz'
17 #     numrepr = []
18 #     for l in txt.lower():
19 #         if l in letters:
20 #             numrepr.append(letters.index(l) / 59)
21 #     while len(numrepr) < 200:
22 #         numrepr.append(0.)
23 #     return (numrepr)
```

The second solution is a string as a list of unique word hashes.

```
26 def prepr(txt):
27    ht = text.hashing_trick(
28        txt, 100000, hash_function='md5',
29        filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',
30        lower=True, split=' '
31    )
32    while len(ht) < 20:
33        ht.append(0.)
34    return list(i/100000 for i in ht)
```

In this case, the text from the table is presented to the program as an array of 20 numbers, each of which is a unique number from 0 to 1, and if the incoming phrase does not contain 20 words, the missing places in the array are filled with zero.

The next step is to take the phrases from the table and convert them to floating point numbers, so that the program has something to learn from:

```
37 file = open(sheet_path, 'r')
38 r = list(csv.reader(file))[1:]
39 keywords = []
40 for i in r:
41    keywords.append(prepr(i[0]))
```

In order not to go through all the values of the array, we convert getting Fibonacci numbers from the table to vectors:

```
44 numbers = []
45 for i in r:
46    i = i[1].strip()
47    if i == '1':
48        numbers.append([1.,0.,0.,0.,0.,0.,0.])
49    if i == '2':
50        numbers.append([0.,1.,0.,0.,0.,0.,0.])
51    if i == '3':
52       numbers.append([0.,0.,1.,0.,0.,0.,0.])
53    if i == '5':
54        numbers.append([0.,0.,0.,1.,0.,0.,0.])
55    if i == '8':
56        numbers.append([0.,0.,0.,0.,1.,0.,0.])
57    if i == '13':
58        numbers.append([0.,0.,0.,0.,0.,1.,0.])
59    if i == '21':
60        numbers.append([0.,0.,0.,0.,0.,0.,1.])
61
62 file.close()
```

Now partition our data set into a matrix of parameters (X) and a vector of target variable (Y)

```
65 X = numpy.array(keywords)
66 Y = numpy.array(numbers)
```

Load our built model:

```
68 # Load our built model
69 model = load_model('weights.h5')
```

We create the model, and add layers one by one. On the input layer input_shape = number of parameters to train. As input activation function we use ReLu (line 75) and the first number in the layer definition is the number of neurons [29].

ReLU can be expressed as follows:

$f(x)x^{+} = max(0,x)$, where x - is the input to the neuron. The advantages of using ReLU:

- Sparse activation: hidden blocks have non-zero output.

- Better gradient propagation: less problems with vanishing gradient compared to sigmoidal activation functions that saturate in both directions.
- Efficient computation: only comparison, addition and multiplication.
- ReLU - scale-invariant: $max(0,ax) = amax(0,x)$ for $a \geq 0$ [29].

In the output we use the Softmax activation function to get 7 results from 0 to 1 summing up to 1. Softmax function is used in machine learning for classification problems when the number of possible classes is more than two (for two classes the logistic function is used). Theoretically, the more layers and neurons in each layer, the more accurate the program, but in practice you need to find the best combination. The formula of the Softmax function is as follows:

$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$, where σ - is softmax, $\vec{z}$ - is the input vector, e^{z_i} - exponential function for the input vector, K - the number of classes in the multiclass classifier, e^{z_j} - exponential function for the output vector. This formula can be denoted as a transposition of the weight matrix of the program multiplied by the feature matrix [30].

Let's look at the implementation of using the input and output vector activation functions.

```
73 # model = Sequential()
74 # model.add(Dense(180, input_shape=(20,), activation='relu'))
75 # model.add(Dense(250, activation='relu'))
76 # model.add(Dense(180, activation='relu'))
77 # model.add(Dense(250, activation='relu'))
78 # model.add(Dense(7, activation='softmax'))
```

Now we need to compile the model using the Adam gradient descent and the Categorical crossentropy error function. Adam gradient descent is a replacement optimization algorithm for stochastic gradient descent. Adam (Adaptive Moment Estimation) is an update to the RMSProp optimizer. This optimization algorithm uses moving averages of both gradients and second moments of gradients. It can be

thought of as a stochastic approximation of gradient descent optimization because it replaces the real gradient (computed from the full data set with its estimate (computed from a randomly chosen subset of the data). Both statistical estimation and machine learning consider the problem of minimizing the target function, which are in the form of a sum:

$Q(w) = \frac{1}{n}\sum_{i=1}^{n} Q_i(w)$, where w is the parameter, which minimizes Q(w). Each term of the sum is usually associated with the i observation in the dataset used for training [32].

When solving any machine learning problem, it is necessary to be able to estimate the result, i.e., a metric is needed. In the case of multiclass classification (which is our case) we usually use categorical cross-entropy.

Let us compile the model:

```
82 model.compile(loss="categorical_crossentropy",
optimizer="adam", metrics=['accuracy'])
```

In the next step we train our program.

```
85 model.fit(X, Y, epochs = 100, batch_size=25) # epochs –
количество циклов обучения, batch_size – размер пакета
```

And let's evaluate the result:

```
88 scores = model.evaluate(X, Y)
89 print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

We display the prediction of the result and its output to the console as follows:

```
92 predictions = model.predict([prepr(text_for_check)])
93 prdct = predictions[0]
94 print('1:  ' + str(round(prdct[0]*100)) + '%')
95 print('2:  ' + str(round(prdct[1]*100)) + '%')
```

```
96 print('3:  ' + str(round(prdct[2]*100)) + '%')
97 print('5:  ' + str(round(prdct[3]*100)) + '%')
98 print('8:  ' + str(round(prdct[4]*100)) + '%')
99 print('13: ' + str(round(prdct[5]*100)) + '%')
100 print('21: ' + str(round(prdct[6]*100)) + '%')
101
102 options = ['1','2','3','5','8','13','21']
103 indx = list(prdct).index(max(prdct))
104 print('Predicted: ' + options[indx])
105 print(max(prdct))
```

Now let's save the resulting architecture:

```
108 model.save('weights.h5')
```

We can adjust the operation of our program as follows:

- by other settings of the incoming data (function prepr, lines 15 and 26);
- change the number of layers;
- change the number of neurons in each layer;
- change the number of training iterations (epochs);
- use different activation functions (activation);
- use different loss functions (loss);
- use different optimizer functions (optimizer).

We see that our program works as it was intended - entering any new task we get an automatic evaluation of the development task in Fibonacci numbers. The longer we will train the program - the better and more accurate the estimation result will be. Such a program will help to make better predictions about the amount of work, even before developers start estimating such work.

Conclusion

The main goal of our research was to create a program that will automatically evaluate software development tasks. The research goal was achieved by analyzing the theoretical and empirical data presented in our paper. In our study, we analyzed the theoretical justifications for evaluating software development tasks and validated them empirically based on data from our work with a test team.

We analyzed why agile methodologies are used in software development, using the Cynefin framework as an example, and determined that software development functions within the complex system of this framework. In such a system, good and best practices do not work, it is based on finding the best solution by experimentation. The key development problem within such a system is the impossibility of perfect planning and prediction, as some unplanned action can always occur.

Further in our research we took a detailed look at how SCRUM framework (which in turn only works in a complex system based on Cynefin framework) functions - we looked at how relevant it is for development teams (over 50% of development teams around the world use SCRUM), we identified the main roles and activities through which such a framework functions. We were interested in the planning activity, specifically the evaluation of tasks during planning by the Planning Poker method. In the third section of the first chapter, we disassembled how such planning is built, we disassembled its base, which was laid down by the Delphi method. We deconstructed the formula of the Delphi method and described the main idea of such a method - the analysis and subsequent processing of expert evaluations on the subject of evaluation, and, as a result, an overall valid and reliable evaluation. The value of review of this method for our research is that it has the same basis as the method for estimating development tasks in poker planning, with one exception - this method does not use Fibonacci numbers. Accordingly, when building our program for automatic task estimation, we consider how this method works, but

we do not build a model based on it.

The fact that Fibonacci numbers are used in the estimation of development tasks is due to the fact that the costs of any development task depend exponentially on its complexity. Such a complex relationship must be approximated. And one way to do this is to use Fibonacci numbers. Fibonacci numbers are able to approximate from sets of problems to sets of costs, and this contributes to the most accurate estimate of the development problem. In the second chapter of our study, we prove this theoretically and empirically.

We begin the second chapter of our study by reviewing the use of Fibonacci numbers in various fields - in architecture, in pseudorandom number generation, in coding theory, in stock price charts, and finally in software development. Identification of the benefits of using such numbers, confirms the fact that the choice of methods for assessing development objectives in Fibonacci numbers is really effective. This is confirmed by the data obtained empirically and analyzed in the second section of the second chapter - burndown curve and current process data.

This section focuses on data that was collected over a 15-month period from December 2019 through February 2021. The paper presents data from 29 sprints of the 7-person development team. For each sprint, a burndown chart is presented where we observe how the development task score changed over the 15 months. These charts showed a good result - on average 89% of the tasks in each sprint were completed, but the task score was not accurate in the first half of the empirical study (the first 12 sprints). From the graphs, we can see that the development test team achieved a fairly accurate score over 15 months, and continues to improve it over the course of the collaboration, and with each new sprint this score will become more accurate. Based on the graphs and results presented, we understand that the development teams themselves improve the basis of their estimates over time, and arrive at an optimal estimation capability, that is, adjusting the information about the elements of the backlog they are estimating. At the time of the study, the team was finding the exponential function empirically, that is, adjusting the information curve

experientially. We have automated this process in our third chapter - our program self-scoring in Fibonacci numbers, allowing us to immediately work in qualitatively planned sprints.

To conclude our research, we presented code for a program that automatically scores development tasks in Fibonacci numbers. The process of estimating the development tasks is not linear, and in order not to lose the exponent of the function it is very effective to use the Fibonacci numbers to perform this estimation. On this basis, we have created a program that classifies the pool of development tasks and evaluates these tasks in Fibonacci numbers. The code of the program is presented with comments. This program works on the basis of a data set consisting of 743 development tasks obtained experimentally in the work with the test team, which is especially valuable. Such a program solves a rather acute problem - the process of training the team to accurately estimate tasks in Fibonacci numbers takes a long time, and with automated estimation teams can concentrate on software development rather than estimating the labor cost of such development.

List of references

1. Berkhin P. Survey of Clustering Data Mining Techniques // *Accrue Software* - 2002. P.56.
2. Boehm W.B. Software Engineering Economics 1st Edition // *Prentice Hall* - 1981. P. 767.
3. Bernstein S., Bernstein R. Schaum's outline of theory and problems of elements of statistics. I, Descriptive statistics and probability, // Schaum's outline series, New York: McGraw-Hill- 1999. P. 21.
4. Bland, J.M., Altman, D.G. Statistics notes: measurement error // *BMJ* - 1996. P.312.
5. Coveyou R.R. Random Number Generation is too Important to be Left to Chance // SIAM - 1069. PP. 70–111.
6. Dalkey N., Helmer O. An Experimental Application of the Delphi // *Memorandum RM-727/1Abridged* - 1962. P.27.
7. Dinwiddie G. Feel The Burn, Getting the Most out of Burn Charts // *Better Software,* Vol.11, Issue 5 - 2009. PP. 26-31.
8. Elliott, R. N. Nature's law—the secret of the universe // New York: 63 Wall St.- 1946. P.64.
9. French S. Cynefin, statistics and decision analysis // *University of Warwick, Coventry, UK. Journal of the Operational Research Society* - 2013. PP. 547–561.
10. Frost A.J, Prechter R.R. Elliott Wave Principle: Key To Market Behavior // *New Classics Library* - 2005. P. 254.
11. Gantt H.L. Work, Wages and Profit // *Easton, Pennsylvania: Hive Publishing Company* - 1974. P.312.
12. Garson, G. D. The Delphi method in quantitative research // *Asheboro, NC: Statistical Associates Publishers* - 2014. P.38.

13. Greshilova A., Stakun V.,Stakun A. Matematicheskie metodi postroeniya prognozov // *M.: Radio i svyaz* - 1997. P.112.
14. Grenning J. Planning Poker // *Renaissance Software Consulting* Vol. 3 - 2002.
15. Heracleous, L., Jacobs, C.D.,Crafting Strategy: Embodied Metaphors in Practice // *Long Range Planning* Vol.41 - 2008. PP.309-325.
16. Knuth D.E. Fibonacci multiplication // *Applied Mathematics Letters* — 1988. — №1 PP. 57—60.
17. Krogh G. von, Roos J., Slocum K. An Essay on Corporate Epistemology // *Strategic Management Journal, Special Issue on 'Rethinking Strategy - The Search for New Strategy Paradigms* - 1994. PP. 53-71.
18. Krusche S. Rugby: An Agile Process Model Based on Continuous Delivery // *Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering* - 2014. PP. 42–50.
19. Rhee J., Nejad T.M. , Comets O., Flannery S., Gulsoy E.B., Iannaccone P., Foster C.. Promoting convergence: The Phi spiral in abduction of mouse corneal behaviors // *Complexity* // Wiley Periodicals Inc.— 2015. 20: PP. 22– 38
20. Rivlin T.J. Chebyshev Polynomials: From Approximation Theory to Algebra and Number Theory // *Pure and Applied Mathematics, John Wiley* - 1990. P. 249.
21. Rescher N. Predicting the Future // *Albany, NY: State University of New York Press* - 1998. P.332.
22. Shannon C.E. A Mathematical Theory of Communication. // Bell System Technical Journal. 27 (3) - 1948. PP. 379–423.
23. Shannon C.E. A Mathematical Theory of Communication. // Bell System Technical Journal. 27 (3) - 1948. PP. 623–656.
24. Snowden D.J., Boone M.E. A Leader's Framework for Decision Making // *Harvard Business Review* - 2007. PP. 69–76.
25. Viswanath D. Random Fibonacci sequences and the number 1.13198824... // *Mathematics of Computation* — 1999. — Vol. 69, no. 231. PP. 1131—1155.

26. Vygodskiy M. «Nachala» Evklida // *Istoriko-matematicheskiye issledovaniya* // M.-L.: GITTL - 1948. PP.217-295.

Sources on electronic media:

27. 13th Annual State Of Agile Report // Available at: https://www.stateofagile.com/#ufh-i-521251909-13th-annual-state-of-agile-report/473508 (accessed 25.05.2020).
28. A Million Random Digits with 100,000 Normal Deviates // Santa Monica, Calif.: RAND Corporation - 2001. Available at: https://www.rand.org/pubs/monograph_reports/MR1418.html (accessed 05.04.2021).
29. Brownlee J. // Available at: "A Gentle Introduction to the Rectified Linear Unit (ReLU)". Machine Learning Mastery. (accessed 13.05.2021).
30. Goodfellow I., Bengio Y., Courville A. // Available at: "6.2.2.3 Softmax Units for Multinoulli Output Distributions". Deep Learning. (accessed 13.05.2021).
31. Keras documentation // Available at: https://keras.io/#why-this-name-keras (accessed 13.05.2021).
32. Kingma D., Ba J. // Adam: A Method for Stochastic Optimization // Available at https://arxiv.org/abs/1412.6980 (accessed 13.05.2021).
33. Manifesto for Agile Software Development // Available at: http://agilemanifesto.org/iso/ru/manifesto.html (accessed 25.05.2020).
34. Sloane N. J. A. The On-Line Encyclopedia of Integer Sequences. "Sequence A000045"// Available at: https://oeis.org/A000045 (accessed 27.02.2021).
35. Scrum guide // Available at: https://www.scrumguides.org/ (accessed 25.05.2020).
36. The NumPy Open Source on Open Hub // Available at: https://www.openhub.net/p/numpy/analyses/latest/languages_summary (accessed on 13.05.2021).
37. Wachsmuth, Bert G. MathCS.org - Real Analysis: Ratio Test // Available at: www.mathcs.org (accessed 25.05.2020).

Appendix

Attachment 1 - Data set with tasks and estimation in story points

Summary	Story Points
Read the JWT cookie	2
Set a signed JWT cookie	2
End-point giving metrics	3
Move the base update window	2
Set up speedcurve to measure tabs	3
Set up sloops	5
Add google optimize code	2
Modal window tweaks	3
Trim header	3
Sorting task list	3
Getting tool data	3
Track of old RabbitMQ	3
Product does not work in 13 safari	1
Error condition doesn't work	5
Put link to button	1
Load testing as a service	5
Release to prod	1
Give front end user email	3
Reflect email	1
ME controller: Give front end information about the user's view	2
Put a cookie on the backend	2
Show dropdown with data	5
End-point that defines the user	3
End-point that gives a list of users	3
Template a new block with a user selection selector	3
Avatar for users with random picture generation	5
Filter users by tabs	5
Create a test account	1
User level logic	5
Don't hide the intu	1
Updated section of the landing page	3
Uploading the company logo	2
Tags on the product	3
Put the trigger for tests in a separate stage after the deployment	5
Put a cookie on the backend	3
Invalid time error	2

Product not opening	3
Show button	1
Fix the page header flicker	8
MR in the text changer	1
End point for sending emails	2
Add new product to the list	1
Add a header on the list	2
Move to UI kit.	5
Display forks after date on list	1
Tab transitions don't work	3
Zeros in time and price don't work	2
Remove auto-scroll on tab transitions	2
Banner layout	3
Get a button on the view page	1
Time in days is not displayed	2
Updated landing page	2
Modulka edit does not close the first time	3
Selector Event	2
Cut out products from the tul list in the create/edit form	1
One chunk and data loader for tabbed pages	5
Remove flicker states	3
Don't block queries and loading of chunks	5
Inplay edit all fields	5
Make selectors editable	5
Add i-tags	3
Visual grouping	3
Display rules for groups	5
Integer numbers in the input	2
Notify external services about limit changes	5
Spike on file storage	8
Eliminate the reason for Layout Shift increase	5
Display settings	5
Misfeed state	3
Local time display	3
Storing the date in UTC	2
Put html tooltip	2
Auto group selector	2
Empty stack	2
Screw dev tools into tests	3
Unified modal deletion + events	3
UX finder fix on search	2
Allow a page to be indexed	3
Limit dropdown stats	5

Above-cap in display mode	5
Make tab available on prod	1
Add create time/update time for all entities	5
Sort by creation date	2
Add days field	2
Redirects	3
Localize	2
Changing fields	3
Create a task from the list: edits	3
Make connector for analytics	5
Limit output of list depending on cookies	5
Make protection for controllers	3
Mode for users with subscription	3
MR: Make changes in list controller	3
Service at the front	5
Rendering on the backend	5
Card adaptivity	3
Order of tabs	1
Downgrade state	3
Loading state (sloops / skeleton)	3
Data collection stack	2
Black product stats	3
Installed tools stack card	5
Ifchik for 10+ products	1
Layout card	3
Blank card stats	3
Time by task: intut + selector	3
Spike on product	5
Change edit icon with appearance when hovering	2
Ability to interact with links	5
Release the iron from the instance s	1
Disable GCP	1
Remove configs	1
Remove the load balance from the load balancer.	1
Soft-delete	3
Get audited for release	5
Retrieve information from the beck	3
Start auditing for the release	3
Remove autoinput	1
API /dashboard/api/graphql does not enforce limit	2
Include default data from the beck	3
Creation modal does not work	3
Updating packages and applying them	3
Replace prices on limit stubs	3

Cyrillic alphabet does not work in public API	3
Amplitude integration	8
Front takes colors from beck	5
Beck should give default flag and color marker for selectors	3
Information icons in a modal	2
Remove banner	1
Cut out the modalks, leaving only the button	2
new currencies for the budget	3
JS Total Blocking Time and Largest Contentful Paint	5
Differentiate types in the dropdown	3
Product service: get list	5
Selector data hook takes data from back (not from moc)	3
Try webtau	2
refactoring test classes	5
create contracts	5
Visual testing	5
Screw AssertJ to the autotests	5
The data for the selectors is taken from the moc through a hook	3
Form Validation	3
Create a service for the form	5
Screw in ready to use selector components	5
Banner	5
Possibility of transfer to another URL	8
Remove badge in the modal code	1
Optimize page speed for the page	5
Create a new tab	5
Remove banner	1
Backup	5
Api for selectors	3
Create User	1
Link stats	2
Fix domain adding with il-ext database	5
Fix reload of landing page	3
List project ids and remove their http requests	3
Counter with a trigger for add-ons	3
Separate module - edit form	5
Tools display for working in another mode	2
Multiselector tool selector	3
Api that returns a list to the selector	3
Add a total cost and time calculation for each group	5

Blank page summary	3
Quick creation from a list	3
Block due date - colors	3
Editing gear	5
Create groups of cards	5
Display task cards on a list	5
Creating from a list / empty stack / page	2
Search for tasks (words, letters, etc)	3
Lock header	1
Screw limit setting	5
Explore visicon	2
Navigate on form	3
Layout: Create Form	13
Skeleton to load a list page	3
Back: Create and Edit Api	5
Back: Create a product entity	8
Create a tab with an empty stack	2
Edits on the whale after the deploat	3
Table settings drag&drop fix	3
Transfer to GCP	5
Pass the data on instead of writing it directly to bigquery	3
Fix banner design	1
Sort by Created date	2
Don't show in hundredths of .00 if you don't enter it	1
Change salt	5
Product not showing up for some users (selective)	3
Upload Upload to Product Limits	5
Exposed actuator	3
State when user has already filled out the data	3
Modal window with 2 experiment steps	5
New tab with a blank state	3
Banner on the page	3
Add button	2
Make a separate locale	3
By hiding a block the user no longer sees this block as open	1
Make limit 30	5
Backend technical improvements for integration	5
Product drops	3
Give prices by API	3
check limits when editing	3
return the blocked token when accessing the page	2

fix tab indents	1
Dropdown in menu disappears when hovering	2
Component for page locking	5
Profile page stats when user downgrades	3
Update metrics in speedcurve	1
Error with time generation in unit tests and screenshot	3
Disconnect appears only after reloading window	2
Downgrade whale components in the NPM package repository	8
Disclose incidents	3
Close product	5
postmortem	5
Disassemble incidents	3
Generate users with data entry	5
Specify full country name in the header	1
Breadcrumb events on Create/Edit pages	2
Deletion modalizer events	1
Deletion modalka equivalents on the list	1
The exact name of the breadcrumb labels in the event	1
Change url	3
Remove Roman House	2
Downgrade letter.	8
User downgrades and we block him	3
Card without a lot of empty space	5
Added redirect	2
Limit service framework	8
Jira shuffle for uptime incidents	5
Not found page on a non-existent page	3
Add limits to server	8
Duplicate tab event	1
After adding, open the setup in a new tab	1
Add connect/ add new buttons	1
If you fill in a note with line breaks, then the text is merged in the card.	1
close /api/actuator/prometheus	3
The connect modulka is not passed if there are no projects	3
Cancel in confirm delete form does not work	2
Metrics	5
The snap button in the modal is broken	2
Connect speedcurve	5
Set up speedcurve notifications	3
Revalidate country names	2

Not filled in data in edit form	3
Connect modal blinks	3
Fix buttons in the table on the wide screen	1
stress testing	5
Show skeleton to load list	2
Update the skeleton profile	2
Fix the form loading process	1
Fix tests for utils	3
Fix the Fullscreen Modal binding loading process	5
resolve incidents	2
Show asterisks and ishku	3
Beck give max and used front	3
Investigate incident	5
Add a Clear all filters button on Nothing found stats	1
Fix the order in the linked projects	2
read a message from a rabbit about deleting reports	3
Reads messages from rabbit about deleted projects	5
Reports write us about report deletion in rabbit	2
Projects write to the rabbit when deleted	3
Projects write information about project deletion to the rabbit	2
Remove project deletion	5
Load testing model	5
SRE checks	5
Create/edit form: fix basement with Save / Cancel buttons	3
Static load problem - folbeck script	3
Serchbar in firefox doesn't redirect to report	3
Create ME service with customization	3
Create chat room in slack where feedback is sent	2
Create/edit: labels are not in the center of the fields	1
Tabs events	3
ProfileIvents	3
List flash incentives	3
Create and edit events	3
Listing events	3
Feedback window	3
Changing icon in menu	1
Type and status - impossible pairs	5
Tool reproduction	2

Create modal: replace text	1
Update checkbox on creation form	2
Make it possible to create based on availability of data in local storages.	5
Unlogged user gets to login form	2
Add validation to forms	5
Service watering	2
Bind blocks to column grid	2
Editing styles in the header	1
Styles changes in header	1
Styles of texts on the tab	1
Configure manual branch deplugging	2
Turn off a rabbit	3
Switch writers	3
Switch readers	3
Federated Queues	5
Instance in GCP	5
List Drawing.	8
All or some of the tools are set off	2
Validation hits the eye when you start printing	8
spinners are not perfect	2
improvement of instances	2
Unlinking reports: do a built-in update	2
Sort countries by frequency	2
Complete validation of website field	2
Client is not deleted	5
Fix display	3
Add validation from design to the Other field	2
Change the size of controls	1
Make JS forms for landing page inside the existing landing page	5
Landing page layout	5
Landing page redirects under different conditions	5
information about the number of linked projects is given along with the main information	5
On the internal page a block with the number of projects	3
the number of linked reports is given together with the main information	3
the block with the number of reports on the inner page	2
Beck gives back 400 errors when enabeling tied reports	3
Scroll reset on any page change	3
Add a banner to the reports	3

breadcrumb contains a broken link	1
alphabetical sorting doesn't work	1
email goes outside the scope of the profile	2
back should not enablature legitimate reports	3
bug budget field	2
Left menu item on Prode	2
Authorization key	3
Traffic	3
Wrap up the traffic on the lowd balance	3
Helm profile for the sale	2
Empty list state	2
Empty report stack	2
List of results not found	2
Do auto-import and remove import button from widget	5
Downgrade the list of linked projects	3
Performance testing	5
Create/edit: Cancel does not close form and does not go back	1
Refresh the stor with tests after editing	3
Creating a modal page	5
Drawing the anchor list of projects	8
Creating front end service of projects	8
Stack: nothing is set	1
Empty list state	3
Implementing project creation	5
Component to forward the property apiUrl	3
project creation and immediate binding	5
project unlinking	3
saving bindings	3
gives the list of projects	3
lists all projects of a user with information about binding	8
Profile: Page not found	3
job title other	2
make the website optional	1
States Something went wrong	2
remove empty fields from pages	2
Replace the button in the header of the profile	2
During editing change button to save changes	1
Add breadcrumbs	2
Phone block - remove everything, leave only the incrpu	1
Bind navigation block to form	3

Load testing on virtual machine with transferred toolkit-landing	5
Functionality transfer to the GCP virtual machine as it is.	5
List header: fix scrolling search and sorting	2
Search in the list of reports	3
Saving the select status values on the card and in the profile	2
Finish the country field with flags	3
Updating the stor with tests	3
Create test units	3
Pass the Security test	3
Pass SRE check	3
Popup Success	1
After creation make redirect to the page	2
NoticeBubble notification about lack of connection	1
Modal confirmation window on deletion	2
Detach report	2
Sending a detachment request - Detach from the list	3
Saving report bindings	3
Beck sends a list of all reports with information	3
Drawing a list of report bindings	8
Drawing list (sorting)	5
List of linked reports gives Beck	5
List: UI edits	3
Facility form: UI edits	2
Finalize industry field	3
Remove serchbar	1
mask the phone's intu	2
Width of cards on different screens	2
Notes appearance	3
List sorting	3
Stats of cards in the list - what went wrong	3
Service - editing the client	3
Service - deleting the client	1
Service - receiving the client list	5
Service - creating a client	5
Add contact type and status	2
Job title - create incumbent	2
Split incumbents - telephone incumbent	2
Profile: scroll and focus on the first field with an error on the subpage	3
Create page entity	5
Card in a list (UI part of the list)	5

Generate 1000 clients to test the rendering speed of the list	2
Universal header component	5
Excluding sloppy clients from speedcurve	2
Back customer profile	5
Component of form completion progress	3
Move user change notification queues to cube	8
Relocate feedback script from blades	5
Switch admin balasers directly	5
Switch the seller to the cloud base	3
Error when parsing js	3
Table header gets stuck on scrolling	2
Error in tag creation	2
Caching	5
Validation in profile	5
metadata	5
Update connectivity document	2
Remove dev dependencies on the prod	2
Caching selectors	3
Change Deployment strategy	5
Optimize settings to overcome connection pool overflow problem	5
Load, test	5
Make a correct connection to the inspector	5
Config connector	3
Authorize	3
Login point in the left menu	3
Banner: Login point	2
Performance metrics backend	5
Frontend build	8
Base via config connector	3
Gitlab CI	5
Autodeploy	2
CRM localization	2
Helm	5
Preparing environment for a new product	3
Explore the feasibility of memo hoc	3
Translate the deployment to helm	5
Change the cache settings to return preochet templates from gcp	3
Deletion data does not always reach BQ	5
front end doesn't return request after quantity changes	3
Close the plate	1
Add custom metrics to prometheus	5
Send metrics to prometheus	5

Bring main metrics to graphana	5
Remove multishare reload	3
Make a list of components in need of refactoring	5
Switch to LM from not found state	1
Fix links in the admin panel	1
Define tool settings (colbeck, limit url) through config	5
Drop all js	3
Switch to hook	8
show banner on all	1
Switch to docker images in all repubs	2
Fix build	3
problems with adobe integration	3
set up monitoring dashboards	3
data is coming from backend on closed limits as well	3
Bug on dashboard	1
Experiment phase 3: 30% of users are shown dashboard by default	5
Limit 5 domains instead of one	1
Button went wrong	2
autosharing of the first project crashed	2
Saw nailed down and added animation to show the modal	3
Move docker images into Artifactory	5
Take the header out of the package	5
list loading as needed	3
Bug with the startup and restor	5
Remove unnecessary connections to the base and rabbit	3
Failed to receive data: put /me in the request	3
Embed spinner in HTML on loading (in the center)	2
Make sure that slopers do not block the interface	5
Reduce response time of tools on the backend to 200	1
Works in DFC	3
Swap dropdown options	1
New dominant button design	3
Change the text with Restore->migrate	1
Cache invalidation by domains several times a day	2
Spinner loading	2
Spike: how to make the dashboard faster?	8
ellipsis in the header does not disappear if possible	3

attribute in /me	3
Remove page on prod	2
Optimize nanosheet rendering	5
Spike: string rendering	8
Event to display a tooltip	1
Publish helm charts	3
Raise instances in a prol cluster with a base	5
Modalka with form and thank you + event	5
Banner + Event	3
Mortal next2 in master, codefreeze by backend	3
give statics with GCP	3
ok got it not clickable	2
remove assembly	3
Tab is gone	2
Make up and make a link with get parameter	3
Add date of last data collection	5
Add tool names	3
Make a banner with the closing date	3
Poor sorting of columns	2
Require timeout on slow internet	5
Change import button	1
Search filter slow	2
Automatic table update	5
Cut banner	2
Localize projects	3
Add custom metrics to your tool	1
Neutralize Hindu "ddos"	5
Swap dropdown options	1
Feedback on switch to slack	2
Filtered state	2
Blank state of projects	2
Fixed feedback fonts	1
Wrap the table in a scroll	2
Show tab by default	3
Connect to Responsive template	5
Update components @react-semcore	5
unlogged project page is empty	3
cast dns for haselcast cluster via helm chart	3
update helm chart projects	5
Check for downgrade with limits exceeded	3
Stat sheet when no data is received	3
Editing via helm chart - switching to form	5
Filtering data in the notebook - search	5

Profile (no validation)	8
Limits dropdown update	2
table header sorting	3
Make splunk alerts	3
Performance upper boundary	5
Add gitlab-ci deplug	5
create a database in the project and transfer the backup there	5
Create k8s nymspaces for the projects	3
The spinner should display instantly without a blank screen	3
Empty projects should be displayed immediately	3
Tech debt in npm	5
Run product in kubernetes	8
Synchronize global storages between builds	5
Users with Adobe	3
Notify service: package	5
Unit tests to load global projects	3
Validate domain and base imports from the old dash	3
fix nana	2
Delete setup data of old campaigns	2
Run load	3
Automate input data submission	3
Automate authorization	3
Build load model	3
Run load testing	8
Link to new tool	1
helm add secrets	3
Helm give up sql-proxy	3
Helm set all customization to default lues	3
Transfer: Additional Stats and Events	3
Extra info in the Roman House feedback	1
Extra info in notis	1
Extra button	3
Bug with long answer	2
Bug with duplicate domain positions on the backend	2
Fiddling with the design menu knobs	5
Display Roman House	3
Implementation of splitting traffic into 2 parts	8
Move first packet to module	5
delete 403	3
Configure project front-end in docker	2
Safari 12 does not give list	5

Bug when trying to delete previously deleted (exposing backend internal classes)	2
Pass the backup check	8
tag after a balloon should be removed from an instance	2
multishare filter does not work properly	2
Investigate redirect problem	5
Widget display on wide monitors	5
does not check out on the root itself	8
Spike: investigate menu	1
add positive tests to PURR	3
fix padded create button	1
Divert traffic to release phase 1 of the experiment	2
Make a channel in slack with feedback	3
Transfer settings from old product to new one on-demand	5
Fix the bug with value display	1
Get rid of the whale style duplicates	2
Cut the feedback bar	1
GA event: display event	1
Spike: find a way to synchronize data packets	5
GA events: add an event to show stats	1
Pingdom notifications.	3
Brackets and i curve are slipping	1
Bug with removal	2
Locating nginx in frontend balancer	3
Locating nginx in cubes	2
Perl module logic in nginx in cubes	5
Widget does not show targets	3
Remove old docking implementation	3
Translations in the interface	1
Manually "register" tools	3
GA event: Events on duplicates	3
Remove button and hang event	1
GA events: Event creation modalities	3
Move into a bundle	5
Raise instances in the product cluster	3
Deploy the Prod via ci	8
Infinite spinner in the create button from an empty stack	2
GA event: Display an empty stack	1
Update view	1
Sending metrics to Prometheus	5
Sending logs to splunk	3
Transferring data export tasks to BQ	5

Moving the extension to GCP	2
Move project deployment to helm	5
Move gaService to a package	3
Move localStorageService into a package	3
Move Service into a package	5
Move product to another repository	8
only the limit should be requested	2
/#create - the project creation link does not work	3
If no data was given do a Reload state line	5
GA events: banner and roman house	2
Add a bagel	2
Rename taboo	1
Work out tool loading	5
Synthetic speedcurve monitoring	2
Connect LUX	3
Rename column headers	1
Checking the blocking task	5
Positive delta metrics must be with plus sign	1
Publish package to fix location bug	1
Add tooltips to deltas	1
Invalid closing	2
fix 404	3
Widget crashes a lot	1
Adjust timings on all operations with dependencies	1
Invitation in notifications	1
create project: url must be truncated to domain	3
input create project	3
UBS java client can't find users with wildcard characters in url	3
Words in locales are truncated	1
Fixed unknown tool	3
State when no more can be added	1
Crop into ellipsis	2
Skeleton should be displayed in the widget when select country is loaded.	2
container uses limit service	2
Localization inside the container component	2
container to a common package repository	3
Front should be checked only when changed	5
Create a stack when the limits run out	3
Trimming whitespace and url passes	2
Delete/modify popup does not open	2

Instant create does not work when sending additional parameters and exceeding limits	3
Pass SRE check for main dashboard to go into beta	5
Create a task list for moving to cubes	1
Move the package	5
Create rules for validating an instance	3
Unify names	2
click gives an error	1
Change the title	1
Multisharing dialog	3
No flag in country selector	1
Check whale email tags	1
Adding Event Value	2
Spike by page loading steps (smart loading)	3
Switch traffic	1
Fix login button	2
Rearrange "table settings" and search	1
Put buttons on the level with the subheading	1
Put a header	1
Default tool position	1
Front should be responsive to 400tki	3
Dropdown layout has moved out	2
Front does not respond to dropdown when changing the flag	3
Change category (analytics)	1
Tultip in the modal code	1
Redirect from serch bar	2
Button and popup localize and move to a shared repository	3
Pills behavior in string search	2
Make actuator and /API/headers endpoints secret	5
Remove introspection from the sale	3
Change the text in emails	3
Remove the experimental left menu	2
Make authorization without secrets	1
Replace the add button with a skeleton	5
Add inspectlet	3
Change the title	3
Split traffic spike - 1st part of experiment	5
Backlighting on the prod	2
Update UX helper	2
Make nice-looking intuitive control (blue highlighting)	2

Put localization into appropriate components	5
Remove hiding condition if you have projects	1
Backup 101	5
Remove button from serch bar	1
Button in a bundle	2
Move elements to a new UI kit	5
Sloppy functionality does not work on the front	5
State errors on the project list if the first request didn't work out	3
Doubling of the tool list in the project deletion modal (this happens if there are sloops)	3
Making changes in new dashboard design (UX logic of the page)	5
popup in the background	2
Duplicate requests - study the problem	3
Translate projects	1
Support whitelist update	2
Something went wrong	1
Add error stack - Project not found	5
Update button and dropdown Limits	3
Checking the login script	2
Run the logon script	2
Check your login script	3
Banner with a questionnaire	3
Add a dropdown button to the package	3
Problem with external localization	3
Redistribute traffic in admin balancer	5
Services Recombination	5
Additional table stats	8
Clean up notifications	3
Add Project Buttons	5
Add a serge bar	1
Debug redundant renders	5
Count and composition of chunks	5
Move CSS to a separate file	3
performance	5
Make user manual with link	1
Need stats that went wrong	2
Make send feedback with a link	3
Add state Failed collection attempt (tool set-up)	3
Add toolset stats (first data collection)	5
Build JS statics	3
Deploy stats	3

Porting styles	3
Pick up minimal build	5
Block transfer	8
GA events to widget	2
Migration to another API	5
Update the header	3
Redirect unlogged to the root	3
Migration of cache to GCP	3
Build data-test-id	5
Prepare survey banner	1
Prepare new UI for the analytics widget	3
Nothing found for the table (Styles/colors)	1
Tulkit dependencies	2
Show banner	5
Adding data	3
Implement mini charts	3
Make a state with domains	3
Dividing block	3
Autodisclosure block	1

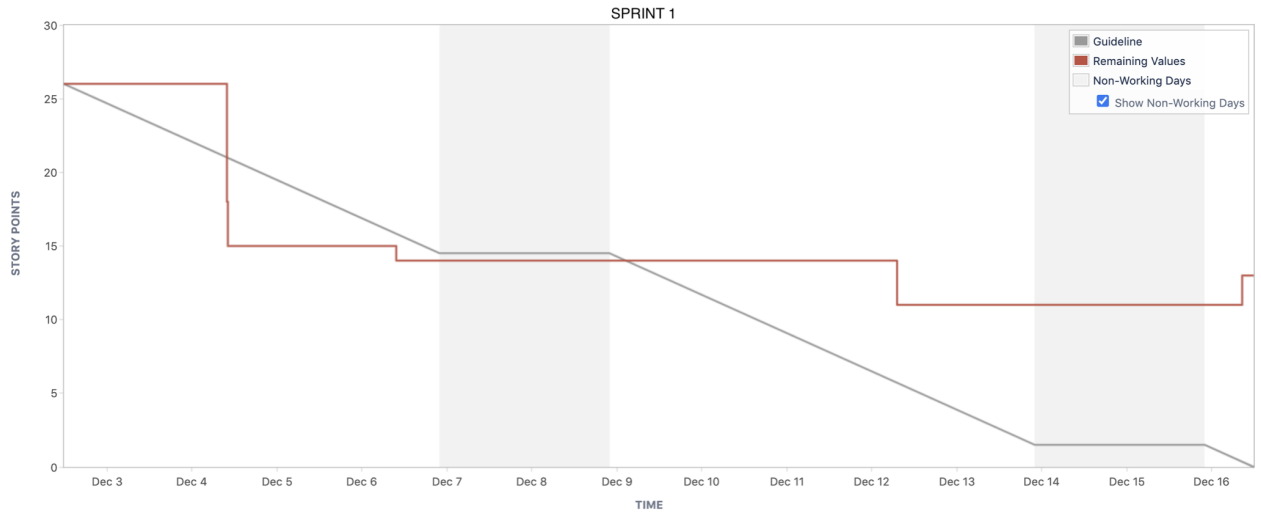
Attachment 2 - Code base

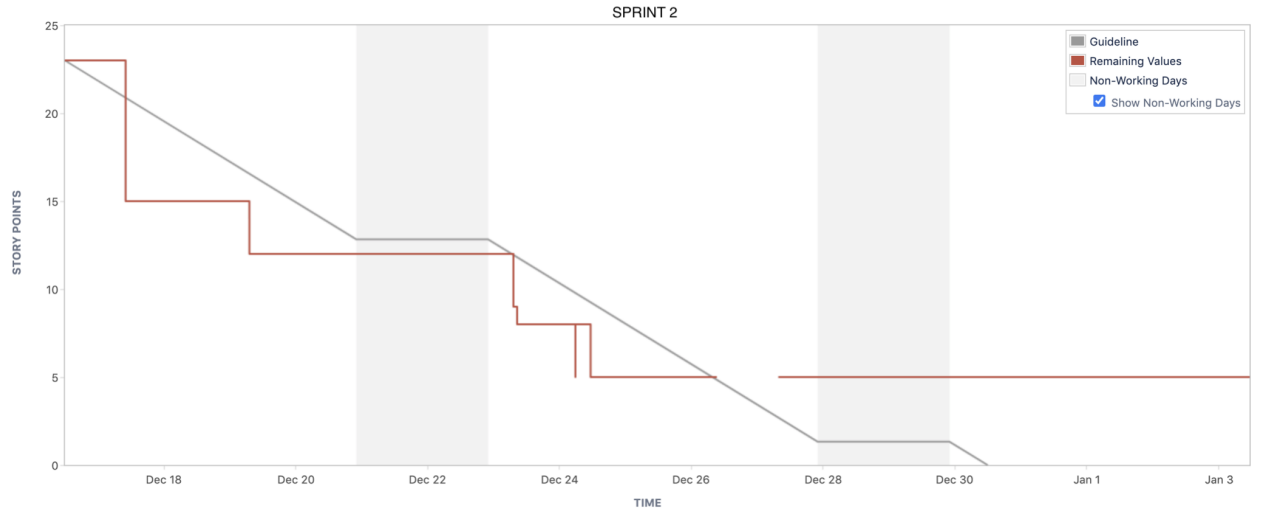
```
# pip install keras
from keras.models import Sequential, load_model
from keras.layers import Dense, Flatten
from keras.preprocessing import text

# pip install numpy
import numpy
import os
import csv

sheet_path = ‘Tasks with estimation.csv' # full name of file with table (no path  if gile lies in one directory with program)
text_for_check = 'codecodecode' # Text for the check

# Option for converting a string to a list of letter indexes
# def prepr(txt):
#     letters = 'абвгдеёжзийклмнопрстуфхцчшщьыъэюяabcdefghijklmnopqrstuvwxyz'
#     numrepr = []
#     for l in txt.lower():
#         if l in letters:
#             numrepr.append(letters.index(l) / 59)
#     while len(numrepr) < 200:
#         numrepr.append(0.)
#     return (numrepr)

# Option to convert a string with a list of word hashes
def prepr(txt):
    ht = text.hashing_trick(
        txt, 100000, hash_function='md5',
        filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',
        lower=True, split=' '
    )
    while len(ht) < 20:
        ht.append(0.)
    return list(i/100000 for i in ht)

# Getting phrases from a table and converting to floating-point numbers
file = open(sheet_path, 'r')
r = list(csv.reader(file))[1:]
keywords = []
for i in r:
    keywords.append(prepr(i[0]))

```

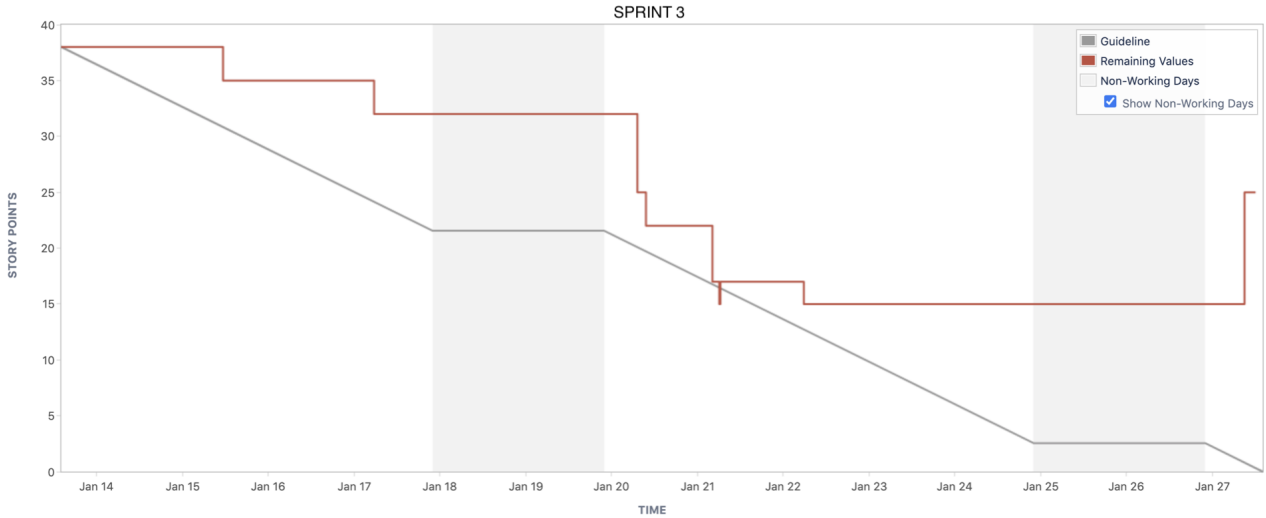
```
# Getting numbers from a table and converting them to vectors
numbers = []
for i in r:
    i = i[1].strip()
    if i == '1':
        numbers.append([1.,0.,0.,0.,0.,0.,0.])
    if i == '2':
        numbers.append([0.,1.,0.,0.,0.,0.,0.])
    if i == '3':
        numbers.append([0.,0.,1.,0.,0.,0.,0.])
    if i == '5':
        numbers.append([0.,0.,0.,1.,0.,0.,0.])
    if i == '8':
        numbers.append([0.,0.,0.,0.,1.,0.,0.])
    if i == '13':
        numbers.append([0.,0.,0.,0.,0.,1.,0.])
    if i == '21':
        numbers.append([0.,0.,0.,0.,0.,0.,1.])

file.close()

# We divide the dataset into a matrix of parameters (X) and a vector of the target variable (Y)
X = numpy.array(keywords)
Y = numpy.array(numbers)

# Loading the built model
model = load_model('weights.h5')

'''TO WRITE ANOTHER MODEL, UNCOMMENT THE CODE BELOW'''
# # Creating models, adding layers one by one.
# model = Sequential()
# model.add(Dense(180, input_shape=(20,), activation='relu')) # On the input layer input_shape = number of training parameters
# model.add(Dense(250, activation='relu')) # Activation function ReLu
# model.add(Dense(180, activation='relu')) # The first number in the layer definition is the number of neurons.
# model.add(Dense(250, activation='relu'))
# model.add(Dense(7, activation='softmax')) # Softmax activation function to get 7 results from 0 to 1 in total giving 1
# # In theory, the more layers and neurons in each layer, the more accurate the network operation, but in practice you need to find some optimal combination

# We compile the model, use gradient descent Adam and the error function Categorical cross entropy
model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=['accuracy'])
```

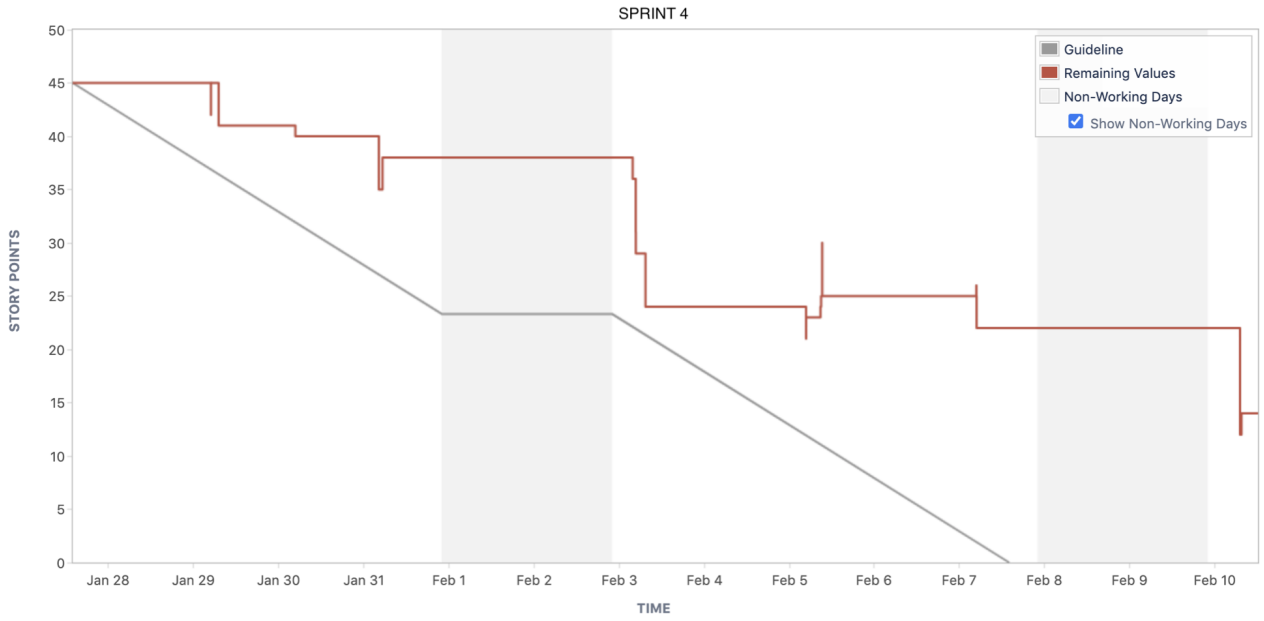
```
83
84 # Training a program
85 model.fit(X, Y, epochs = 100, batch_size=25) # epochs - number
of training cycles, batch_size - package size
86
87 # Evaluating the result
88 scores = model.evaluate(X, Y)
89 print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
90
91 # We predict the result, output it to the console
92 predictions = model.predict([prepr(text_for_check)])
93 prdct = predictions[0]
94 print('1:  ' + str(round(prdct[0]*100)) + '%')
95 print('2:  ' + str(round(prdct[1]*100)) + '%')
96 print('3:  ' + str(round(prdct[2]*100)) + '%')
97 print('5:  ' + str(round(prdct[3]*100)) + '%')
98 print('8:  ' + str(round(prdct[4]*100)) + '%')
99 print('13: ' + str(round(prdct[5]*100)) + '%')
100 print('21: ' + str(round(prdct[6]*100)) + '%')
101
102 options = ['1','2','3','5','8','13','21']
103 indx = list(prdct).index(max(prdct))
104 print('Predicted: ' + options[indx])
105 print(max(prdct))
106
107 # Saving the resulting architecture
108 model.save('weights.h5')
```

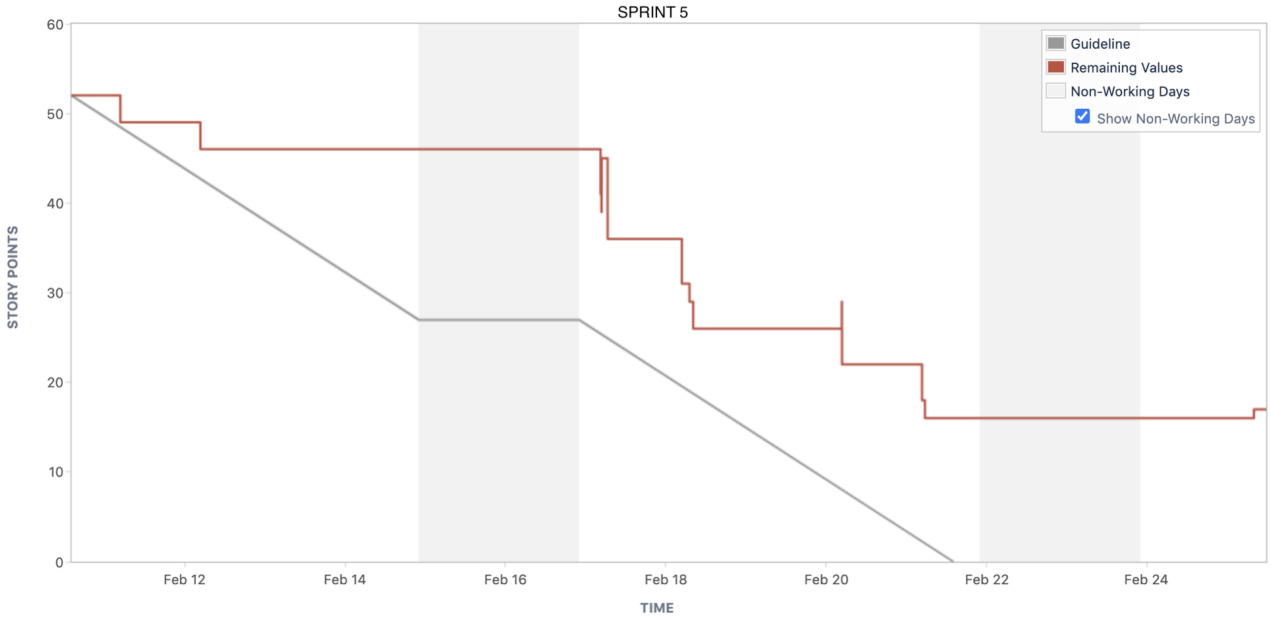
Attachment 3 - Sprint burndown charts 1-29

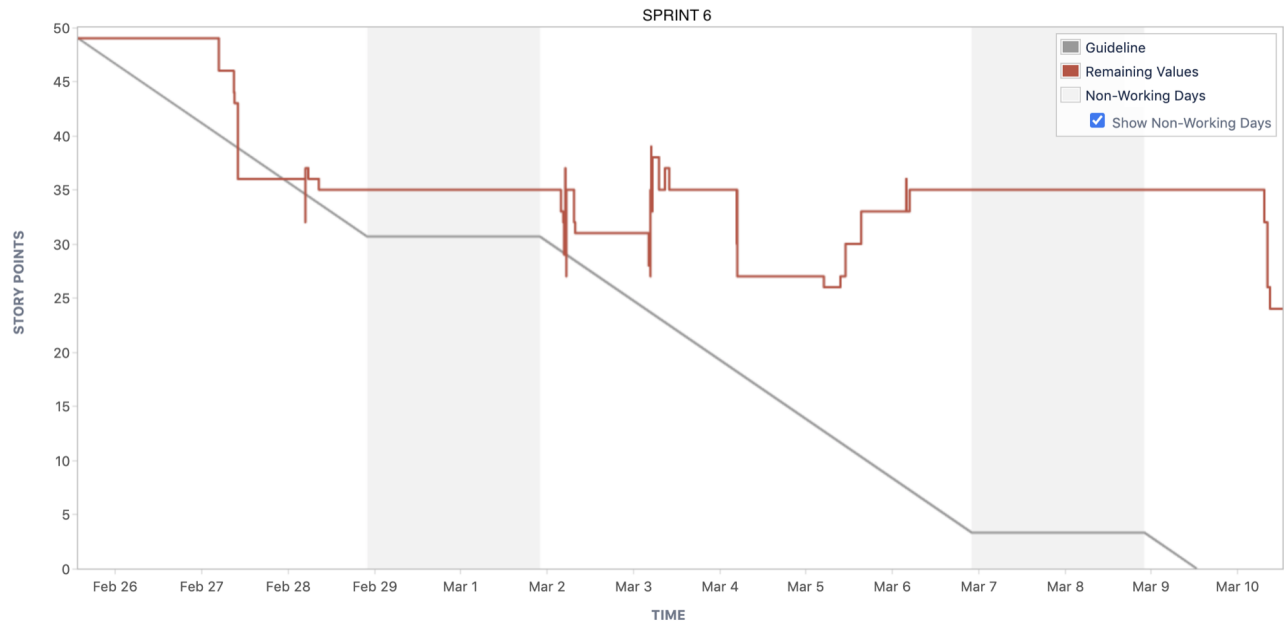


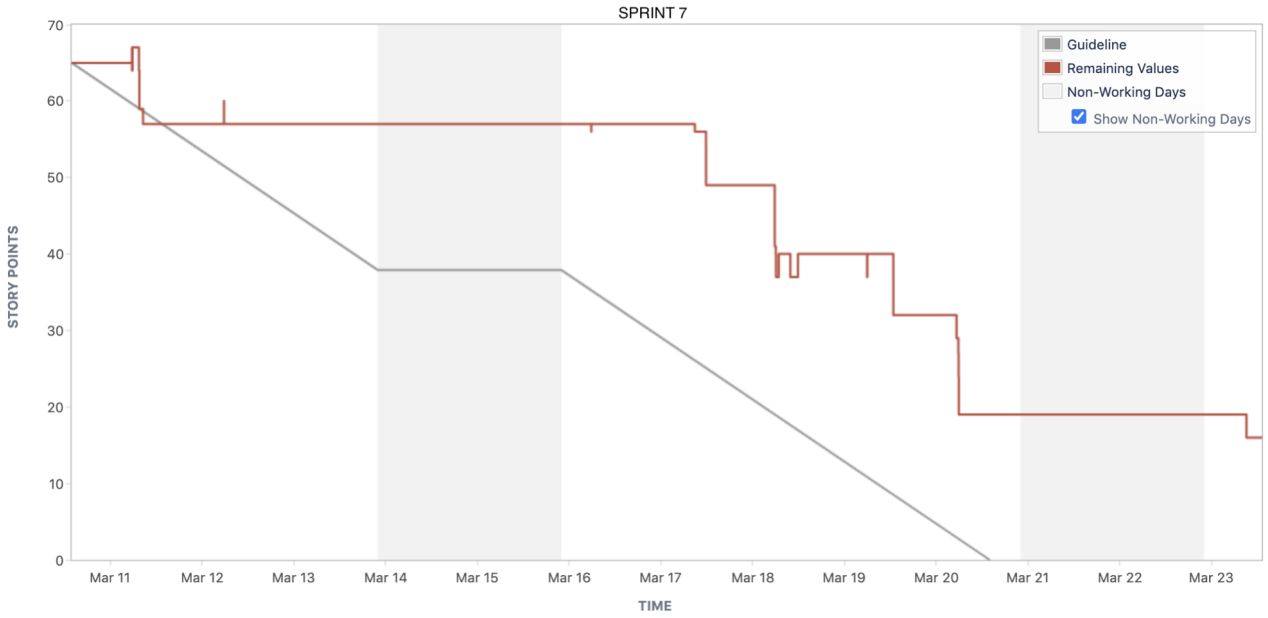


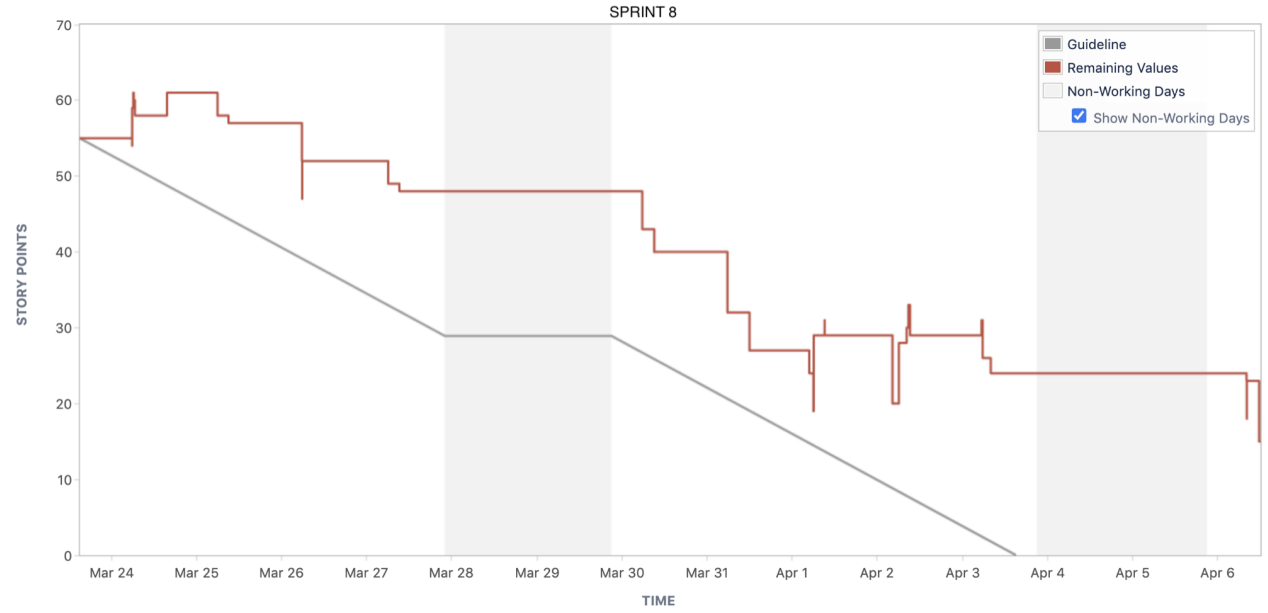


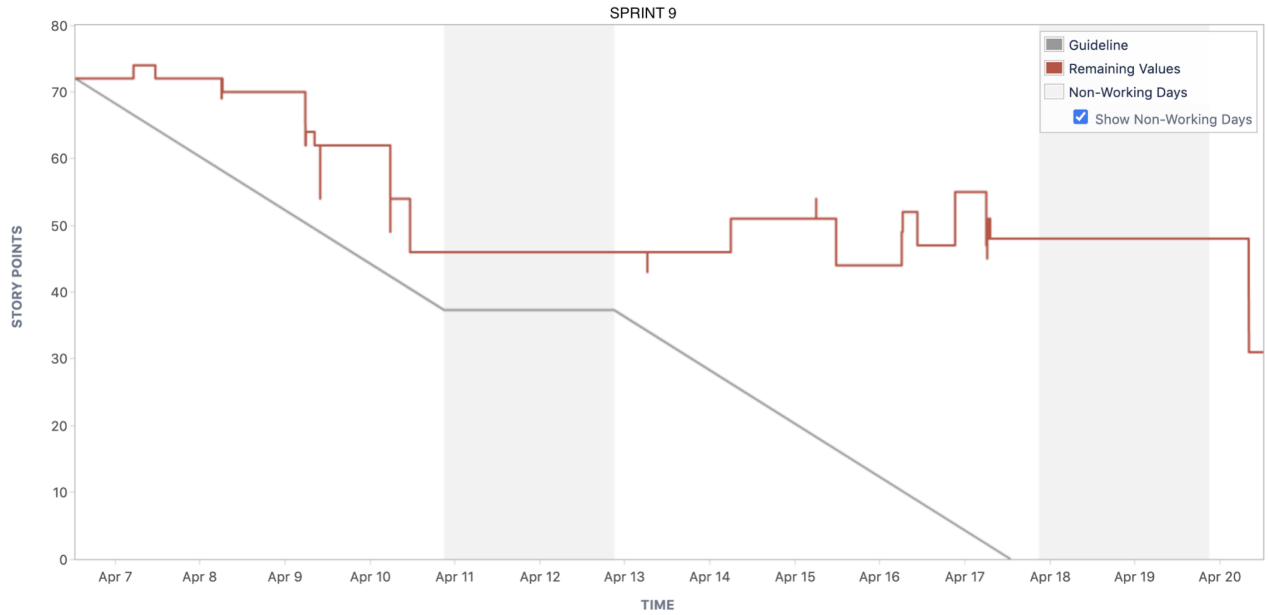

SPRINT 4
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
STORY POINTS
50
45
40
35
30
25
20
15
10
5
0
Jan 28
Jan 29
Jan 30
Jan 31
Feb 1
Feb 2
Feb 3
Feb 4
Feb 5
Feb 6
Feb 7
Feb 8
Feb 9
Feb 10
TIME

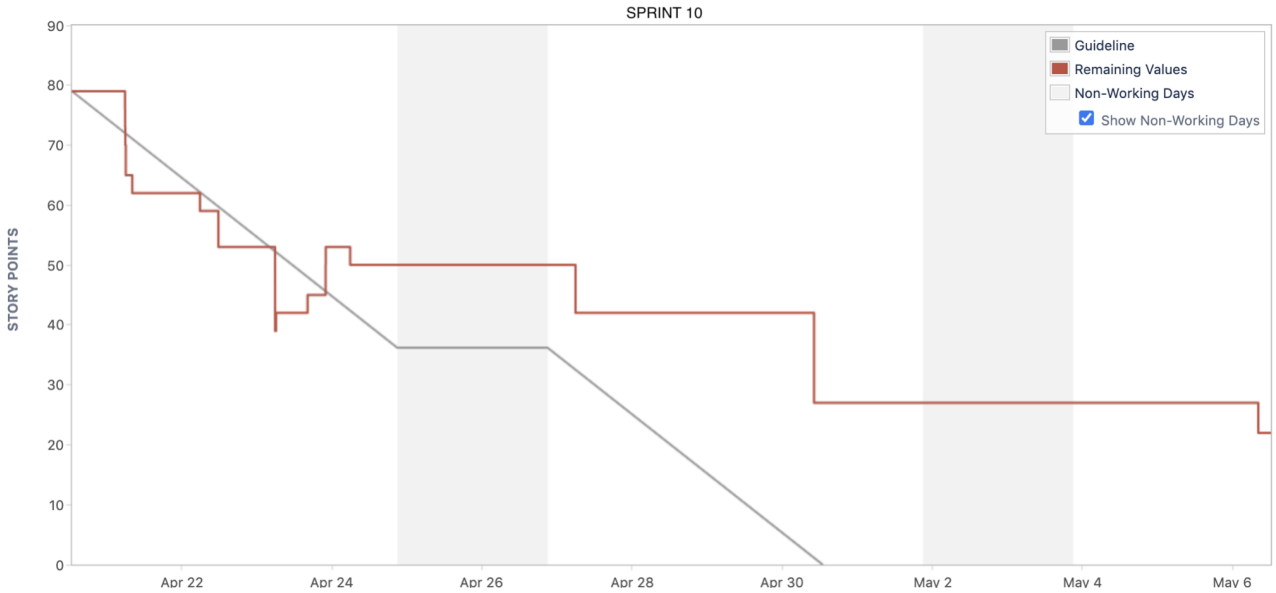

SPRINT 5
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
STORY POINTS
60
50
40
30
20
10
0
Feb 12
Feb 14
Feb 16
Feb 18
Feb 20
Feb 22
Feb 24
TIME

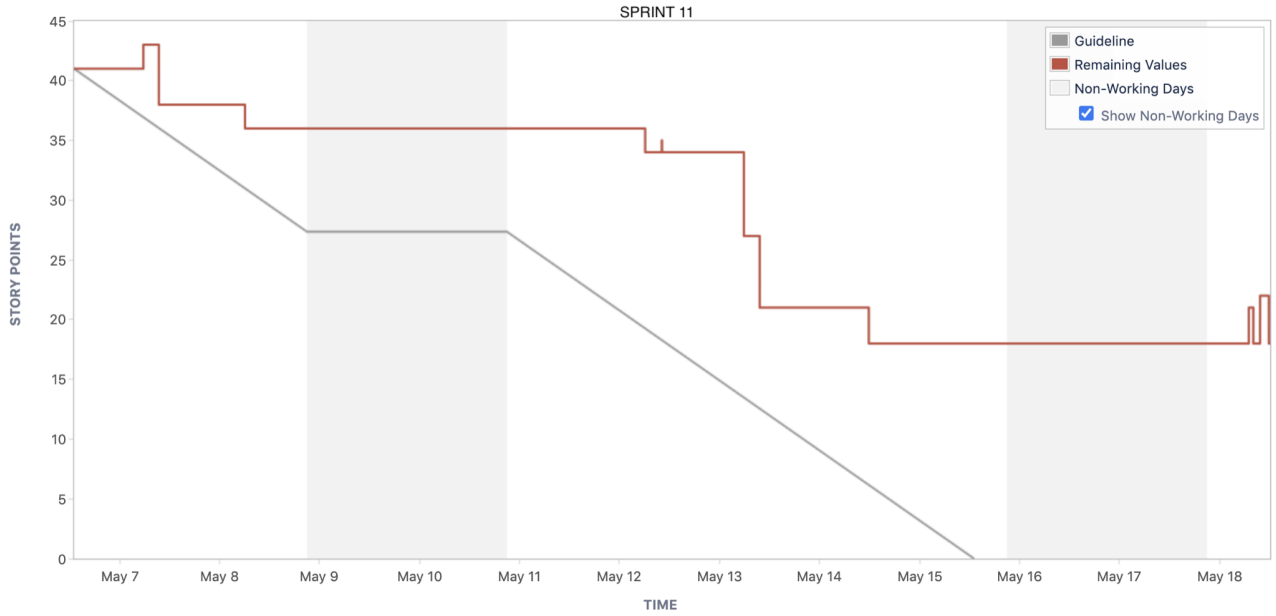

SPRINT 6
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
STORY POINTS
50
45
40
35
30
25
20
15
10
5
0
Feb 26
Feb 27
Feb 28
Feb 29
Mar 1
Mar 2
Mar 3
Mar 4
Mar 5
Mar 6
Mar 7
Mar 8
Mar 9
Mar 10
TIME

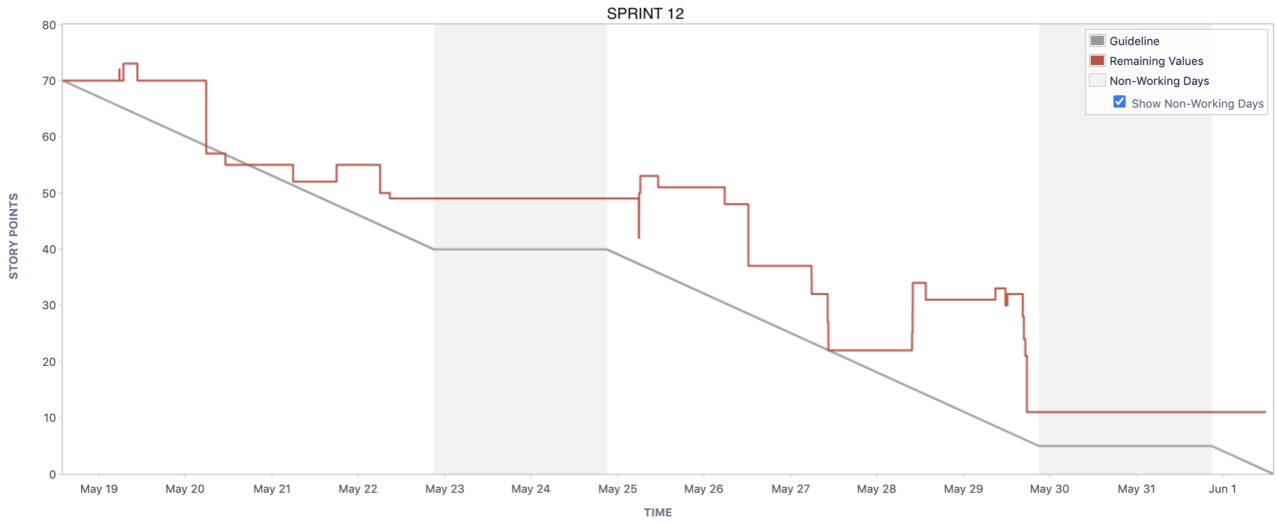

SPRINT 7
STORY POINTS
TIME
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
0
10
20
30
40
50
60
70
Mar 11
Mar 12
Mar 13
Mar 14
Mar 15
Mar 16
Mar 17
Mar 18
Mar 19
Mar 20
Mar 21
Mar 22
Mar 23

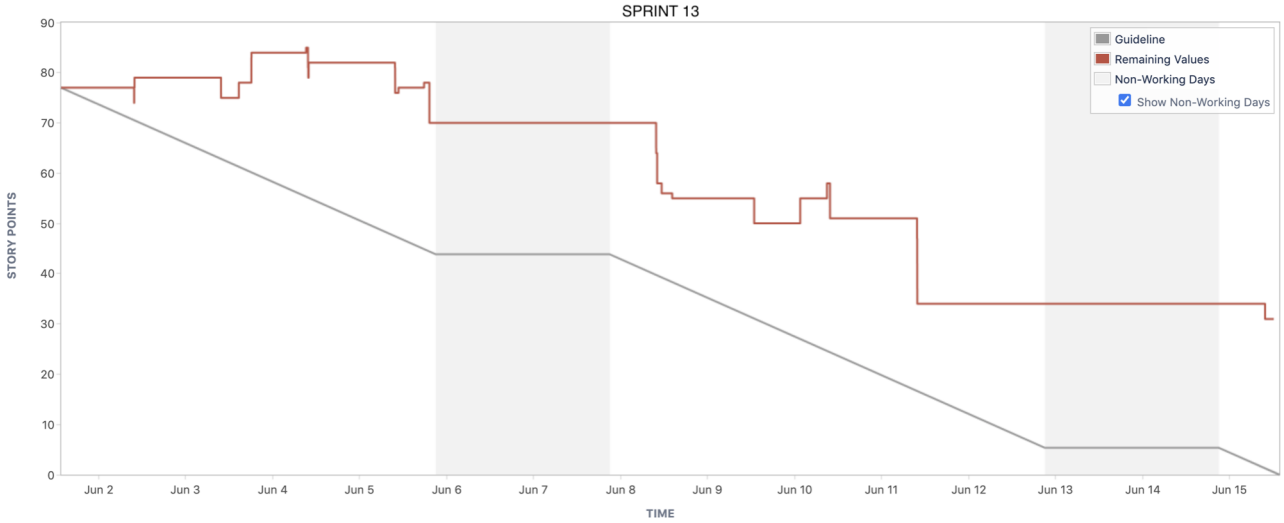

SPRINT 8
STORY POINTS
TIME
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
0
10
20
30
40
50
60
70
Mar 24
Mar 25
Mar 26
Mar 27
Mar 28
Mar 29
Mar 30
Mar 31
Apr 1
Apr 2
Apr 3
Apr 4
Apr 5
Apr 6

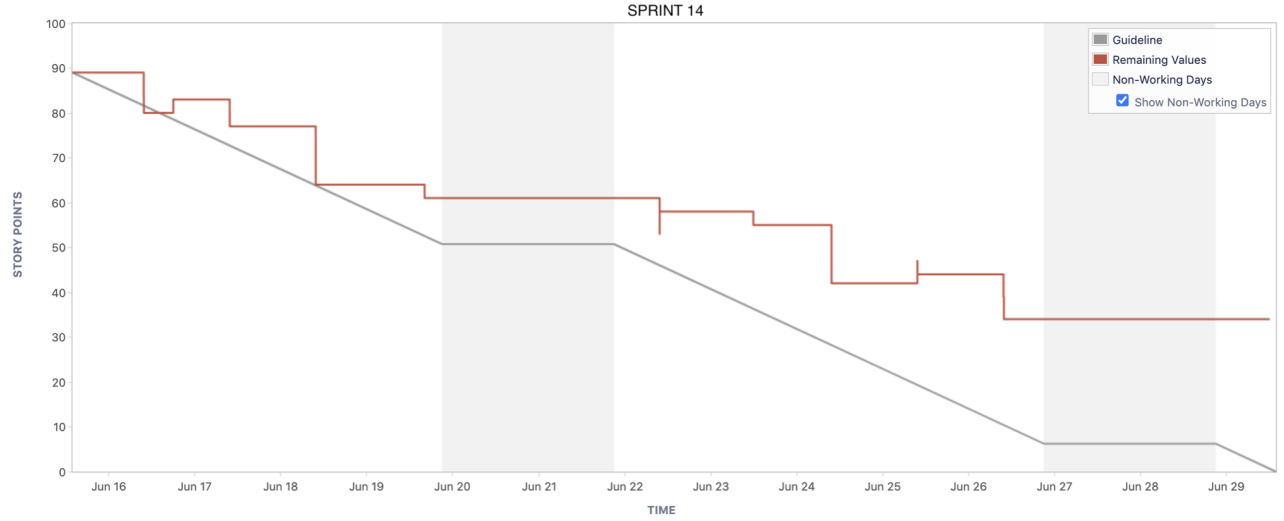

SPRINT 9
STORY POINTS
TIME
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
0
10
20
30
40
50
60
70
80
Apr 7
Apr 8
Apr 9
Apr 10
Apr 11
Apr 12
Apr 13
Apr 14
Apr 15
Apr 16
Apr 17
Apr 18
Apr 19
Apr 20

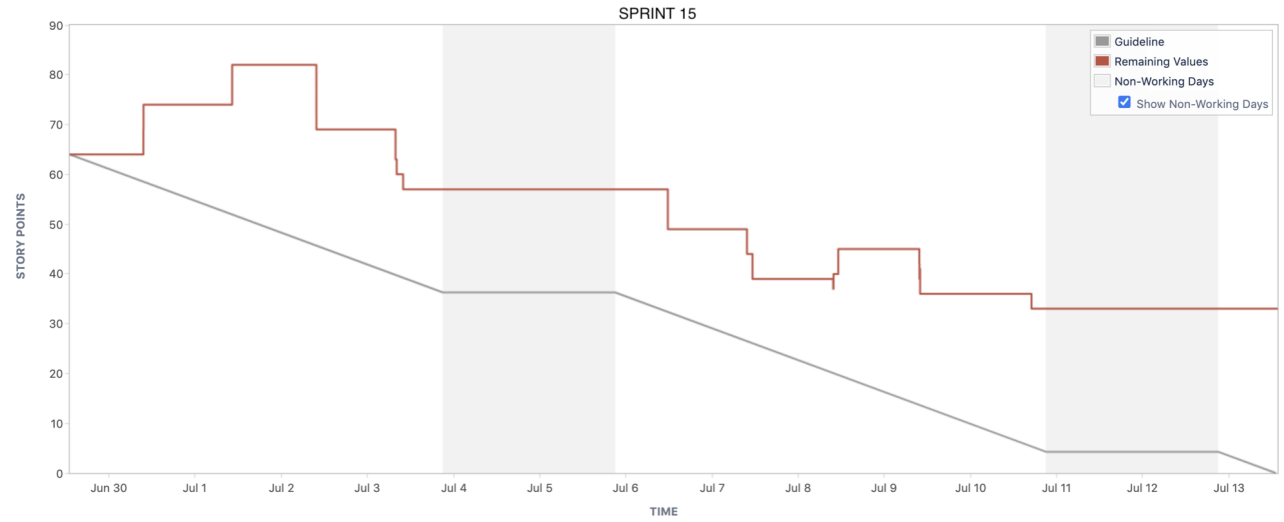

SPRINT 10
STORY POINTS
90
80
70
60
50
40
30
20
10
0
Apr 22
Apr 24
Apr 26
Apr 28
Apr 30
May 2
May 4
May 6
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days

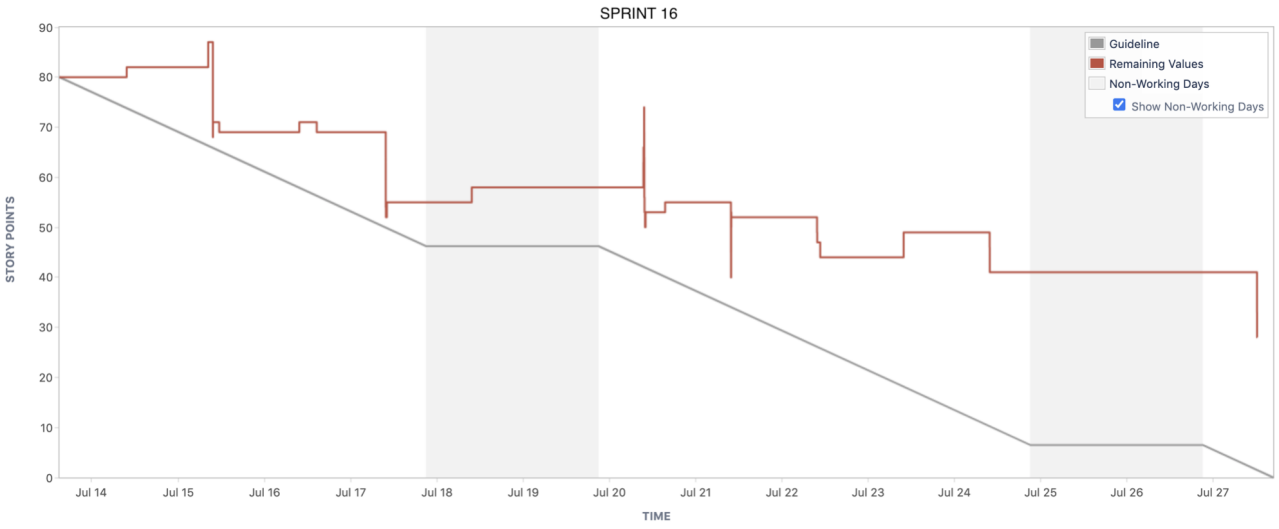

SPRINT 11
STORY POINTS
45
40
35
30
25
20
15
10
5
0
May 7
May 8
May 9
May 10
May 11
May 12
May 13
May 14
May 15
May 16
May 17
May 18
TIME
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days

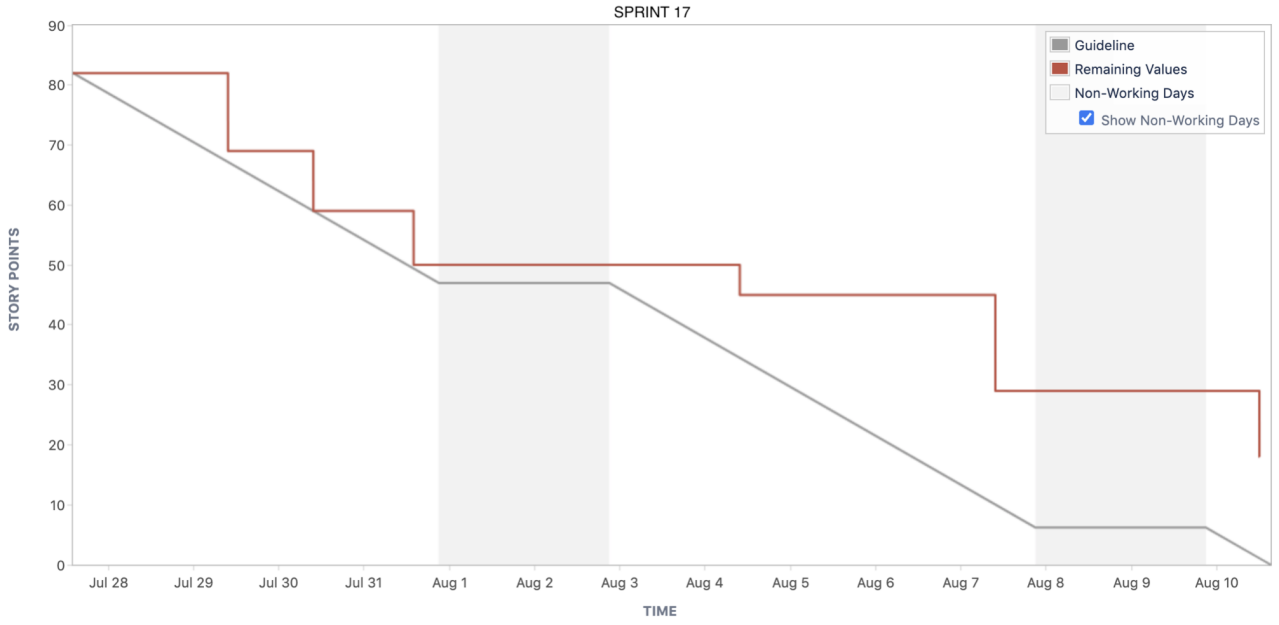

SPRINT 12
STORY POINTS
80
70
60
50
40
30
20
10
0
May 19
May 20
May 21
May 22
May 23
May 24
May 25
May 26
May 27
May 28
May 29
May 30
May 31
Jun 1
TIME
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days

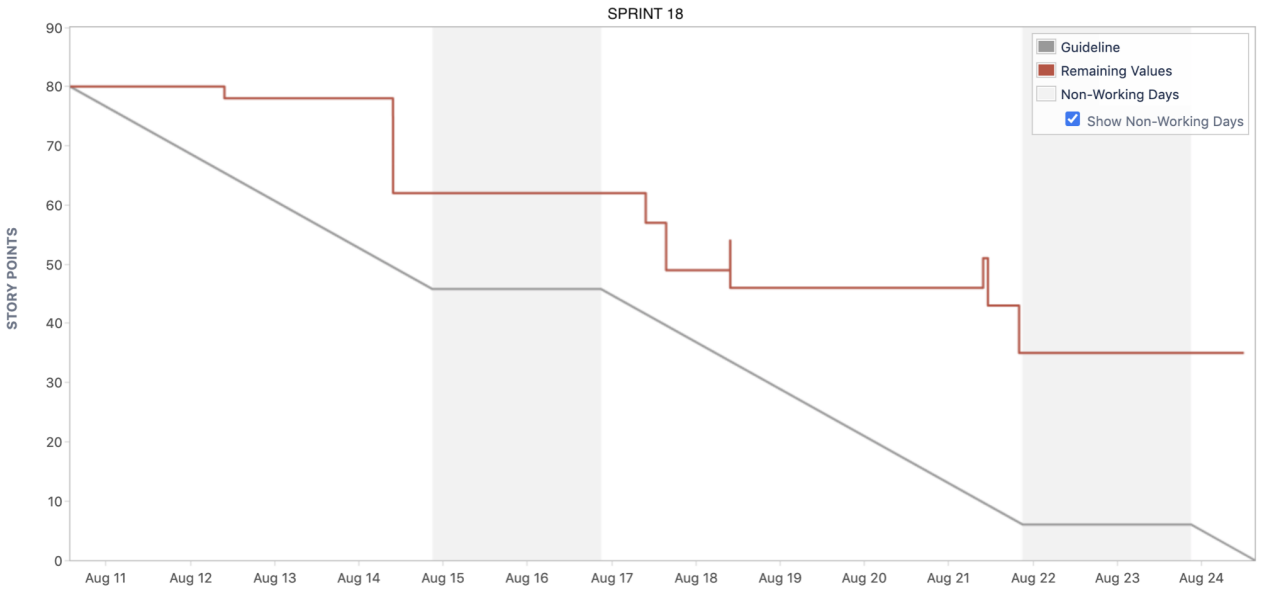

SPRINT 13
STORY POINTS
TIME
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
0
10
20
30
40
50
60
70
80
90
Jun 2
Jun 3
Jun 4
Jun 5
Jun 6
Jun 7
Jun 8
Jun 9
Jun 10
Jun 11
Jun 12
Jun 13
Jun 14
Jun 15

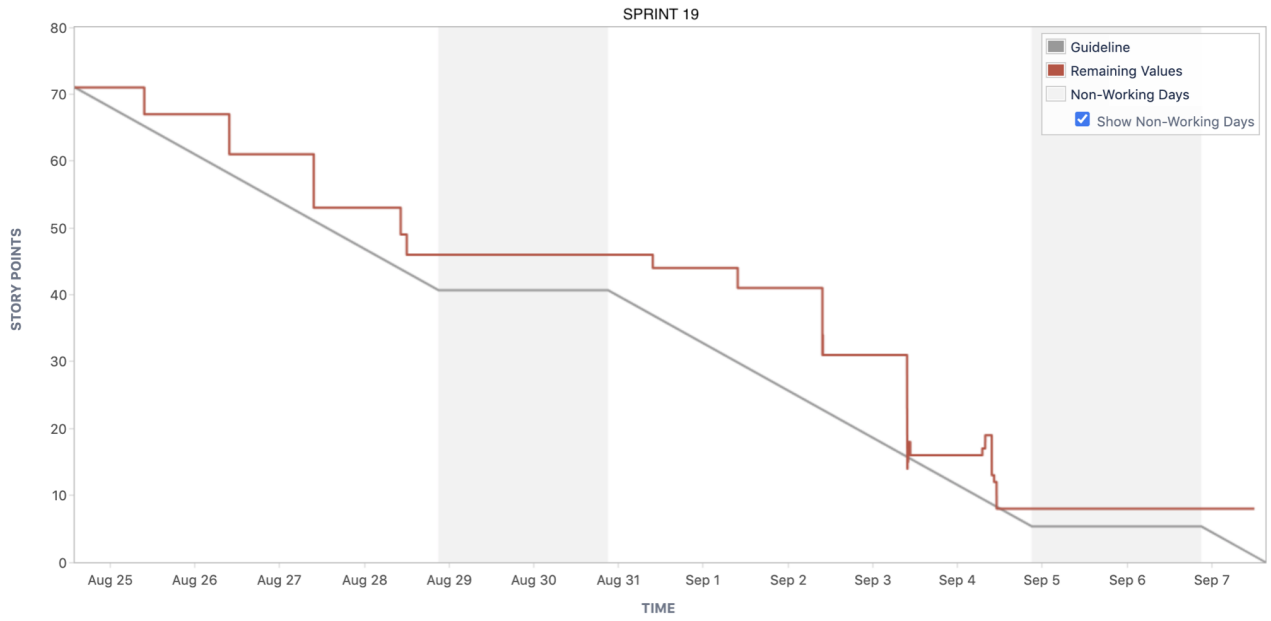

SPRINT 14
STORY POINTS
TIME
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
0
10
20
30
40
50
60
70
80
90
100
Jun 16
Jun 17
Jun 18
Jun 19
Jun 20
Jun 21
Jun 22
Jun 23
Jun 24
Jun 25
Jun 26
Jun 27
Jun 28
Jun 29

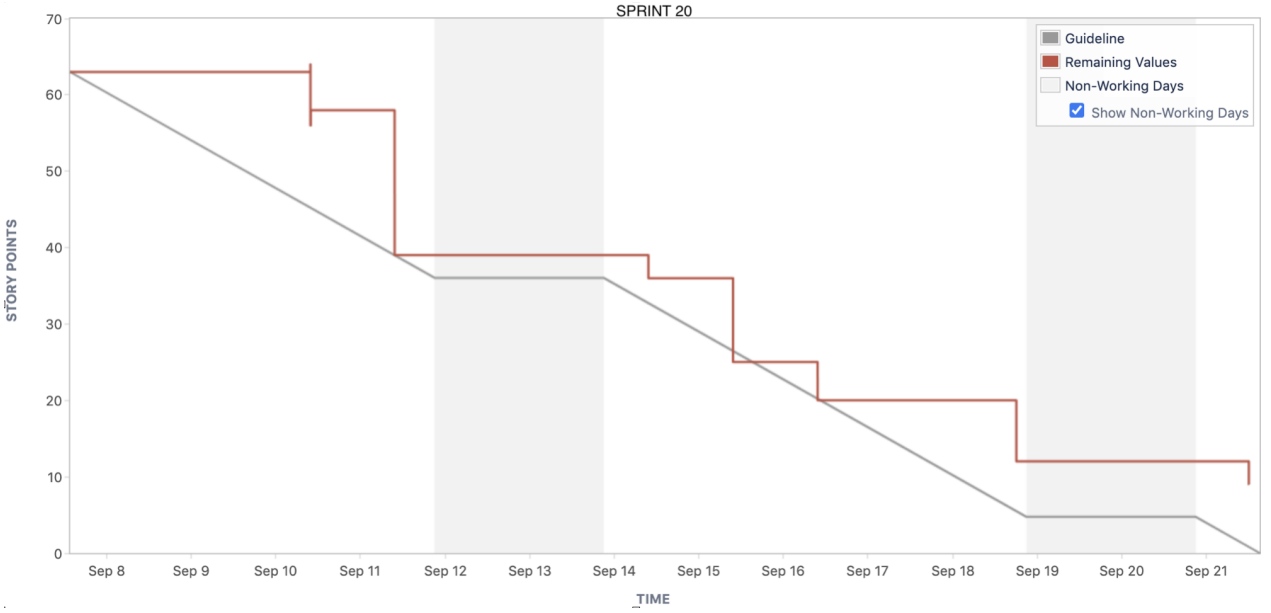

SPRINT 15
STORY POINTS
TIME
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
0
10
20
30
40
50
60
70
80
90
Jun 30
Jul 1
Jul 2
Jul 3
Jul 4
Jul 5
Jul 6
Jul 7
Jul 8
Jul 9
Jul 10
Jul 11
Jul 12
Jul 13

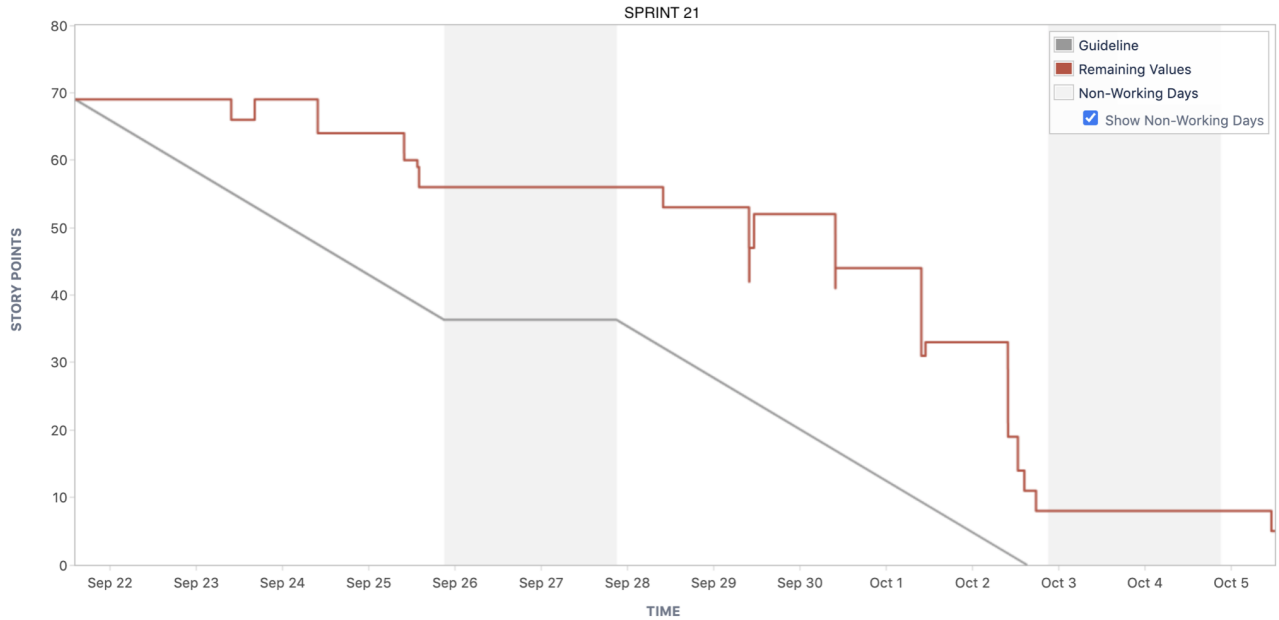

SPRINT 16
STORY POINTS
TIME
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
0
10
20
30
40
50
60
70
80
90
Jul 14
Jul 15
Jul 16
Jul 17
Jul 18
Jul 19
Jul 20
Jul 21
Jul 22
Jul 23
Jul 24
Jul 25
Jul 26
Jul 27

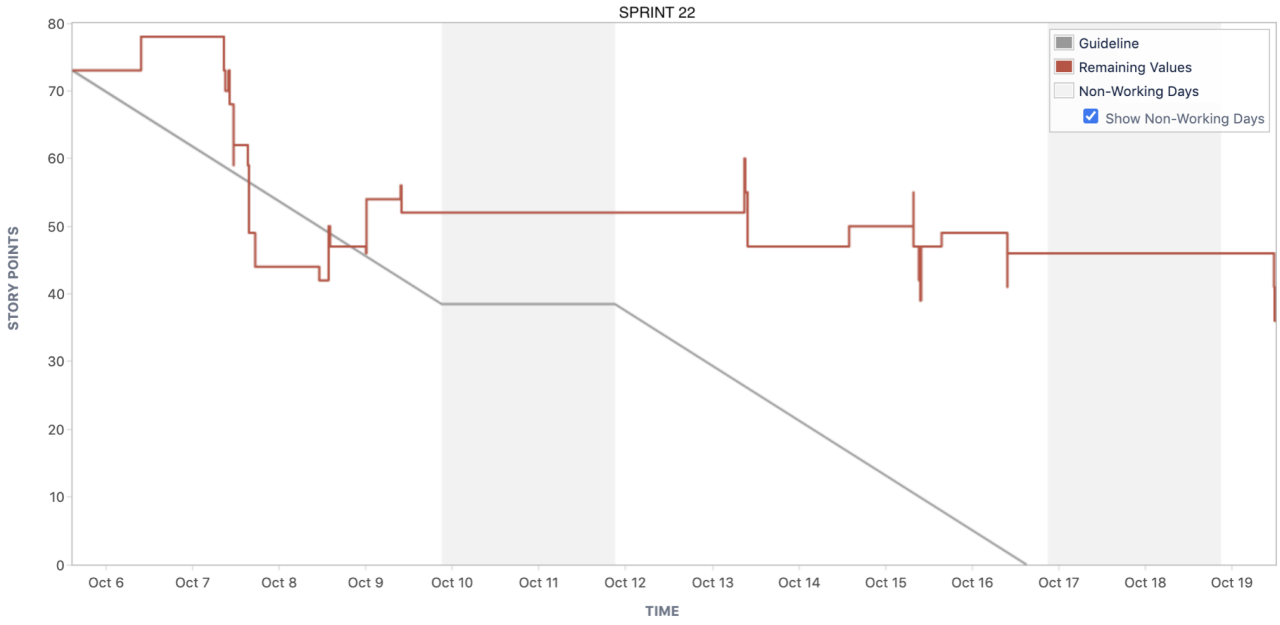

SPRINT 17
STORY POINTS
TIME
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
0
10
20
30
40
50
60
70
80
90
Jul 28
Jul 29
Jul 30
Jul 31
Aug 1
Aug 2
Aug 3
Aug 4
Aug 5
Aug 6
Aug 7
Aug 8
Aug 9
Aug 10

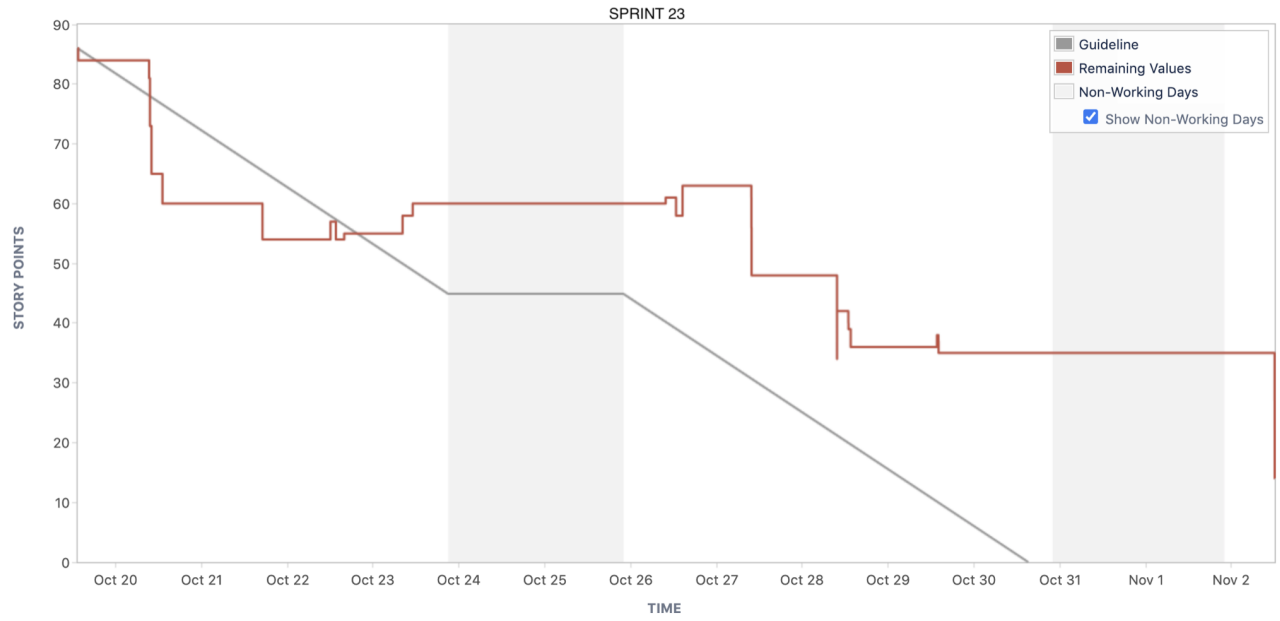

SPRINT 18
STORY POINTS
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
0
10
20
30
40
50
60
70
80
90
Aug 11
Aug 12
Aug 13
Aug 14
Aug 15
Aug 16
Aug 17
Aug 18
Aug 19
Aug 20
Aug 21
Aug 22
Aug 23
Aug 24

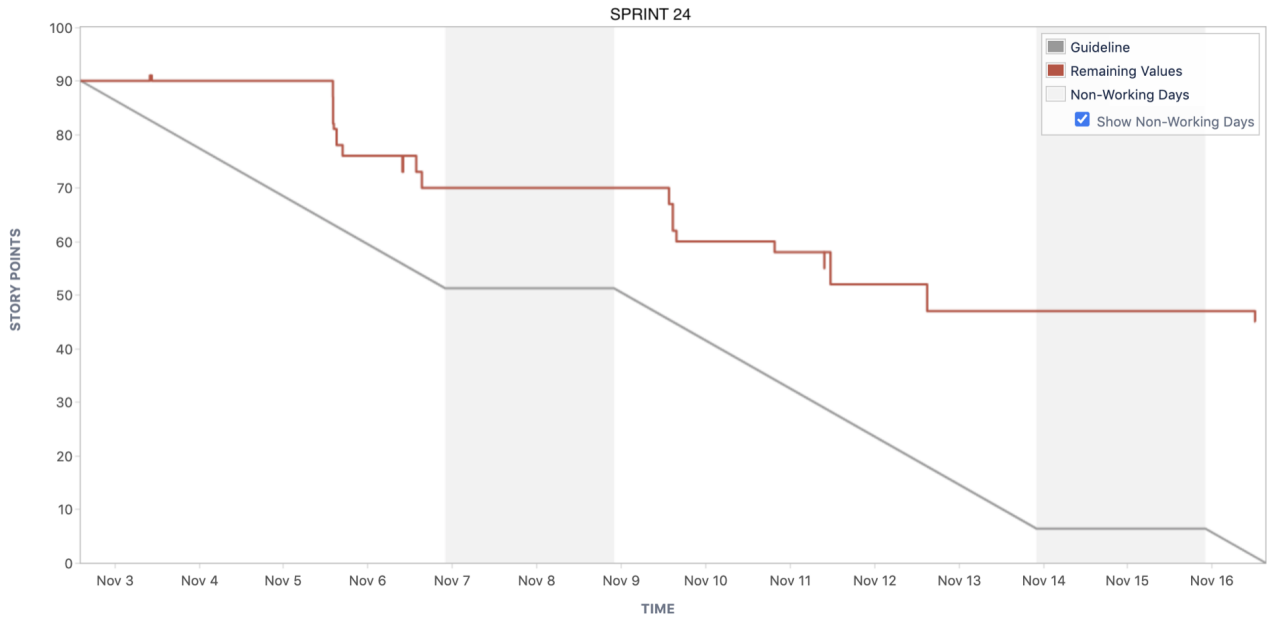

SPRINT 19
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
STORY POINTS
0
10
20
30
40
50
60
70
80
Aug 25
Aug 26
Aug 27
Aug 28
Aug 29
Aug 30
Aug 31
Sep 1
Sep 2
Sep 3
Sep 4
Sep 5
Sep 6
Sep 7
TIME

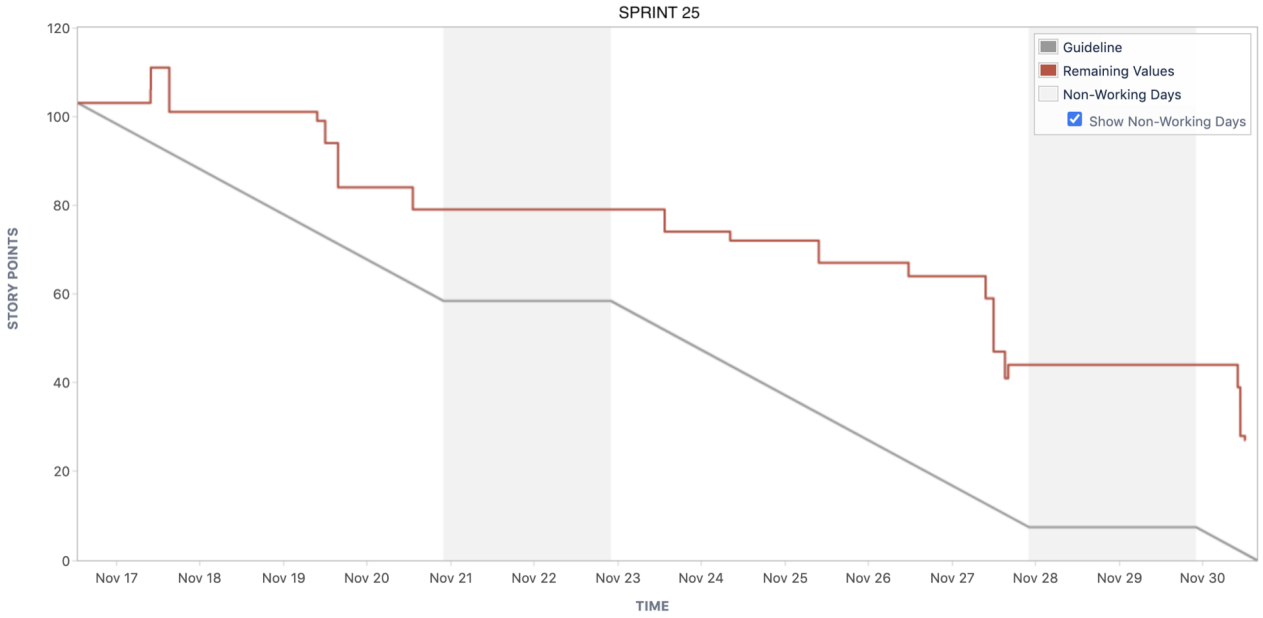

SPRINT 20
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
STORY POINTS
0
10
20
30
40
50
60
70
Sep 8
Sep 9
Sep 10
Sep 11
Sep 12
Sep 13
Sep 14
Sep 15
Sep 16
Sep 17
Sep 18
Sep 19
Sep 20
Sep 21
TIME

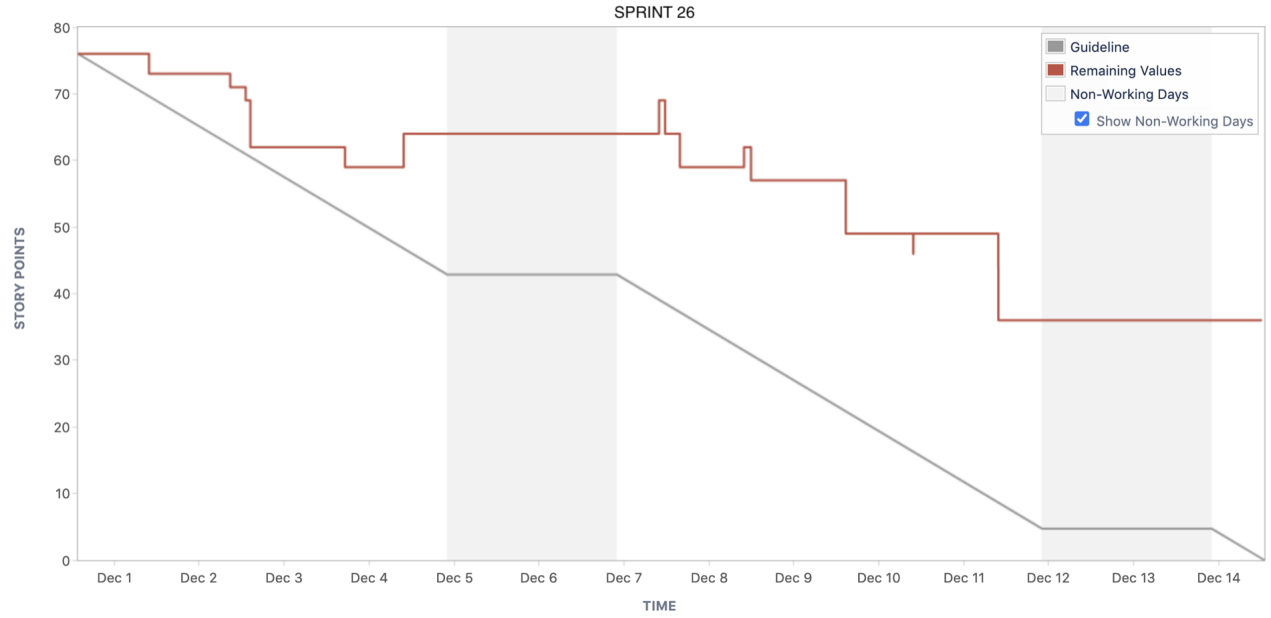

SPRINT 21
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
STORY POINTS
0
10
20
30
40
50
60
70
80
Sep 22
Sep 23
Sep 24
Sep 25
Sep 26
Sep 27
Sep 28
Sep 29
Sep 30
Oct 1
Oct 2
Oct 3
Oct 4
Oct 5
TIME

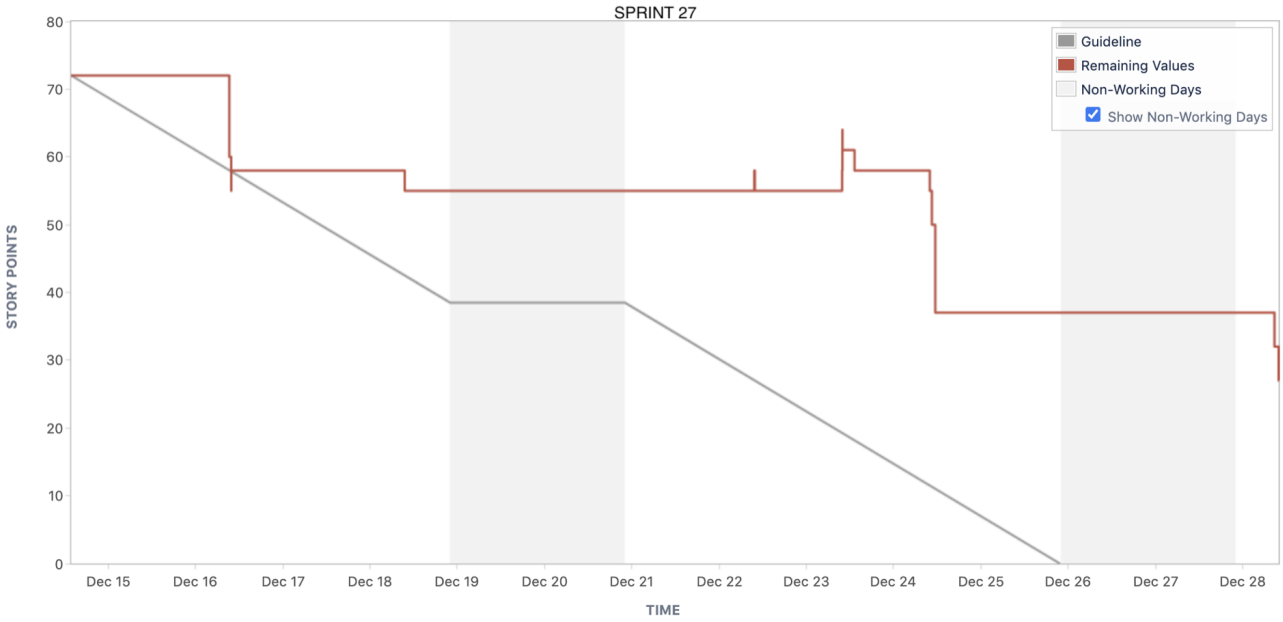

SPRINT 22
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
STORY POINTS
0
10
20
30
40
50
60
70
80
Oct 6
Oct 7
Oct 8
Oct 9
Oct 10
Oct 11
Oct 12
Oct 13
Oct 14
Oct 15
Oct 16
Oct 17
Oct 18
Oct 19
TIME


SPRINT 23
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
STORY POINTS
0
10
20
30
40
50
60
70
80
90
Oct 20
Oct 21
Oct 22
Oct 23
Oct 24
Oct 25
Oct 26
Oct 27
Oct 28
Oct 29
Oct 30
Oct 31
Nov 1
Nov 2
TIME

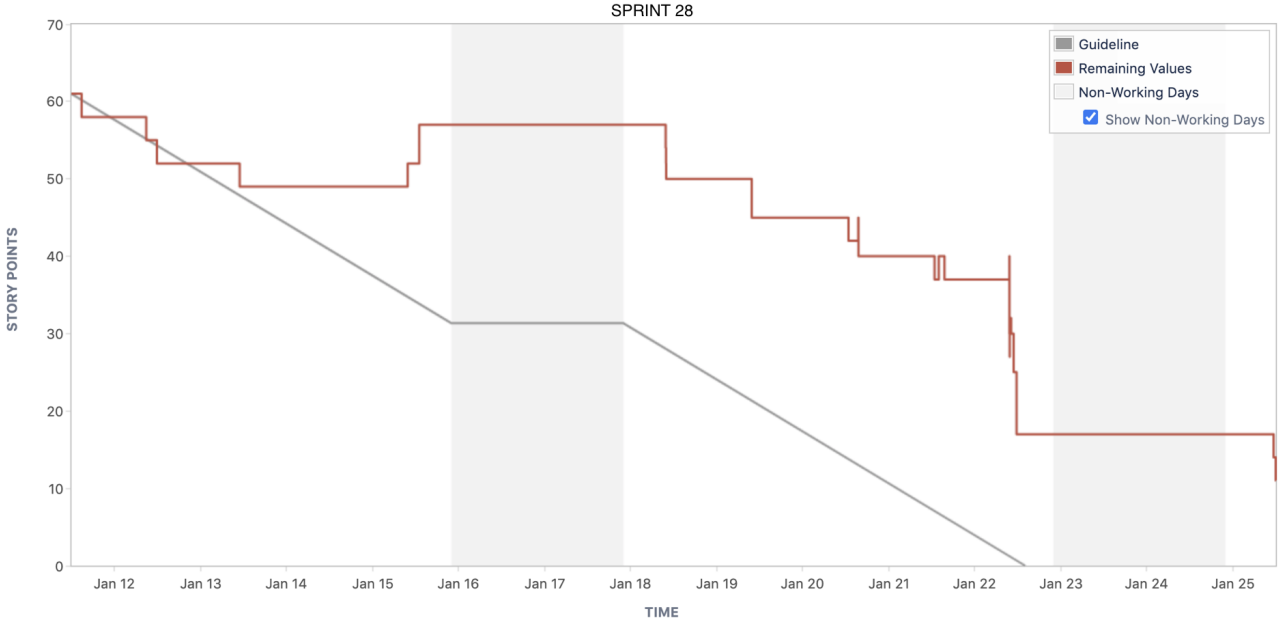

SPRINT 24
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
STORY POINTS
0
10
20
30
40
50
60
70
80
90
100
Nov 3
Nov 4
Nov 5
Nov 6
Nov 7
Nov 8
Nov 9
Nov 10
Nov 11
Nov 12
Nov 13
Nov 14
Nov 15
Nov 16
TIME


SPRINT 25
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
STORY POINTS
0
20
40
60
80
100
120
Nov 17
Nov 18
Nov 19
Nov 20
Nov 21
Nov 22
Nov 23
Nov 24
Nov 25
Nov 26
Nov 27
Nov 28
Nov 29
Nov 30
TIME


SPRINT 26
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
STORY POINTS
0
10
20
30
40
50
60
70
80
Dec 1
Dec 2
Dec 3
Dec 4
Dec 5
Dec 6
Dec 7
Dec 8
Dec 9
Dec 10
Dec 11
Dec 12
Dec 13
Dec 14
TIME


SPRINT 27
Guideline
Remaining Values
Non-Working Days
Show Non-Working Days
STORY POINTS
0
10
20
30
40
50
60
70
80
Dec 15
Dec 16
Dec 17
Dec 18
Dec 19
Dec 20
Dec 21
Dec 22
Dec 23
Dec 24
Dec 25
Dec 26
Dec 27
Dec 28
TIME

SPRINT 28



SPRINT 29