SILESIAN UNIVERSITY OF TECHNOLOGY PUBLISHING HOUSE

SCIENTIFIC PAPERS OF SILESIAN UNIVERSITY OF TECHNOLOGY 2025
ORGANIZATION AND MANAGEMENT SERIES NO. 215

PLANNING IMPROVEMENTS BASED ON AGILE REQUIREMENTS ANALYSIS FOR SOFTWARE PROJECTS WITH UNCERTAIN REQUIREMENTS

Ewa MARCHWICKA[1*], Tomasz CYBULSKI[2]

[1] Wroclaw University of Science and Technology; ewa.marchwicka@pwr.edu.pl,
ORCID: 0000-0001-6097-784X
[2] tomcyb27@gmail.com
* Correspondence author

Purpose: The objectives of the paper are to identify agile software projects with uncertain requirements that require planning adjustments and to suggest planning improvements that can be applicable in similar project contexts.
Design/methodology/approach: Semi-structured interviews, literature review and documentation analysis were used as research methods. 509 different IT projects of software outsourcing companies were analyzed in the terms of exceeding estimated time.
Findings: As a result, one project type has been indicated as underestimated. A planning approach incorporating user story prioritization, based on the Kano model has been suggested. Prioritizing project tasks if often used in Agile context, but prioritization process is usually very subjective and performed only by one person (Product Owner). The approach presented in this paper suggests to extend prioritizing techniques to all parties that might be interested (like team members or stakeholders) by collecting questionnaire answers. The process can be done iteratively, which is characteristic for Agile.
Research limitations/implications: The limitation is the superficial analysis of the quality of the approach. Only a general scheme has been presented. Performing detailed quality analysis is planned as future research.
Practical implications: Improving standard prioritization techniques should be valuable for Agile practitioners, because it means better planning. In particular it helps to quicker develop a Minimal Viable Product (MVP) and achieve users' satisfaction. The research indicates one particular Agile project type, for which this approach should be helpful.
Originality/value: According to the literature review research performed by the authors, this paper fills in the gap in the literature of mid-term planning in Agile context.
Keywords: project planning, Agile approach, Kano model, prioritization.
Category of the paper: research paper.

1. Introduction

The dynamic growth in demand for software and the in-creasing digitization of society are having a significant impact on organizations involved in IT projects. The rapidly changing business environment is driving more and more companies to adopt Agile approaches in project management to increase their flexibility and enhance their competitiveness in the market. Classical project management methodologies (also referred to as waterfall) assume fixed project scope that is the input of long-term planning, which tries to answer how long it would take to implement the given set of requirements. On the other hand, Agile methodologies promote adapting project scope to a fixed timeframe of one Sprint (Kosztyan et al., 2020). But this short-term Agile planning cannot guarantee the success of some outsourcing projects (Krancher, 2020). Outsourcing IT projects means hiring an external vendor to handle IT tasks or entire projects, leveraging their expertise and reducing costs (Nicolas et al., 2018).

This study focuses on projects with uncertain requirements, realized by outsourcing companies for which planning improvements are needed. The objective of this paper is twofold. The first objective is to identify outsourcing company project types that need planning improvements and the second one is to suggest and discuss potential planning improvements. What is more, a sample method based on Kano model is presented that allows to incorporate the discussed improvements. The method development is not the objective of this paper, but it has been presented to demonstrate how easy is to apply the improvements.

The Software Extension to the PMBOK® Guide defines software projects as endeavors that involve creating new software products, modifying existing software, integrating multiple software components, expanding software capabilities, or modifying software infrastructure within an organization. Furthermore, software projects can be initiated to address service requests, meet software maintenance needs, or provide operational support. These activities are considered projects if they are defined as temporary undertakings aimed at delivering specific outcomes (PMI, 2013).

The article is structured as follows: Section 2 presents a literature review, Section 3 describes the research methods used to achieve the objectives of the study, Section 4 presents the obtained results, including the identification of issues in IT project planning and an analysis of the underlying causes of these problems. Section 5 discusses potential improvements aimed at addressing these issues. Finally, Section 6 summarizes the conclusion of the study.

2. Literature review

Below a literature review in the areas related to the topic of this articles are presented. The related areas are IT-project planning, effort estimation techniques, prioritization techniques, Kano model and IT project planning.

2.1. IT-Project planning

When using Agile approach, only minimal planning is promoted (Fernandez-Diego et al., 2020). Simultaneously, in Agile approach only Sprint planning horizon is usually considered, and mid-term planning seems to be neglected in the literature. But mid-term planning can still be an important topic in many IT projects (Kosztyan, 2022; Kosztyan et al., 2023), also including the Agile ones or in the projects that use hybrid approach (Kosztyan et al., 2020).

2.2. Effort estimation techniques

No matter in which context (Agile, hybrid or classic) project planning is considered and no matter what planning horizon is the focus of the research (short term or mid-term), effort estimation techniques (Fernandez-Diego, 2020) are required and very important in all project planning applications and methods. The following estimation techniques can be listed: planning poker, expert judgement, wideband Delphi, machine learning, neural network, functional size measurement, regression, algorithmic methods, fuzzy logic, swarm intelligence, Bayesian network, Monte Carlo, Statistical combination, Principal Component Analysis, COCO-MO II, use case point, change effort prediction, ontology model, experience factory, stories prioritization. It should be noted that uncertainty is a common characteristic of estimation (because of many factors, including risks that may materialize during project execution). To handle un-certainty, it is a common practice (Shirazi et al., 2017) and common Agile practice (Cao et al., 2010), to make scope changes as the project progress-es. In Agile it is the time not the scope that should be fixed (Kosztyan, 2020). Scope changes may include adjusting the scope of single user stories, but it can also mean changes made on the project level, when some of the non-important user stories need to be excluded from the scope (Kosztyan, 2020). This is when prioritization techniques come into play.

2.3. Prioritization techniques

Requirements techniques are very popular in Agile planning context (Kosztyan et al., 2020; Kosztyan, 2022; Kosztyan et al., 2023), including the context of global software development (Ali, Lai, 2021). Some popular examples of prioritization include Must-Should-Could-Would Technique (MoSCoW), Analytic Hierarchy Process (AHP), Binary-Search Tree (BST), Voting Techniques and Kano model (Saher et al., 2018).

2.4. Kano model and IT project planning

Kano model is a popular model for assessing the quality of products and services and it appears most frequently in the context of quality (Vavpotic et al., 2019; Golrizgasthi et al., 2019). What is more, Kano model is often cited in reference to product development (Francisco, SantAnna, 2019; Li, Zhang, 2021) and requirements engineering (Campese, Hornos da Costa, 2019). What was already mentioned above, there are Kano model applications to requirements prioritization (Saher et al., 2018). Some other literature positions include applications to minimal viable product development (Lee, Geum, 2021).

No literature positions were found that incorporate Kano model for IT project planning. To the best knowledge of authors this paper is the first one that discusses Kano model in the context of IT project planning. Of course, there is an indirect reference from Kano to IT project planning as Kano is listed as one of the prioritization techniques used for Agile planning (which was described above), but this paper presents an original approach for Kano model representation and incorporates an original and dedicated questionnaire for assessing Kano model quality categories. Additionally, no literature positions were found that consider Kano model in the context of long-term Agile planning.

3. Methods

3.1. Research methods

The research methods used to achieve the objectives of this study included analyzing documentation related to historical data from completed IT projects of outsourcing companies and conducting semi-structured interviews with project managers of the analyzed projects. Another research method that was used was literature reviews.

3.2. Research design

The documentation was used in the problematic projects' identification process. Semi-structured interviews combined with the documentation analysis were used in the analysis of the problem causes. The results of this analysis, combined with literature reviews were used for developing potential planning improvements.

The research design utilized in this work has been visualized in Figure 1.

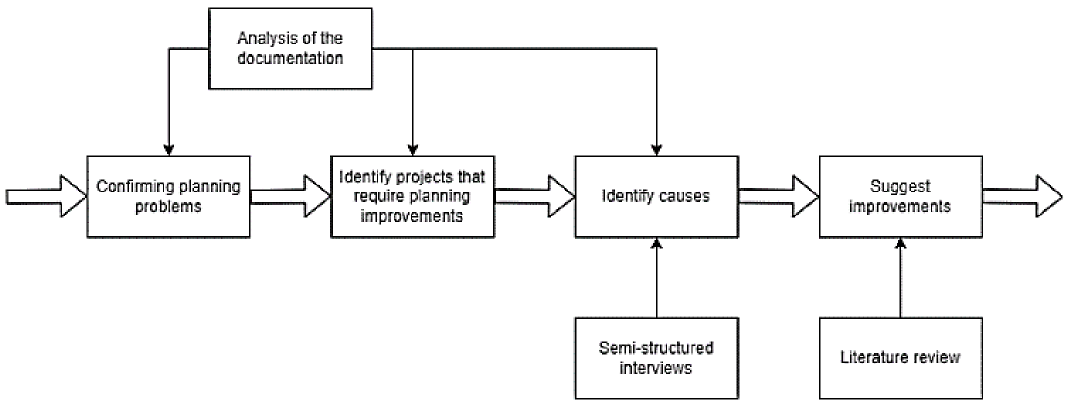
Analysis of the documentation
Confirming planning problems
Identify projects that require planning improvements
Identify causes
Suggest improvements
Semi-structured interviews
Literature review

Figure 1. Research design.

The results of the first three steps will be described in the Section 4.

3.3. Data collection

A total of 509 IT projects of outsourcing companies were collected, analyzed and categorized into different types, based on requirement types. The classification presented in this paper not a standard one that can be found in the literature and has been extracted from the available projects' documentation. This classification reveals the wide spectrum of IT projects that can be realized in outsourcing companies. There are many other classifications that can be suggested here but are not considered. For example (Ghodeswar, Vaidyanathan, 2008) presents a classification that is based on business processed. Figure 2 illustrates the distribution of the number of completed projects by project type.

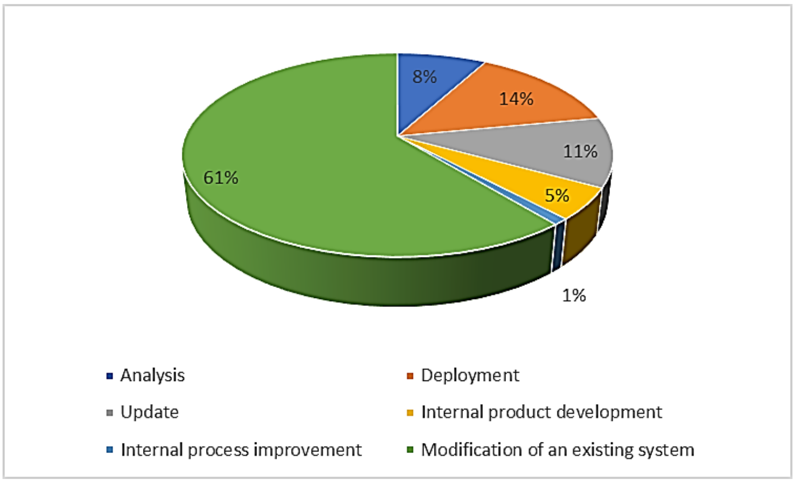

Figure 2. Research design.

Projects of the "Analysis" type aim to gather the business requirements of the client. The end result of projects in this category is the development of a functional specification. For new clients, the analysis is conducted with the purpose of later implementing the system at the client's location. The "Modification of an existing system" projects is connected either with modification of the existing system configuration or client-oriented product development. Another type of project is "Internal process improvement". These projects implement enhancements to the internal company processes, including automation of parts or the entire process, and the implementation of new tools to facilitate customer support or collaboration between teams. "Internal product development" projects aim to deliver additional functionality in the offered product based on common industry-specific customer requirements. The goal of internal product development projects is to increase future sales by meeting the shared demands of potential clients and retaining existing clients. "Update" projects are intended to implement system updates for a specific client. On the other hand, "Deployment" projects aim to carry out the implementation of the production system for the client. They are preceded by prior analysis.

The dataset contained projects realized in outsourcing companies for different industries. The industries included: automotive, customer services, logistics, energy sector and public sector.

Although in the great number of projects Agile methodologies were declared to be used, more than 90% of the projects were fixed-price projects, where costs (and also the time) were defined in project contract. It means that these kinds of projects required prior time estimations before they could be started, which is the characteristic of waterfall approach. It also means that project estimations data were available for the projects that were analyzed.

4. Results

4.1. Confirming planning problems

To confirm problems in the area of IT project planning in outsourcing companies and later analyzing their potential causes, the preliminary stage involves confirming that planning problems indeed occur. It is realized by analyzing data related to the difference between estimated and actual project durations. These data are visualized using a histogram. The histogram bins are assigned with a granularity of 20%. The results are presented in Figure 3.

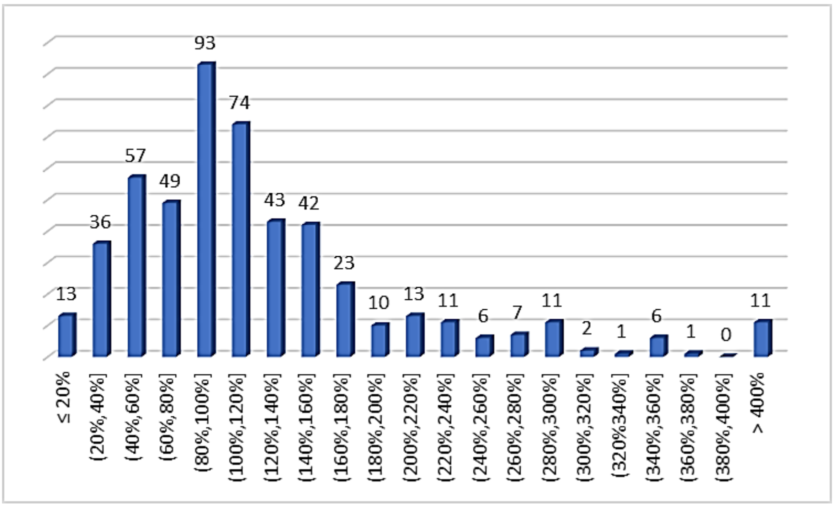

Figure 3. All projects – histogram depicting differences between real and estimated duration.

The obtained results indicate that the actual durations are consistent with the estimates for the largest group of projects, with a deviation of up to +/-20% of the total project duration considered an acceptable margin (two bins that are placed around the reference value of 100%). However, this group represents only 33% of all completed projects. The remaining projects were either underestimated or overestimated. The detailed distribution is presented in Figure 4.

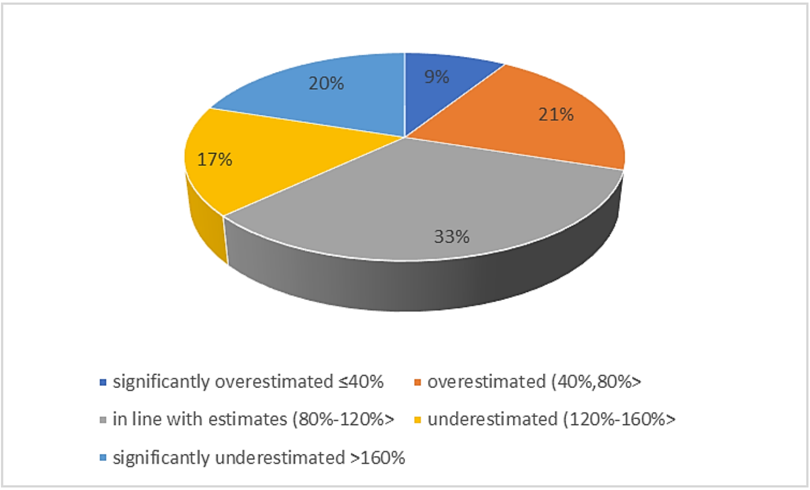

Figure 4. All projects – distribution of overestimated and underestimated projects.

4.2. Identifying project types that need planning improvements

To identify project types that need planning improvements, the standard deviation of the estimated project durations is calculated for each type of project undertaken. The results of the comparison are presented in Table 1. Based on these results, two types of projects were identified with the largest deviations between the actual project durations and the initially estimated durations – internal process improvement projects and internal product development projects. Since internal process improvement projects accounted for approximately 1% of all projects analyzed (Figure 2), the focus is on further analyzing internal product development projects. The presented data suggests that internal process improvement projects are not given the appropriate planning, which can negatively impact their chances of success and achieving their intended objectives.

Table 1.
Basic characteristics of differences between real and estimated durations grouped by project type

Project type	Actual average duration of the project as a percentage of the initial estimated time	Standard deviation
Analysis	105%	0,61
Modification of an existing system	116%	0,88
Deployment	144%	0,96
Internal process improvement	28%	0,13
Internal product development	230%	4,19
Update	166%	1,84
Final average	130%	1,41

The data indicates that the actual duration of internal product development projects averaged at 230% of the initially estimated duration and exhibited the highest variability among all types of projects undertaken. Figure 5 illustrates a histogram depicting the distribution of differences between the actual and initially estimated project durations for internal product development projects.

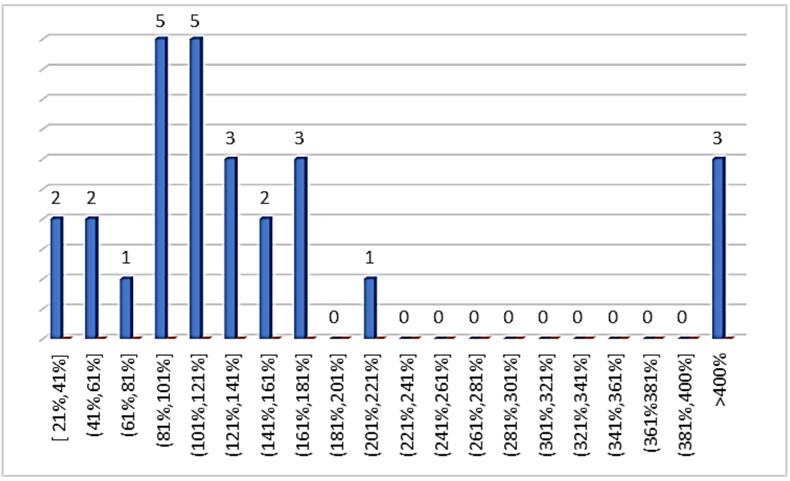

Figure 5. Internal product development projects – histogram depicting differences between real and estimated duration.

Next, the differences for this type of projects are divided into groups and presented in Figure 6.

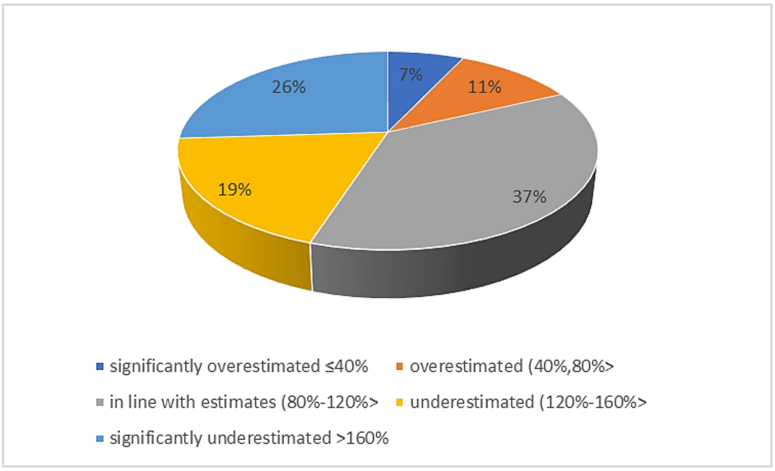

Figure 6. Internal product development projects - distribution of overestimated and underestimated projects.

When comparing the distribution of actual project durations for internal product development projects to the distribution for all projects, it should be noted that despite a higher percentage of projects completed in line with initial estimates, the percentage of highly underestimated projects has increased (red color, Figure 6). Based on the above analysis, a planning problem has been identified for internal product development projects.

4.3. Analysis of the causes of the problem in project planning area

The identified problem was further analyzed to determine its causes. For this purpose, the documentation of selected completed projects was reviewed in detail, and semi-structured interviews were conducted with project managers. Based on this, the causes related to the problem were identified.

One of the causes of the problem that was frequently indicated by project managers is the variability of requirements during project execution. This is an inherent challenge in IT projects because early requirement analysis cannot fully determine the software users' needs.

In the case of internally developed product development projects, the planning phase involves collecting and analyzing requirements, which are then used for implementing novel functionalities by the programming team. Implemented functionalities are then presented to internal stake-holders who verify the produced software. At this stage, overlooked requirements and defects can be identified, or requirement coverage by implemented functionality can be validated. Since such projects are executed internally (which is in fact not typical for the outsourcing companies), the project duration is mainly constrained by internal decisions within the organizations, creating a tendency to include a large number of new requirements extending the project scope and causing scope creep. It should be noted that in this case, the software is not evaluated by external users, suggesting that some internally identified requirements may not provide business value.

It is also worth noting that each implemented software functionality in the project aims to meet the needs of users defined during the requirements analysis. In the examined internally developed product development projects, it is often mentioned that user requirements are determined based on interviews with software users and their own industry experience.

The problem of ad-hoc determination of the functionalities priorities mentioned above results in lower-priority requirements being implemented earlier than those associated with higher-priority requirements, even though they may have lower business value. As a result, the project duration is extended because most of the necessary functionalities are implemented in a later order.

In the literature there exist some formal methods that can be used to prioritize requirements in projects, for example MoSCoW method, which is a technique used to determine the priority of requirements based on stakeholders' positions. Other popular examples incorporate AHP, BST, voting techniques or Kano model (Saher et al., 2018).

5. Discussion

Based on the conducted analysis that ended up with identified causes for the planning problems and supported by the literature review (Figure 1), some selected planning improvements can be proposed. When developing these improvements, the focus was on general solutions that can be applied in any organization carrying out IT projects. Additionally, the proposed improvements were assumed to be simple and easy to implement as part of existing software development processes. The improvements needed to address the main problems described above, i.e. they need to address requirements uncertainty and need to incorporate the process of selecting user story priorities that includes business value.

5.1. Improvements regarding reducing requirement uncertainty

The problem of requirement uncertainty is a well-known issue in IT projects. In the literature, requirement volatility in a project is often described as an almost inevitable part of IT projects, and it is also reflected in one of the principles mentioned in the Agile Manifesto: "Welcome changing requirements, even late in development" (Beck et al., 2023). The causes of requirement volatility can be identified already during the requirement gathering stage. Communication problems may arise when gathering information about requirements from clients or potential system users by business analysts, as well as during the transfer of requirements to the implementation team, leading to misinterpretation of the conveyed information by one of the parties. Presenting requirements in the form of extensive documentation also does not solve the problem because the recipient may lose important elements related to the requirements amidst the overload of details. Furthermore, each party may rely on existing knowledge that may not be relevant to the current situation. Additionally, some knowledge within organizations may not be formalized, making it difficult to codify requirements derived solely from the context and culture of the organization. Another cause is the temporary nature of requirements described in requirement documents. They only address the requirements specified at the time of document development. If the document is not regularly updated during software implementation, the resulting software will not incorporate requirement changes that occurred between their collection and the delivery of the software. Another factor contributing to requirement volatility is the prior experience of potential software users. If the existing software is being replaced, it is often due to a lack of expected features. Therefore, users expect the new system to include previous features as well as new features that were not available in the previously used software. This leads to the constant expansion of the requirement list throughout the implementation of the new software (Kelly, 2024). To minimize requirement uncertainty, an iterative process is proposed, that allows to update the plan, after development team gains more knowledge about the requirements. It leads to the conclusions that the standard Sprint-based approach can be applied, which is not a great discovery. But it is important is that this iterative approach can be applied not only to the set of requirements that are planned to be implemented in the current Sprint, but to all the requirements available for the project. On the other hand, when time needed for delivering new functionalities is long (which corresponds to the waterfall approach), it is difficult to obtain feedback from end users regarding the usefulness of the introduced changes. Therefore, internal product development projects should be aligned with the Sprint release cycle, allowing for the inclusion of previously developed functionalities in the next software release, rather than incorporating all changes only when the project scope has been completed. When requirements are frequently refined and prioritized (which will be described in the following subsection), it is easier to stop product development or re-negotiate the cost with the client, after project exceeds too much the time that was initially agreed with the client. This perfectly fits in the outsourcing company scenario for the fixed-price project that was described in the Introduction section. The above observations led to the following recommendations that should be considered for an internal product development project:

1. Developing a minimum viable product (MVP) should be the objective of the project from its very beginning. A minimum viable product refers to a product or its part that has a minimal set of functionalities providing value to the end user. Focusing on delivering the MVP helps avoid pro-longed and ultimately unnecessary work.
2. Additional effort of automating manual testing tasks should be considered in project planning, which pays off in later development phases when releasing subsequent software versions. Other repetitive manual tasks should also be automated. This can include automated testing and practices such as continuous integration and continuous delivery. This also seems to be obvious, but it is easy to be forgotten when planning internal product development projects with a fixed price model, considering that the client can refuse the plan, when the forecasted time horizon becomes too long or may sign a contract with a competition in such a case.

5.2. Improvements regarding prioritizing software features

The literature suggests that one of the reasons for failures in traditional project planning are the requirements (and requirements risks) that are not handled correctly (Krancher, 2020). In particular, the prioritization of requirements that is based on the value it brings to users and clients is very important (Saher et al., 2018). The assumption in traditional planning is that all identified work will be completed (Kosztyan et al., 2020), which means that work is performed in the most favorable order from the perspective of the implementation team. Therefore, towards the end of the project, if not all features have been implemented, some of them are abandoned. Because the implementation was not carried out in the order of the most valuable business features, some features with higher value than those delivered are left out (Cohn, 2006).

To improve the process of prioritizing features to be implemented, the use of the Kano model has been proposed, and a complete novel procedure for prioritizing based on this model has been developed. Despite the extensive description of the Kano model in the literature, there is a lack of a comprehensive procedure that incorporates the Kano model in software release planning, taking into account all steps from requirement gathering to priority determination for software features.

To enhance the process of prioritizing software features to be implemented within the scope of internal IT project planning, the following proprietary procedure utilizing the Kano model has been developed. The procedure consists of five steps as presented in Figure 7.

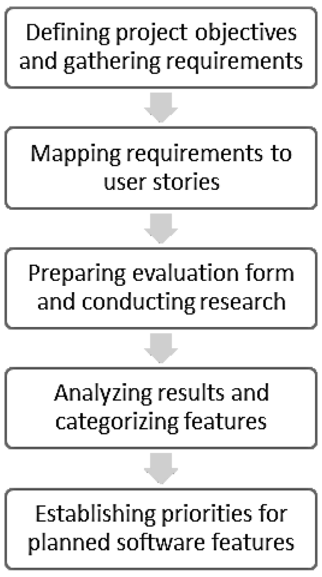
Defining project objectives and gathering requirements
Mapping requirements to user stories
Preparing evaluation form and conducting research
Analyzing results and categorizing features
Establishing priorities for planned software features

Figure 7. New procedure for internal product development project planning.

To illustrate the practical application of the developed prioritization procedure, the implementation of each step of the procedure is described in detail below:

1. Defining the project goal and gathering requirements. Prior to project planning, the organization establishes a high-level vision and product roadmap. At this stage, the product manager is responsible for preparing the initial project goal and, consequently, the product developed within the project. Before starting to gather requirements and analyzing stakeholders, the product manager should determine the strategic reason for developing the product according to the product manager's assumptions. To achieve this, the product manager should analyze current opportunities and threats, ideally based on both historical and current data. SWOT analysis can be a helpful method (Benzaghta et al., 2012). Based on this analysis, the product manager, together with the forming project team, can justify why the proposed product vision is the right direction for development. The next step is gathering requirements, which enables the preparation of a formal requirements specification. To do this, it is recommended to use techniques such as brainstorming, documentation analysis, interface analysis, focus groups, interviews, observations, prototyping, reverse engineering, or questionnaires (Young, 2002; Maguire, Bevan, 2002).

2. Mapping requirements to user stories. The identified requirements should be presented in the form of user stories. Expressing needs in this format facilitates understanding of the planned software features that will address the discovered user requirements. It is important to ensure that the user stories are at an appropriate level of generality, detailed enough to understand the expected functionality but without unnecessary details that overshadow the overall vision of the software features. To accomplish this, a hierarchical approach can be used. For example, identified general requirements are listed in the top of the hierarchy. Then, for each requirement, the user activity related to the software is described. In the final step, for each user activity, low-level details should be specified in the form of a user story. The user story template "As a <user>, I want <need>, so that <goal to achieve>" can be used for this purpose. Similar hierarchical approach, together with similar user story template is presented in (Wautelet et al., 2014) but requirements are called functionalities there. An example table illustrating mapping of requirements to user stories is presented in Figure 8.

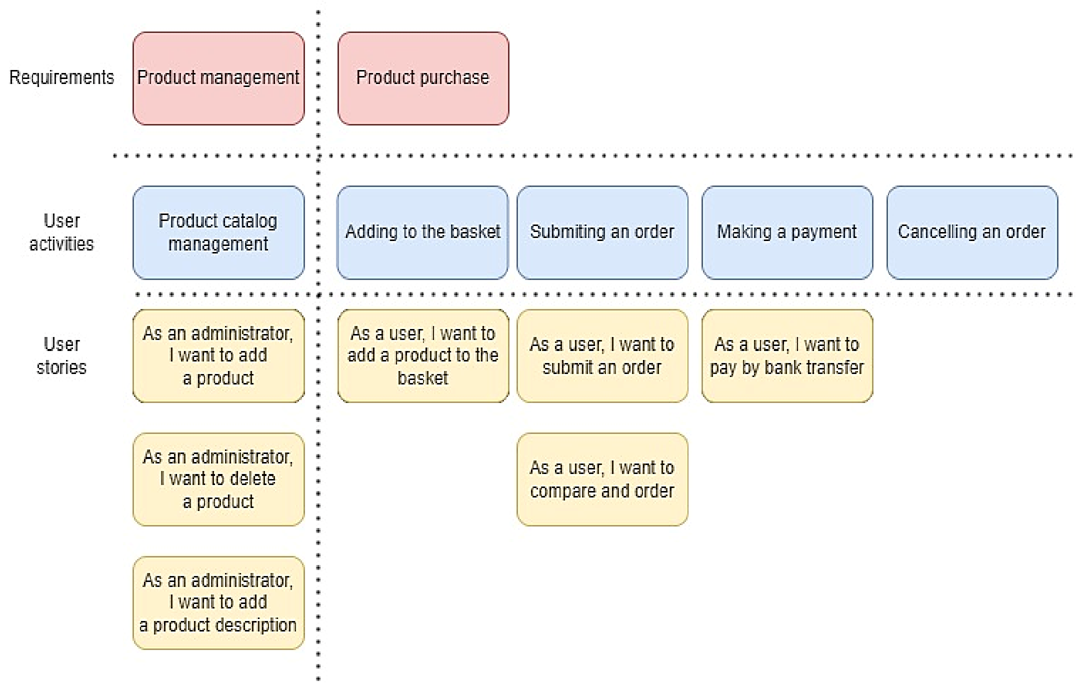
Requirements
Product management
Product purchase
User activities
Product catalog management
Adding to the basket
Submiting an order
Making a payment
Cancelling an order
User stories
As an administrator, I want to add a product
As a user, I want to add a product to the basket
As a user, I want to submit an order
As a user, I want to pay by bank transfer
As an administrator, I want to delete a product
As a user, I want to compare and order
As an administrator, I want to add a product description

Figure 8. Illustrative example of mapping requirements to user stories.

3. Preparation of the Kano evaluation form and conducting attractiveness assessment. For each defined user story (NOTE: the subsequent text refers to user stories, although they are not in a standard form of a user story template) a Kano evaluation form can be developed to assess the attractiveness of the planned user stories among its potential users. The evaluation form suggested in this paper consists of a list of functional (i.e., “How would you assess the software if the user story is implemented”) and dysfunctional questions (i.e., “How would you assess the software if the user story is NOT implemented”) for each examined user story. The template of the evaluation form is presented in Table 2.

Table 2.
User stories evaluation form template

How would you rate the product if:	Like it very much	Like it	Neutral	Don't like it	Don't like it very much
Satisfaction	**2**	**1**	**0**	**-1**	**-2**
1a. you could …					
1b. you couldn't …					
2a. you could …					
2b. you couldn't …					
…	…	…	…	…	…

The Kano model suggests five possible quality categories for the products and services, that are based on the customer satisfaction level and the degree to which the product or service can be delivered (Aslamiyah, 2023). These categories are: Must-be (here referred to as MB), One-dimensional (here: 1D), Attractive (here: AT), Reverse (here: RV) and Indifferent (here: IN). These categories are usually depicted in two listed dimensions of satisfaction level and a degree of implementation. MB category means that when this product or service is implemented, customers are neutral but are not satisfied when it is not implemented. 1D category means that customers are satisfied when the product or service is implemented and dissatisfied when not. AT means satisfaction when product/service is achieved fully, but it does not cause dissatisfaction when it is not implemented. RV means high level of dissatisfaction when implemented and IN is reserved for product/services that neither cause satisfaction nor dissatisfaction and are indifferent for the customer. In this paper we suggest a different representation of the Kano model that is adjusted to the construction of the questionnaire that was proposed: the dimensions are five possible levels of satisfaction when user story is implemented (-2, -1, 0, 1, 2) and five levels of satisfaction when user story is not implemented (-2, -1, 0, 1, 2). This leads to the model presented in the following figure, where MB = <0,-2>, 1D = <2,-2>, AT = <2,0>, IN = <0,0> and RV =<-2,0> (Figure 9).

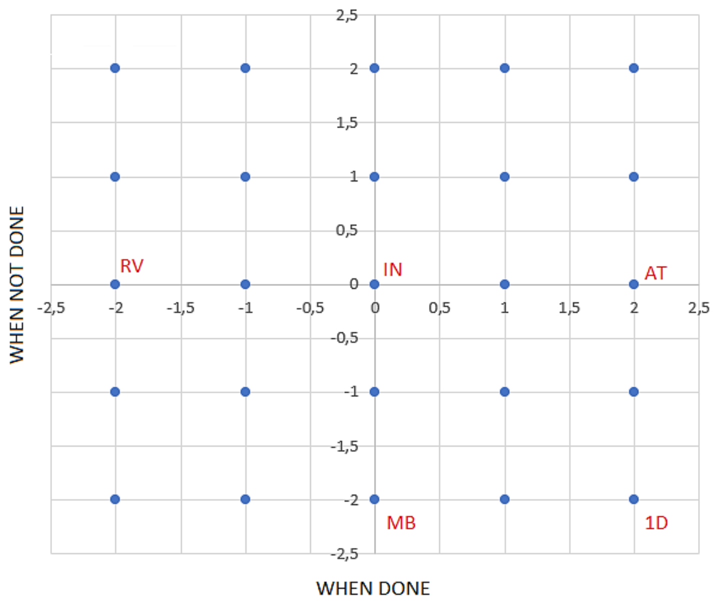

Figure 9. Kano model representation adjusted to the suggested evaluation form.

Having a questionnaire, it is necessary to select a research group and conduct the survey. To obtain reliable research results, the research group should consist of an adequate number of respondents representing different groups of potential users.

To verify the credibility of the obtained responses, a so-called lie scale (Stupnicki, 2015) can be included in the questionnaire. For this purpose, the same software feature questions should be repeated in the response sheet using different formulations and placed in different sections of the form. Then, a reliability criterion should be established, for example, the agreement of 4 out of 5 pairs of questions forming the lie scale. If there is only one pair with inconsistent responses, the survey is considered reliable. If two or more pairs have inconsistent responses, the survey of that particular respondent should be disqualified. An example of a completed form is presented in Table 3.

Table 3.
An example of a completed form

How would you rate the product if:	**Like it very much**	**Like it**	**Neutral**	**Don't like it**	**Don't like it very much**
Satisfaction	**2**	**1**	**0**	**-1**	**-2**
1a. you could display an analysis of the history	X				
1b. you couldn't display an analysis of the history					X
2a. you could undo the last operation		X			
2b. you couldn't undo the last operation					X
3a. you could add a profile photo	X				
3b. you couldn't add a profile photo			X		

4. Developing survey results and assigning categories to user stories. Based on the obtained results from the previous step, each user story U should be assigned to a category of the Kano model ({MB, 1D, AT, IN, RV}). To perform this assignment, for each user story U that is considered, all functional responses and all dysfunctional responses are summed (possible responses are integer values from the set of possible values {-2,-1,0,1,2}), and the average response is calculated as $U_f = \sum_{i=1,\ldots,N} U_f^i$, for the functional responses and $U_{df} = \sum_{i=1,\ldots,N} U_{df}^i$ for dysfunctional responses, where N is the number of responses collected. Then a pair $<U_f, U_{df}>$ is determined and a nearest Kano representative of a given category (see Fig. 9) is determined. Obtaining similar distances for more than one Kano representatives may indicate the presence of subgroups within the research group that perceive the user story differently. In such a case, the responses should be reanalyzed by splitting the research group into sub-groups. Also obtaining big spread of the questionnaire answers (e.g. standard deviation that is greater than a given threshold) may indicate similar problem.

An example result of assigning user stories to categories of the Kano model is presented in Table 4.

Table 4.
An example of assigning user stories to the categories of the Kano model

User story	Distance to MB	Distance to 1D	Distance to AT	Distance to IN	Distance to RV
Displaying an analysis of the history	1,58	0,70	1,58	2,12	3,81
Undoing the last operation	0,5	2,06	2,5	1,5	2,5
Adding profile photo	1,41	1,41	1,41	1,41	3,16

For the first functionality, "Displaying historical analysis", the shortest distance to category representatives indicates that it is a one-dimensional (1D) user story. This suggests that this function linearly affects the satisfaction of the end user – the more extensive it is, the more positively it will impact user satisfaction. For the functionality of "Undoing the last operation", the closest representative is the must-be (MB) category. This indicates that this user story must be implemented, otherwise it will significantly decrease the satisfaction of the end user. In the case of the last user story, "Adding a profile picture", the responses indicate that it can belong to all: MB, 1D, AT or ID categories (as distance to all of them is similar). This suggests that there may be subgroups within the surveyed group that perceive this function differently.

5. Determining the priorities of the planned software functionalities. Based on the categories assigned to the user stories, each user story should be given a priority. User stories with low priority should be excluded from the project scope first if there are insufficient resources to implement them. When assigning priorities and following the Kano model, the following observations can be done:
   - Must-be user stories must be implemented before software release.
   - One-dimensional user stories should be included as much as possible, but some of them can be abandoned if it is not feasible to implement all of them.
   - Attractive user stories should be implemented at least in a small number, as they result in a significant increase in user satisfaction when present in the product.
   - Indifferent and reverse user stories should be abandoned as they do not bring value to the user or negatively affect their satisfaction.
   - Questionable user stories can be reanalyzed in another research group. If conflicting results continue to be obtained, they should be reformulated or abandoned.

According to the improvements suggested in the previous subsection, the presented procedure (excluding its first step) should be performed in every Sprint. It then allows tracking the progress of the MVP version comparing to the initial time estimations of the product development.

5.3. Validation procedure

In order to measure the quality of the proposed method, a survey of software usability evaluation among its users can be conducted after the release of the MVP version, in the planning of which the proposed procedure was used. Based on the obtained results, it is possible to assess whether the new version of the software meets the needs of users to the expected extent. These results can then be compared with the results prior to the improvements described in this article. The described procedure should be supplemented with a procedure for evaluating the quality of planning. By measuring the difference between the estimated and actual MVP release time and expressing it as a percentage of the estimated value (as was done in the planning problem identification presented at the beginning of this paper), a simple assessment of planning quality can be obtained. Both above procedures are suggestions only. Collecting the results from these procedures will be the subject of further research.

The advantages of the proposed procedure include the previously mentioned simplicity and the ability to be implemented as part of an existing software development process. Among the disadvantages of the proposed procedure, issues with verifying the reliability of results obtained from respondents' answers and a lack of systematic procedure for determining final priorities can be highlighted. The first issue can be minimized by using the previously described lie scale. The second issue is the limitation of the current method, which was only a suggestion, and its development was not the main objective of this article. Anyway, this systematization is planned as part of further research. The advantages and disadvantages of the proposed procedure, are summarized in Table 5.

Table 5.
Advantages and disadvantages of the proposed procedure

Advantages	Disadvantages
- Simplicity of the procedure - Ability to implement the procedure as part of an existing agile software development process	- Difficult to verify the credibility of the survey results - Lack of a systematic procedure for determining the final priorities of user stories, based on the described Kano forms

6. Summary

The application of the Kano model in Agile context is already present in the literature (Saher et al., 2018) but the contribution of this paper is the analysis that was performed that helps to determine project types (based on project requirements criteria) for which it is best to use the Kano model. Project classification suggested in this article has not been described in the literature. What is more, the author’s method of collecting inputs for the Kano model, that is based on the questionnaire, is original and novel and has not been presented in the literature so far. Another contribution of the paper is that the Kano model is applied to project planning, which, according to the authors' literature analysis, has not been described anywhere, at least not in case of mid-term planning that goes beyond the Sprint frame.

Based on the conducted research, it has been found that Agile project management methodologies like requirements prioritization described could help to improve project planning.

References

1. Ali, N., Lai, R. (2021). Global software development: A review of IT practices. *Malaysian Journal of Computer Science*, *34*.
2. Aslamiyah, S. (2023). Implementation of the Kano model and importance and performance analysis in the development of a web-based knowledge management system. *Journal of Information Systems and Technology Research.*
3. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D. (2001). *Manifesto for Agile Software Development.* Retrieved from: https://agilemanifesto.org, April 2023.
4. Benzaghta, M.A., Elwalda, A., Mousa, M.M. (2012). SWOT analysis applications: An integrative literature review. *Journal of Global Business Insights*.
5. Campese, C., Hornos da Costa, J.M. (2019). Requirements management methods: Does the literature need an update? *Product Management and Development*, *17(1).*
6. Cao, L., Ramesh, B., Abdel-Hamid, T. (2010). Modeling dynamics in agile software development. *ACM Transactions on Management Information Systems*.
7. Cohn, M. (2006). *Agile estimating and planning*. Pearson Education, Inc.
8. Fernández-Diego, M., Méndez, E.W.R., González-Ladrón-de-Guevara, F., Abrahão, S., Insfran, E. (2020). An update on effort estimation in agile software development: A systematic review. *IEEE Access*, *8*.
9. Francisco, M.G., Sant'Anna, O.C. (2019). Design for Six Sigma integrated product development reference model through systematic review. *International Journal of Lean Six Sigma*.
10. Ghodeswar, B., Vaidyanathan, J. (2008). Business process outsourcing: An approach to gain access to world-class capabilities. *Business Process Management Journal*.
11. Golrizgashti, S., Hejaz, A.R., Farshianabbasi, K. (2019). Assessing after-sales services quality: Integrated SERVQUAL and fuzzy Kano's model. *International Journal of Services Economics and Management*, 137-166.
12. Kelly, A. (2004). *Why do requirements change?* Overload.

13. Kosztyan, Z.T. (2022). MFPP: Matrix-based flexible project planning. *Expert Systems with Applications*, *42(9).*
14. Kosztyan, Z.T., Jakab, R., Novak, G., Hegedus, C. (2020). Survive IT! Survival analysis of IT project planning approaches. *Operations Research Perspective*, *7*.
15. Kosztyán, Z.T., Novák, G., Jakab, R., Szalkai, I., Hegedűs, C. (2023). A matrix-based flexible project-planning library and indicators. *Expert Systems with Applications*, *216*.
16. Krancher, O. (2020). Agile software development practices and success in outsourced projects: The moderating role of requirements risk. *Lecture Notes in Business Information Processing*, *383*.
17. Lee, S., Geum, Y. (2021). How to determine a minimum viable product in app-based lean start-ups: Kano-based approach. *Total Quality Management & Business Excellence*, *32*.
18. Li, M., Zhang, J. (2021). Integrating Kano model, AHP, and QFD methods for new product development based on text mining, intuitionistic fuzzy sets, and customers' satisfaction. *Mathematical Problems in Engineering*.
19. Maguire, M., Bevan, N. (2002). *User requirements analysis: A review of supporting methods*. Proceedings of the IFIP 17th World Computer Congress - TC13 Stream on Usability: Gaining a Competitive Edge.
20. Nicolas, J., Carrillo De Gea, J.M., Nicolás B., Fernández-Alemán J.L, Toval A. (2018). On the Risks and Safeguards for Requirements Engineering in Global Software Development: Systematic Literature Review and Quantitative Assessment. *IEEE Access, 6*, 59628-59656.
21. Project Management Institute, Inc. (2013). *Software extension to the PMBOK® Guide (5th ed.)*. Project Management Institute, Inc.
22. Saher, N., Baharom, F., Romli, R. (2018). A review of requirement prioritization techniques in agile software development. *International Journal of Scientific & Technology Research*.
23. Shirazi, F., Kazemipoor, H., Tavakkoli-Moghaddam, R. (2017). Fuzzy decision analysis for project scope change management. *Decision Science Letters*, *6*, 395-406.
24. Stupnicki, R. (2015). *Analiza i prezentacja danych ankietowych*. Wydawnictwo AWF.
25. Vavpotič, D., Robnik-Šikonja, M., Hovelja, T. (2019). Exploring the relations between net benefits of IT projects and CIOs' perception of quality of software development discipline. *Business & Information Systems Engineering*.
26. Wautelet, Y., Heng, S., Kolp, M., Mirbel, I. (2014). *Unifying and extending user story models*. 26th International Conference on Advanced Information Systems Engineering.
27. Young, R, Grumman, N. (2002). Recommended Requirements Gathering Practices. *Computer Science*.