C. Wohlin and A. Andrews, "Prioritizing and Assessing Software Project Success Factors and Project Characteristics using Subjective Data", Empirical Software Engineering: An International Journal, Vol. 8, No. 3, pp. 285-308, 2003.

Prioritizing and Assessing Software Project Success Factors and Project Characteristics using Subjective Data

Claes Wohlin
Dept. of Software Engineering and Computer Science, Soft Center
Blekinge Institute of Technology
Box 520, SE-372 25 Ronneby
Sweden
+46 457 385820
claes.wohlin@bth.se

Anneliese Amschler Andrews
Computer Science Department
Colorado State University
Fort Collins, CO 80523-1873
USA
+1 970 491 7016
aaa@cs.colostate.edu

Abstract

This paper presents a method for analyzing the impact software project factors have on project success as defined by project success factors that have been prioritized. It is relatively easy to collect measures of project attributes subjectively (i. e. based on expert judgement). Often Likert scales are used for that purpose. It is much harder to identify whether and how a large number of such ranked project factors influence project success, and to prioritize their influence on project success. At the same time, it is desirable to use the knowledge of project personnel effectively. Given a prioritization of project goals, it is shown how some key project characteristics can be related to project success. The method is applied in a case study consisting of 46 projects. For each project, 6 success factors and 27 project attributes were measured. Successful projects show common characteristics. Using this knowledge can lead to better control and software project management and to an increased likelihood of project success.

Keywords

Project success, subjective measures, project assessment.

1. Introduction

War stories from failed software projects are much too common [Glass98]. Failed software projects often make the headline news, but it should be remembered that most projects are not disastrous. This does not mean that there is not room for improvement; running software projects is difficult and requires continuous vigilance to ensure that projects are successful. In particular, it is important to manage risk and work actively on risk prevention and mitigation.

Many organizations collect a myriad of data about projects that is supposed to help them to assess projects. Not all are objective measurements like number of defects or effort spent. Other project attributes that have been claimed to impact project success are based on project personnel's expert (subjective) judgement. Such data is most easily and commonly collected as part of a questionnaire. We call such measurements sub-

jective and the project attributes measured subjective variables. There are two types; project variables (e. g. stability of management team) and success variables (e. g. timeliness of release, perceived quality of the software). While some of those attributes could no doubt be measured objectively, the reality at many organizations is that this is too much overhead and subjective measures as defined above are easier and faster to collect. The question then becomes whether and how much use one can make of them. This is the general topic this paper tries to address. Specifically, we want to identify which subjective project variables are key drivers for project success as embodied by success variables. With a large number of project variables, it is not only important to know which subset of project variables drives certain project success variables, but also to determine priorities.

A method for analyzing project variables with respect to a single success variable was introduced in [Wohlin00], but projects normally have several success criteria to fulfill (e.g. timeliness of delivery and quality of software). Thus, the method is extended here to allow multiple success criteria. Knowing which project variables primarily drive which project success variables as well as overall project success allows software risk prevention and mitigation for future projects as follows: project variables can be measured during a project. If their level is not high enough, this identifies a specific risk which can then be managed.

More formally, let pv_1, pv_2, … pv_i,…, pv_k be the project variables and sv_1, sv_2, … sv_j … sv_m be the success variables. Further let sv_{agg} be an aggregate success variable that ranks overall project success based on values for all (prioritized) success variables. This can be achieved either through an explicit aggregation function or through lexicographic ordering of the relative importance of success variables. This ranking function defines (desired) priorities between project goals. For example, is it more important to deliver on time or that the number of defects is below a certain threshold? Success variables must be prioritized to ensure that they reflect the best combination of project goals. The objective is to find estimators that are able to predict project success from project variables. This is illustrated through the formula below.

$$f\langle pv_1, \ldots, pv_k \rangle = sv_{agg}$$

Both pv_i and sv_j are rank order variables. The objective is to identify which project variables are good estimators for the aggregate success variable. Specific values for project variables also afford an opportunity to set goals related to these values to obtain desired outcomes in terms of a success variable or aggregate success variable. When the project is completed, the success variables are measured. The project and success variables for a software project are shown in Figure 1. The project variables describe key drivers and characteristics of the software project and can be measured (or estimated) prior to starting a project and then tracked as the project progresses.

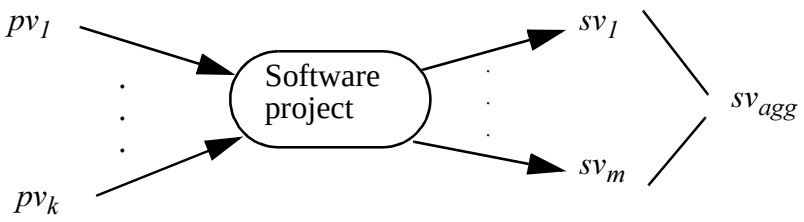

FIGURE 1. Project and success variables.

The paper is outlined as follows. Section 2 gives a brief background of some related work. An overview of the method, in Section 3, gives a basic understanding of its objectives and its steps. Section 4 introduces two ways of aggregating several success variables into one success variable. The steps introduced in [Wohlin00], which are able to cope with a single success variable are described briefly in Section 6 to highlight how it may be used for an aggregate success variable. The method is then illustrated in a case study in Section 9, and finally some conclusions are presented in Section 10.

2. Background

The key idea in this paper is to understand the main reason behind software project success, and to support the use of this understanding. Thus, the focus is on the project level. Recently, a book addressing software success on an organizational level has been published [Hoch00]. The book is based on a major survey of software development companies. It concludes that there are patterns that are typical for a successful company and that these patterns differ from those for a less successful company. The aim is similar to our approach, although we focus on the projects of a single company. Further our method identifies key project factors that influence the likelihood for success rather than trying to state general reasons that may or may not apply in a specific case.

The aim of our work is hence also different from, for example, CMM [Paulk95]. CMM is also based on knowledge from surveying many companies and then structuring the acquired knowledge into levels that a company may reach. This means that the CMM, and other similar methods, is fairly prescriptive in the sense that they make recommendations of what ought to be improved. In addition, the main focus is on improving the organizations. Our approach is driven by project variables (and their measured values) that are specific for a particular company and the software it develops. Thus our method is a tool for identifying your own key factors for software project success rather than using a general set of factors.

From a more technical point of view, the work is based on the use of subjective measurement [Hughes96]. The latter has traditionally primarily been used in effort estimation [Gray99], although it recently was applied to identification of risk [Ropponen00]. In addition, subjective data has been used in assessing the cost-effectiveness of software inspections [Briand00]. The method combines the use of project data and expert opinion. Other uses of expert (subjective) data include cost estimation [Briand98] and effort estimation [Höst97], [Höst98]. Chulani et al. [Chulani99] performed a Bayesian Analysis of several versions of the COCOMO II model, part of which is based on subjective measures of parameters that together form a calibration constant. They show that these models can benefit from considering expert judgement as well as other project data. The potential benefits of using subjective measurements has hence been recognized in several areas of software engineering, although to the best of our knowledge it has not been part of a method for analyzing software project success.

3. Method overview

Figure 2 shows the high level analysis process. Roman numerals on the right side of the figure identify the major steps. They correspond to the subsection headings explaining each major step.

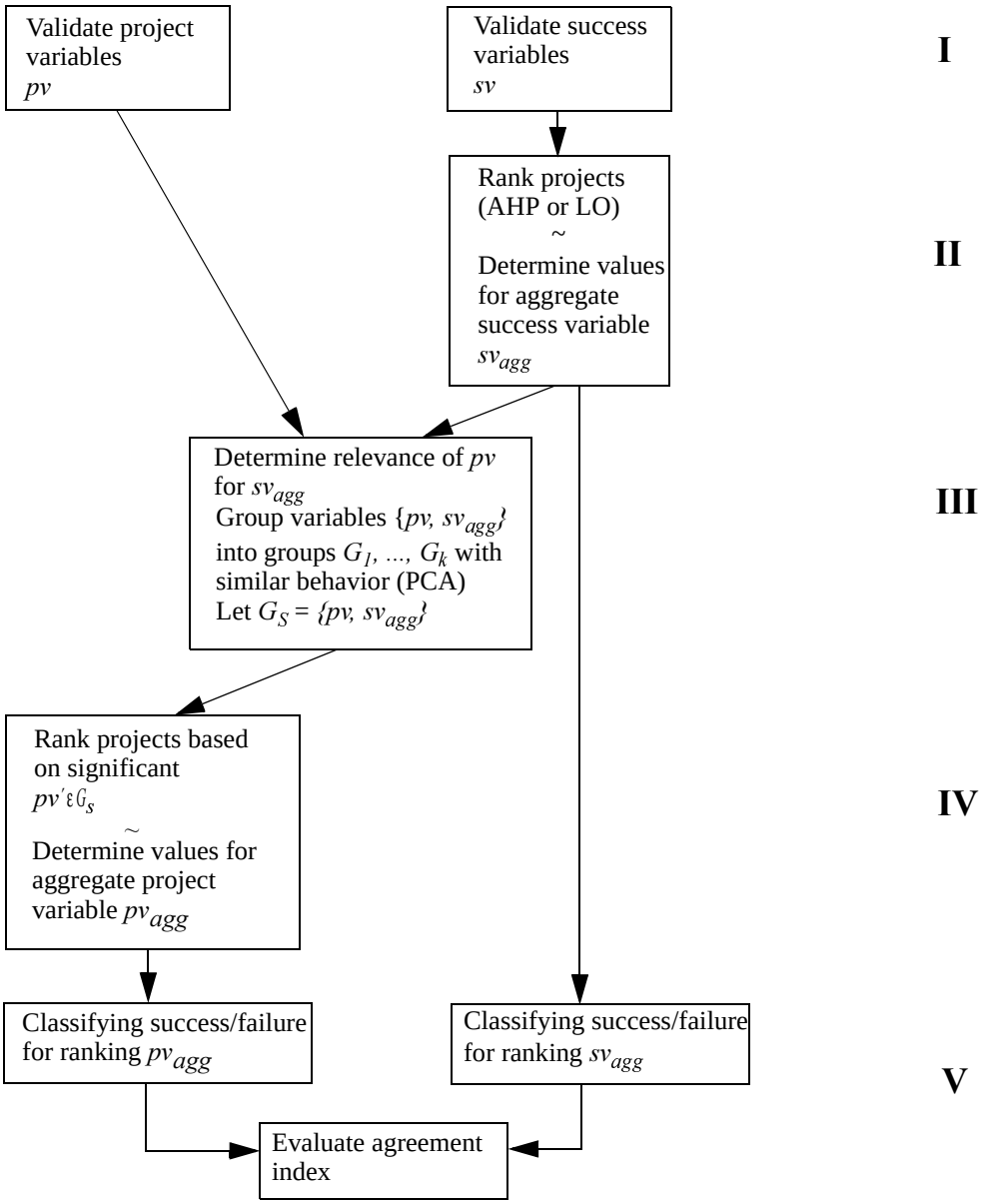

FIGURE 2. Method overview.

3.1 Validate project and success variable data

This step validates both project and success variables. Given their subjective nature, this is important to make the data credible.

3.2 Rank projects by project success

Then projects are ranked by considering all success variables systematically. Methods to do so include AHP [Saaty80] and lexicographic ordering (LO). In software engineering, this has previously been applied to software requirements as in [Karlsson97], where requirements were judged based on value and cost and in [Shepperd01] to estimate project effort when having little objective data. This rank ordering defines a new aggregate success variable sv_{agg}. It is defined as a rank order (ordinal) metric.

3.3 Determine project variables with similar behavior as sv_{agg}

The purpose of this step is to reduce the (often large) number of project variables to those whose behavior is most in tune with the behavior of the aggregate success variable. I. e., if one of them increases, the other does as well; if one of them decreases, the other does, too. One of the most popular methods to identify and group variables with such behavior is Principal Components Analysis (PCA) [Kachigan86]. It has been extensively used in software engineering, e. g. [Khoshgoftaar94], Khoshgoftaar96], [Ohlsson98], [Ohlsson99].

The motivation behind using PCA for identifying the key project variables is based on two things. First, we would like to reduce a possibly large number of variables. PCA is very suitable for abstracting joint behavior from variables. Thus, PCA helps in reducing the number of variables (we need not consider project variables in components other than the one that contains the aggregate success variable). Second, we are primarily interested in a method that identifies variables that covariate, since the objective is to identify those key project characteristics that drive project success. The project variables grouped with the aggregate success variable by PCA (at a high enough loading) provide a solution to this question. Basically, we use the covariance calculations in PCA to identify variables that covariate, i.e. project characteristics that covariate with the success variable.

As an aside, multiple linear regression, with as many variables as we have, is bound to lead to strange combinations of positive and negative parameters in the regression formula. One possible solution to this is to apply PCA to reduce the explanatory variables that may enter the model. If this is not done, then the results may be difficult to interpret (and to use for risk mitigation). For example, [Conte85] report on a regression formula that fits a dependent variable (number of defects) with a series of independent variables (including size, complexity, and lines of comments among other things). The parameter for size is positive. It is also positive for lines of comments. A misleading interpretation would have been to conclude that more comments are “bad” since they lead to more defects. Had PCA been used, both the size measure and the number of lines of comments measure might have been grouped with number of defects, but with different loadings representing to which degree they explain variance in the component and thus their importance for a given component.

In other words, at this stage in the analysis, we are not interested in a prediction model, rather in identification of key project variables, and this is done using PCA by finding the project variables that covariate with the success variable. We have chosen not to use the findings from the PCA in a prediction model, and hence the result of this step is a set of project variables $pv'_1, pv'_2, \ldots pv'_i, \ldots, pv'_k$ which show statistically similar behavior with the aggregated success variable sv_{agg}. We call these key project variables.

3.4 Rank projects based on key project variables

As in step 2, this rank determines values of a new (rank order) measure for evaluating projects based on the set of key project variables $pv'_1, pv'_2, \ldots pv'_i, \ldots, pv'_k$ which have been aggregated into an aggregate measure pv_{agg}.One measure how well the key

project variables are able to predict levels of project success would be how much the ranks defined by pv_{agg} and sv_{agg} agree.

3.5 Classify success and failure for both pv_{agg} and sv_{agg}

Considering each rank individually for both rankings may be more detailed than necessary, especially if we are only interested in project success or failure (a much coarser ranking consisting of only two values). For this situation a classification step is added that defines which ranks for pv_{agg} and sv_{agg} determine the boundary between success and failure. Based on this classification, we can now compute an Agreement Index [Altman91]. It determines how well the two classifications agree and thus can substitute for each other. If the agreement index is high, it means that we can use key project variables to predict success or failure for a project.

The following sections explain each of these steps in more detail and provide justification, limitations, and references for prior uses in software engineering for each technique used (if applicable). The analysis method is an extension of [Wohlin00]. The five step procedure suggested in [Wohlin00] can handle only a single success variable, i. e. the method is capable of identifying key project characteristics for a specific success variable. The method is, however, not able to cope with prioritizing and aggregating several success variables. We extended the method is to include this capability.

4. Validate project and success variables

We assume that these variables have been measured based on expert judgement, i. e. are subjective. [Briand00] makes an excellent point about the value of using expert opinion: it is valuable and should be included if possible, particularly if it is not available through other means such as direct measurements, observations, or experimentation. However, we need to be concerned with the validity of subjective data. It is data, in our case measured on an ordinal scale. These scales must be well-defined.

For measuring subjective variables various rating schemes exist, where the most commonly used are ordinal scales [Fenton96]. The meaning of the different values on the scale should be determined and these should provide a good differentiation between projects. Methods to define scales reliably are described in [Mayrhauser90]. Ideally, subjective metrics for such variables should be evaluated for inter-rater and re-test reliability [Allen79]. Generally, well-known expert knowledge elicitation techniques are designed to prevent problems with expert-based subjective data (bias, uncertainty, incompleteness). When done properly, subjective data collection based on expert judgement means that data is gathered formally, in a structured manner and that its validity can be evaluated (e. g. see [Meyer91] and [Allen79].

Besides investigating the validity of the subjective measures for project and success variables, it is important to determine their relevance for a specific analysis scenario, i.e. a measured data set. The values obtained for a particular success variable must be able to differentiate between projects. If that is not the case, then either the projects are similar with respect to this variable and there is no need to include it in the analysis, or the scale is badly formulated making projects appear equally good (or bad). To address

this issue, one may require either or both of the following conditions to hold for measures of the success variables:

- Require the median value of the measures to lie close to the population median. For example, for a five-point scale one may require that the median value is equal to 3. If this is not the case, the measures have a tendency to be clustered either on the low- or high-end of the scale.
- Require that for each value of the metric, there is at least one project attaining that value. For example, for the five-point scale this would require that all values from 1 to 5 are present in the data set.

5. Rank projects by overall project success

Usually more than one project success variable is important when judging project success. Not all have the same importance in every situation. Thus, it is important to weigh different success variables against each other and to prioritize them. Often all aspects seem essential and it is hard to say that there is something that is less important. Software requirements have a similar problem, i.e. everything seems important.

One opportunity to prioritize different success variables, and obtain a ranking, is to use the analytic hierarchy process (AHP) [Saaty80]. It uses pairwise comparison to obtain an ordered list. In software engineering, this has previously been applied to software requirements as in [Karlsson97], where requirements were judged based on value and cost and in [Shepperd01] to estimate project effort when having little objective data. AHP allows for weighting the success variables by quantifying their relative importance. This set of weights is then used with the measured values of the success variables to determine the values of the aggregate success variable.

In situations when weights cannot be determined, lexicographic ordering (LO) can be used instead. It is based on ranking the relative importance of the success variables. Both methods are described in more detail in Appendix A1 and A2. Either method results in a ranking of the projects that takes into account the importance of each success variable. The project success ranking can differ depending on the method used. It is also influenced by the application domain's specific needs and the resulting impact on what project outcomes are important. For example, a safety-critical system would rank safety and reliability very highly while a low risk mass-market system like a game might emphasize timeliness in hitting the market window.

Both prioritization alternatives, Analytic Hierarchy Process (AHP) and Lexicographic Ordering (LO), result in an aggregate success variable that ranks all projects based on overall project success. This ranking defines the new aggregate success variable that reflects overall project success.

Both AHP and LO rank projects based on the importance of the success variables. Unlike AHP, LO does not pose problems for rank order variables, since it sorts the projects based on the relative importance of the different success variables. On the other hand, it does not provide a ratio level aggregate success variable and it does not quantify inconsistency problems with respect to relative importance of success variables. We recommend LO when scales are widely different or when there are concerns about using rank order scales in the comparisons of AHP.

6. Group key project variables with aggregate success variable

[Wohlin00] developed a method to assess project success based on a single success variable and to group it with key project variables. Like we do here, [Wohlin00] assumes that both project variables and the success variable are measured subjectively (based on expert judgement) on ordinal scales. Similarly, the objective is to identify which project characteristics are most important to achieve success. The method can be used to assess projects, but also in planning and managing risks of software projects. If it is possible to estimate the values of different project characteristics, it is possible to judge whether a project is likely or not to become a success for a given success variable. If the outcome of this early assessment does not meet expectations, it is possible to re-plan or take special actions in order to try to address the risk of not becoming successful.

We already validated in step 1 that all project and success measures have been formulated so that a higher value on the scale is more desirable. In addition to validity and relevance of project and success variables, project variables must also be relevant to the aggregate success variable. Relevant project variables must have a positive correlation with the aggregate success variable. If that is not the case, there are either some underlying factors that were not captured, or the scale does not represent what we think. For example, one may expect that higher competence in a project will make it more likely to be successful, yet if the most competent personnel is assigned to the most difficult and demanding projects this may prevent a positive correlation between project variables and the aggregate success variable. This is why project variables that do not have a positive correlation with the aggregate success variable are removed from the analysis. The objective is to ensure that the outcome of the analysis makes intuitive sense and is useful for planning and controlling software projects with the intention of managing project risk.

Data in too many databases are collected without a purpose (except to collect data). This often leads to measures that are highly correlated because they measure almost the same thing. Different techniques exist to extract a useful subset. One such technique is Principal Component Analysis (PCA) [Kachigan86]. PCA groups correlated variables or variables with the same behavior into a number of principal components where each component accounts for the maximum possible amount of the variance for the variables being analyzed. The number of principal components extracted may vary depending on the data set and the method chosen for extraction. PCA can help in reducing the number of variables. It also provides support in identifying variables that vary together. This is exactly our objective: to identify those key project variables that vary in the same manner as the aggregate project success variable.

Analyzing project variables with the aggregate success variable using PCA allows us to determine which project variables behave similarly and to identify which project variables behave similar to the success variable. The latter is the key concern in this study. Commonly, variables with a loading of 0.7 or higher are of particular interest since they explain most of the variation of the principal component. Thus, the components are primarily interpreted from the variables with a loading of 0.7 or higher, but other variables in the principal component may also be considered. The latter primarily refers to variables having their highest loading in the component. This loading should

also be higher than that of the success variable. If the success variable has the highest loading and no project variable has a loading higher than 0.7, then we have failed to capture the success variable with the project variables.

Several different situations may occur. For example, the success variable may have the highest loading in a factor and the highest loading for a project characteristic may be above or beneath the threshold of 0.7. Several different combinations of this type may happen in terms of which variables have the highest loadings, and how they relate to the threshold value. This is discussed in more detail in [Wohlin00].

The result of this step is a set of key project variables $pv'_1,...,pv'_k$ that exhibit the same behavior as the aggregate success variable sv_{agg} for the data set analyzed and thus can be considered as candidates for key project characteristic that drive overall project success.

7. Rank project based on key project variables $pv'_1,...,pv'_k$

This step determines an aggregate measure for the key project variables by summing (for each project) the values of the key project variables. This measure determines a rank order of projects and thus the ordinal values of pv_{agg}, the ranking of projects based on the key project variables.

Note that other ways of determining the rank order for pv_{agg} are possible (one being lexicographic ordering based on loading values in the PCA or on the value of the Spearman Correlation coefficient of a key project variable with the aggregate success variable).

Finally, the Spearman correlation is determined between the rankings based on the aggregate key project variable pv_{agg}, and the ranking based on the aggregate success variable sv_{agg}.

8. Classification of success according to both aggregate variables and evaluation of agreement

Each of the aggregate measures has to be mapped into project success or failure. Then an agreement index [Altman91] is computed that evaluates the degree of agreement and hence the degree to which key project variables are able to capture project success.

Using both rankings, we classify projects into two classes, i.e. the successful and unsuccessful respectively (by project variables and by aggregate success variable). Each of the aggregate measures is mapped into project success or failure. This classification may be represented in the form of a diffusion matrix or contingency table. Other classification schemes may be used as, for example, in [Ohlsson99], where three classes are used to classify fault-prone components.

TABLE 1. A diffusion matrix for successful projects.

		Success variable	
		Successful	Unsuccessful
Project variables	Successful	*Correct*	*Misclassification*
	Unsuccessful	*Misclassification*	*Correct*

It is difficult to compare different classifications directly since we may not have the same number of projects that are successful (or unsuccessful) based on project variables and based on the success variable. To compare the degree to which the two classifications agree (and could be used one instead of the other) a measure of agreement between the classifications can be computed. This is called an Agreement Index [Altman91], or kappa statistic. In our case it evaluates the degree to which key project variables are able to capture project success. In software engineering, the kappa statistic has been applied to interrater agreement of process assessments [El Emam99].

The agreement index requires comparable scales for the two classifications, i.e. based on project variables or the success variable. This prevents setting widely different thresholds for each classification, since there would be different numbers of projects judged successful by one method vs. another. To circumvent this problem, we use each ranking of projects and then consider the upper x% of projects successful and the lower (100-x)% unsuccessful projects. For example, x could be 50%, or it could be 75%. Setting x depends on company expectations, and expert judgement.

As agreement index we use the kappa statistic [Altman91]. Briefly, the kappa statistic for our simple case of two raters (models) and two levels (successful and unsuccessful projects) can be explained as follows.

Table 2 shows an example, where the cells state the proportions of the projects with a given rating according to model 1 and model 2. For example, p_{11} = 0.20 means that 20% of the projects are considered successful according to the subjective project variables and successful according to the success variable. The columns and rows are summarized (last column and last row respectively in Table 1), which is indicated with p_{01}, p_{02} p_{10} and p_{20}.

TABLE 2. A diffusion matrix for successful projects.

		Success variable		
		Successful	Unsuccess-ful	Sum
Project variables	Successful	p_{11}	p_{12}	p_{10}
	Unsuccess-ful	p_{21}	p_{22}	p_{20}
	Sum	p_{01}	p_{02}	

The entries in Table 1 are used to derive an agreement index. Let P_A be the proportion in which there is agreement. Then, P_A becomes

$$P_A = \sum_{i=1}^{2} p_{ii}$$

This agreement includes cases in which the agreement is obtained by chance. To remove the effect of chance behaviour, the extent of agreement that is expected by chance is defined as

$$P_E = \sum_{i=1}^{2} p_{i0} \times p_{0i}$$

The agreement index is then defined as

$$\kappa = \frac{P_A - P_E}{1 - P_E}$$

To be able to understand the degree of agreement, the kappa statistic is usually mapped into a rank order scale describing the strength of agreement. Several such scales exist, although they are by and large minor variations of each other. Three scales are presented in [El Emam99]. Here the scale suggested by Altman [Altman91] is used. It is shown in Table 3.

TABLE 3. The Altman kappa scale.

Kappa statistics	Strength of agreement
< 0.20	Poor
0.21-0.40	Fair
0.41-0.60	Moderate
0.61-0.80	Good
0.81-1.00	Very good

An agreement index is derived as a value below 1 and the closer it is to 1, the better is the agreement between classifications. The actual value is then mapped to an interpretation of the agreement. The interpretation suggested by [Altman91] is given in Table 3.

We now apply this analysis method to a major case study.

9. Case study

This method is now applied to an actual case study for two purposes. One is to illustrate its use on an example. The second is to evaluate in a large context to which degree this method works and what kinds of results it is able to produce. Is it really able to identify project success based on key project variables?

9.1 Data

The case study is based on data from the NASA-SEL database [NASA-SEL92]. Further information about NASA-SEL can be found in [Basili95]. Projects in the database available consist of a rich set of project descriptors. They include parameters related to schedules and estimates, resource use (both manpower and computer use), a variety of product characteristics related to structure, size, growth, and change data. The focus here is however on using the subjective data to understand what drives project success. Subjective evaluations rank the projects in terms of problem complexity, schedule constraints, nature of requirements, team ability, management performance, discipline, software quality, etc.

The database contains data from more than 150 projects that span five years. Of these, we selected 46 for analysis, based on completeness of project data recorded. These 46 projects represented a variety of types of systems, languages, and approaches for software development. The subjective variables, in the database, are measured on a five-point Likert scale with the higher value denoting more of the quality ranked. In total, 27 project variables and 6 success variables are measured for the 46 projects. The six success variables are: Agreement of software with requirements (AGGRE), Quality of software (QSOFT), Quality of design (QDES), Quality of documentation (QDOC), Timeliness of delivery (TIMELI), and Smoothness of acceptance test (ACCTEST).

While some of the subjective variables in theory could have been measured non-subjectively (e. g. the number of changes in test plans) they were not measured like this in this case. All project and success variables were the result of expert judgement. In addition, the database did not include information that would have allowed us to infer priorities for success or project variables. Thus we had to make our own assumptions. We did so in order to be able to illustrate all parts of the analysis method. We assumed the following: Timeliness of delivery (TIMELI) is the most important success variable. Quality of software (QSOFT) is more important than quality of the design (QDES). Smoothness of acceptance test (ACCTEST) is more important than quality of documentation (QDOC). Note that this constitutes neither a ranking according to AHS nor LO, but serves as a starting point for further analysis for either method.

9.2 Analysis

1. Validation of project and success variables

   The first step is to determine whether all success variables distinguish sufficiently between projects (i. e. are relevant) Two success variables do not fit the criteria for further inclusion in the analysis. They are AGREE (Agreement of software with requirements) and QDOC (quality of documentation). Figure 3 shows the frequency distributions of the two variables.

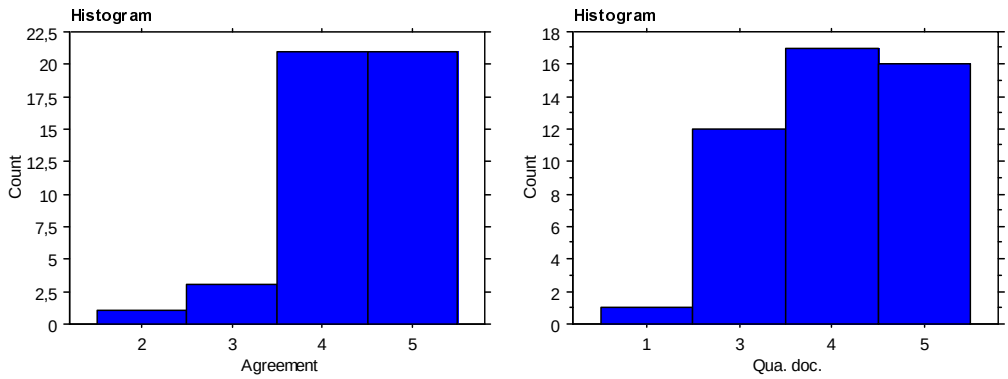

FIGURE 3. Frequency distribution for the two removed success variables.

No project is ranked “AGGRE=1” and likewise no project is ranked “QDOC=2”. Further, both variables cluster around the high values. The median is 4 for both variables. It seems that the organization is able to obtain fairly high quality with regard to these two values in most projects. Hence there is no immediate need to focus on any specific project characteristics to improve them. In accordance with rules for step 1, these two success variables are removed from further analysis.

This leaves four success variables to aggregate into one success variable: QSOFT, QDES, TIMELI, and ACCTEST.

Project variables are validated based on the assumption that more of a property is better. This is the case for all 27 project variables.

2. Prioritization and aggregation of success variables

   The assumptions about relative importance of the remaining four success variable (as stated in section 6.1.) do not specify sufficiently information necessary for either AHP or LO. We illustrate how to determine priorities and aggregate success variables for both methods.

   AHP Prioritization:

   Step 1. There are four success variables, resulting in a 4×4 matrix. All elements in the diagonal have the value “1”. We define the variables in the matrix as follows: (1) QSOFT, (2) QDES, (3) TIMELI and (4) ACCTEST

Step 2. Four success variables require six pairwise comparisons. The six pairs are found in Table 4 together with the value in the comparison using AHP. Table 4 also defines which of the two variables in the comparison is more important and the values of the corresponding preference matrix entries.

TABLE 4. A scenario of the pairwise comparison of success variables.

Pair	Value	More important
(1) QSOFT vs. QDES (2)	5	QSOFT: $x_{12}=5$, $x_{21}=1/5$
(1) QSOFT vs. TIMELI (3)	3	TIMELI: $x_{31}=3$, $x_{13}=1/3$
(1) QSOFT vs. ACCTEST (4)	3	QSOFT: $x_{14}=3$, $x_{41}=1/3$
(2) QDES vs. TIMELI (3)	7	TIMELI: $x_{32}=7$, $x_{23}=1/7$
(2) QDES vs. ACCTEST (4)	3	ACCTEST: $x_{42}=3$, $x_{24}=1/3$
(3) TIMELI vs. ACCTEST (4)	5	TIMELI: $x_{34}=5$, $x_{43}=1/5$

The results in Table 4 define the remainder of the matrix which is as follows:

$$N = \begin{bmatrix} 1 & 5 & 1/3 & 3 \\ 1/5 & 1 & 1/7 & 1/3 \\ 3 & 7 & 1 & 5 \\ 1/3 & 3 & 1/5 & 1 \end{bmatrix}$$

Step 3. Next, we compute the eigenvector of *N.* First, the sums of the columns are (4.53, 16, 1.68, 9.33). Second, the *M* matrix is obtained as:

$$\begin{bmatrix} 0.221 & 0.312 & 0.198 & 0.321 \\ 0.044 & 0.062 & 0.085 & 0.036 \\ 0.662 & 0.438 & 0.595 & 0.538 \\ 0.074 & 0.188 & 0.119 & 0.107 \end{bmatrix}$$

Third, the sums of the rows are (1.052, 0.227, 2.233, 0.488). They are normalized to obtain the priority vector P = (0.26, 0.06, 0.56, 0.12).

Step 4. P states that QSOFT accounts for 26 percent of the success variables' importance, QDES for 6 percent, TIMELI for 56 percent and ACCTEST for 12 percent.

Step 5. The goal of this step is to compute the consistency index CI. The first step is to multiply the preference matrix N with the priority vector P *(R=N*P):* R = (1.11, 0.232, 2.36, 0.499). The λ vector is now obtained by dividing each element in R with the corresponding element in the priority vector *P.* Thus, we obtain λ as (4.26, 3.87, 4.21, 4.16). λmax is the average of the values in vector λ. λmax =4.12. The consistency index CI = (λmax - n)/(n-1) = (4.12-4)/(3) = 0.04.

The consistency ratio is computed by dividing CI by the RI value for n=4 (see Table in Appendix A1). RI=0.9. This gives a CR value of CR=0.04/0.9 = 0.05. This is a very good value. The values for the aggregate success variable can now be computed for all 46 projects by weighting the values of the different success variables with the percentage values obtained from the priority vector. More formally, we have:

SV_{agg} = 0.26 * *QSOFT* + 0.06 * *QDES* + 0.56 * *TIMELI* + 0.12 * *ACCTEST.*

The values for SV_{agg} are used to rank the 46 projects. The ranking is shown in Table 5.

Lexicographic ordering (LO):

Based on the assumptions defined with the data in section 6.1., TIMELI is most important. It is followed by QSOFT, which in turn is followed by ACCTEST. QDES is the least important success variable of the four success variables. TIMELI>QSOFT>ACCTEST>QDES. The projects in the database can now be sorted. The projects are first sorted after the most important success variable, i.e. TIMELI, then the projects within a value for TIMELI are sorted based on the second most important success variable, i.e. QSOFT, and so on. The ranking of the projects using the lexicographic method is shown in Table 5.

TABLE 5. Project rankings using AHP and Method LO respectively.

Project No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
AHP	41	10	3	46	23	34	28	15	45	35	4	28	33	23	23	23
LO	38	10	3	46	23	34	28	19	45	35	4	28	33	23	23	23
Project No.	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
AHP	1	44	10	42	38	39	10	43	21	16	8	18	22	40	5	35
LO	1	44	10	42	39	40	10	43	17	16	9	20	22	41	5	35
Project No.	33	34	35	36	37	38	39	40	41	42	43	44	45	46		
AHP	28	13	32	27	9	35	16	2	7	31	5	18	18	14		
LO	28	13	32	27	8	35	14	2	7	31	5	15	20	18		

Comparison of AHP and LO:

The similarity in ranking using the two methods are apparent from Table 5. The Spearman correlation between the rankings using AHP and LO is 0.995. The high correlation indicates two things. First, there is little difference between the two methods in defining the aggregate success variable for this specific data set. However, one should not draw the conclusion that this is the case for all data sets. Given the similarity of results for both methods, we chose to use the results from AHP during the remainder of the analysis.

3. Principal component analysis

Before determining which project variables show similar behavior to the aggregate success variable, we need to determine whether the project variables are positively correlated with it, i.e. are relevant. For both versions of sv_{agg} (the one determined by AHP and for the one determined by LO) two project variables show a negative correlation and are removed from further analysis. In both cases the same two variables are removed. They are the experience of the team in general and the team's experience of the environment in which the project is conducted. While one would agree that experience is valuable for project success in general, it may not contribute to project success for this set of projects for a variety of reasons. One could be that in this environment, experts may have been assigned to the riskier projects and thus prevent experience from correlating with success.

A principal component analysis (PCA) is conducted for the aggregate success variable and the remaining 25 project variables. The analysis provides two main results. First, it is possible to identify which project variables vary in the same way as the

success variable. These are the key project variables or characteristics that are most closely related to the success of the project. These project variables are to some extent the variables that drive the success of a specific project. Second, it is also possible to study how the PCA groups the different variables to gain a better understanding of which variables vary together, although this is not the main focus of the analysis method. The result of the PCA is shown in Table 6.

Three variables in the first factor have a loading above 0.7; these are marked with dark grey in the table. One of them is the aggregate success variable with a loading of 0.818. The other two variables are the quality of the project plan (loading 0.860) and compliance with the plan (loading 0.811). The results indicate that, given the prioritized success variables as success criteria, it is crucial to plan the project properly and then to strive to comply with the plan. Other variables, which have their highest loading in the first factor, are: Programming practices (0.647), Management performance (0.588), Schedule (0.538) and Team ability (0.500). These variables are marked with light grey in the table. The variables with high loading in the first factor, given that the aggregate success variable is in the first factor, provide a hint of other aspects that may be important for this particular organization to achieve success. The other factors, group other variables whose behavior is closely related. These other factors can bed use to try to identify common patterns among project variables, but this is outside the scope of this paper.

TABLE 6. PCA for the aggregate variable and the 25 project variables.

Variable	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7
Complexity	0.039	0.732	0.405	0.093	0.077	0.027	0.257
Schedule	0.538	0.468	-0.076	0.385	-0.113	-0.006	0.230
Req. stability	0.334	-0.577	-0.137	-0.001	-0.405	-0.003	-0.305
Quality of req.	0.481	0.281	0.541	0.166	-0.392	-0.038	0.117
Doc. req.	0.371	0.012	0.657	-0.193	-0.018	0.161	-0.147
Req. reviews	0.078	0.188	0.859	-0.099	0.305	0.107	-0.123
Team ability	0.500	0.382	0.317	-0.326	0.137	0.223	0.040
Turnover	0.053	-0.153	-0.003	-0.097	-0.080	0.000	-0.863
Man. perf.	0.588	0.247	0.069	-0.341	0.290	0.150	0.085
Man. exp.	0.368	0.137	0.180	-0.068	0.727	-0.063	0.113
Stability man.	0.315	-0.650	-0.024	0.087	0.219	0.066	0.063
Project plan.	0.860	-0.102	0.108	0.028	0.037	0.117	-0.081
Plans followed	0.811	-0.245	0.138	0.175	-0.034	0.188	-0.024
Prog. practices	0.647	0.271	0.188	-0.010	-0.196	0.374	0.334
Req. meth.	0.342	0.759	0.172	0.095	0.153	-0.012	0.149
Design meth.	0.305	-0.028	0.532	0.127	-0.100	0.612	0.027
Testing meth.	0.328	-0.163	0.114	0.060	-0.051	0.821	-0.061
Test plans	0.368	0.296	0.183	-0.081	0.164	0.741	0.081
QA	0.459	0.185	0.608	-0.007	0.136	0.376	-0.079
CM	0.524	0.620	0.253	-0.018	0.040	0.099	0.034
Access	0.096	0.870	-0.076	-0.142	0.118	0.077	0.027
Dev. to terminals	0.379	0.599	0.050	-0.334	-0.052	0.206	0.176
Memory	-0.57	0.357	0.540	-0.038	0.079	0.039	0.252

TABLE 6. PCA for the aggregate variable and the 25 project variables.

Variable	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7
Response time	-0.005	0.689	-0.143	0.263	0.352	0.051	-0.252
Support	0.080	0.091	-0.053	0.907	0.079	0.087	0.075
Agg. success	0.818	-0.005	0.035	0.007	0.124	0.183	-0.120

4. Rank projects based on key project variables

   The results from the PCA can now be used to rank the projects based on the two key project variables identified (project plan and compliance to plan). The rankings form the basis for the classification into successful and unsuccessful project in the next step. It is also possible to determine the Spearman rank correlation between the rankings based on the aggregate success variable and the key project variables. This is primarily done to get an indication of the level of correspondence between the two rankings. In this case the correlation is 0.682, which shows that there is a relationship between the two rankings, although not close to equivalence. The correlation is not the key issue; the actual classification of a project into either being successful or unsuccessful is what we are really after.

5. Classification of success and failure

   Assume ranks have been assigned to the 46 projects as rank 1 to rank 46 where a rank of 1 indicates the top project and a rank of 46 indicates the bottom project (as in Table 4). The projects are classified into successful and unsuccessful projects respectively (upper and lower half). Because of ties in ranks, 24 projects are successful, 22 are not. This means that projects that have a rank of 1 to 24 are regarded as successful and a rank of 26 to 46 denotes an unsuccessful project. This classification is performed for both rankings (by key project variables and by aggregate success variable). Based on the aggregate success variable, 26 projects are successful, 20 are not. Next we determine how often the two classifications agree or not. This is summarized in the diffusion matrix of Table 7.

TABLE 7. A diffusion matrix for successful projects in the case study.

		Success variable	
		Successful	Unsuccessful
Project variables	Successful	21	3
	Unsuccessful	5	17

Most projects are correctly classified. In cases where the key project variables indicate a success, 21 of 24 projects also become a success according to the aggregate success variable. Similarly, 17 projects pinpointed as unsuccessful based on key project variables also become unsuccessful based on the aggregate success variable. Only 5 projects become successful when key project variables indicate an unsuccessful project.

To further investigate how well the classification, based on key project variables, agrees with that based on project success after the fact, we compute an agreement index. For the classification in Table 7, the agreement index is 0.65. According to Table 3, this is a “good” agreement, and we conclude that for this set of projects it is

possible to use the key project variables as a basis for identifying successful and not so successful projects.

This particular case study has shown two main results:

- It is possible to apply the proposed method to identify key project characteristics that influence project success. As a measure of success, we have shown that it is feasible to derive an aggregate success variable from a number of success criteria.
- In the case study, “project planning” and “ability to follow the plans” are crucial project characteristics to obtain a successful project, where the success is defined as the aggregate variable derived.

These results are encouraging. They indicate the importance of identifying key project variables given priorities for project success (as defined in the aggregate success variable). If we are able to estimate values of key project variables before starting a project, it helps in assessing potential for project success. Consequently actions may be taken to increase the likelihood of project success. Thus, the method may be used as a software project risk management tool.

While the analysis of this case study data is carried out on finished projects, it also provides valuable information about what to strive for when launching a new project. When launching a project, it is possible to estimate the values of the key project variables, and, if the values are too low to indicate project success based on analysis of past projects, it is necessary to re-think or be prepared to be less successful than hoped for.

10. Conclusions

This paper has shown that it is possible to identify key project variables that have a major impact on and relate to software project success. A new method of aggregating project success variables has been introduced. It is also shown how the aggregate success variable can be analyzed together with project variables to identify the key project variables that are crucial to achieving project success. The method has successfully been applied in a case study.

Given the success of the case study, the results show that subjective measures may be useful in project success analysis. One of the advantages is that subjective measures are normally fairly simple to collect. The simplicity of the metrics collection and the straightforward analysis of the data make the new method a valuable contribution to the tools available to managers when launching new projects. The method can be used to assess success in historical projects, but the main benefits are obtained if we are able to estimate the project variables before starting the project and use this information to obtain guidance on how to increase the likelihood of project success. The method can be used in early risk management or risk mitigation since it gives managers an opportunity to control and predict success of a specific project. Further work includes application of the method in ongoing projects and evaluation of its predictive capabilities.

Acknowledgment

We are grateful to Frank McGarry for giving us access to the NASA-SEL data which we used in the case study to illustrate our approach. We would also like to thank the reviewers for their comments.

References

[Allen79] M. Allen and W. Yeh, “Introduction to Measurement Theory”, Brooks/Cole Publishing, 1979.

[Altman91] D. Altman, “Practical Statistics for Medical Research”, Chapman-Hall, 1991.

[Basili95] V. Basili, M. Zelkowitz, F. McGarry, J. Page, S. Waligora and R. Pajerski “SEL's Software Process-Improvement Program”, IEEE Software, November 1995, pp. 83-87, 1995.

[Briand96] L. Briand, K. El Emam and S. Morasca, “On the Application of Measurement Theory in Software Engineering”, Journal of Empirical Software Engineering, Vol. 1, No.1, pp. 61-88, 1996.

[Briand98] L. Briand, K. El-Emam, F. Bomarius, “COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment”, Procs. 20th International Conference on Software Engineering, Kyoto, Japan, pp. 390-399, 1998.

[Briand00] L. Briand, B. Freimut, and F. Vollei, “Assessing the Cost-Effectiveness of Inspections by Combining Project Data and Expert Opinion”, International Symposium on Software Reliability Engineering - ISSRE 2000, IEEE Computer Society Press, Los Alamitos, 2000.

[Chulani99] S. Chulani, B. Boehm, B. Steece, “Bayesian Analysis of Empirical Software Engineering Cost Models”, IEEE Transactions on Software Engineering, vol. 25, no. 4, July/August 1999, pp. 573-583.

[Conte85] S. D. Conte, H. E. Dunsmore, V. Y. Shen, “Software Engineering Metrics and Models”, Benjamin/Cummings Publishing Co., Menlo Park, CA, 1985.

[El Emam99] K. El Emam, “Benchmarking Kappa for Software Process Assessment Reliability Studies”, Empirical Software Engineering: An International Journal, No. 4, pp. 113-133, 1999.

[Fenton96] N. Fenton and S. L. Pfleeger, “Software Metrics – A Rigorous & Practical Approach”, International Thompson Computer Press, 1996.

[Glass98] R. Glass, “Software Runaways: Lessons Learned from Massive Software Project Failures”, Prentice Hall, 1998.

[Gray99] A. R. Gray, S. G. MacDonell, and M. J. Shepperd, “Factors Systematically Associated with Errors in Subjective Estimates of Software Development

Effort: The Stability of Expert Judgement", Procs. of the Sixth International Software Metrics Symposium, pp. 216-227, 1999.

[Hoch00] D. J. Hoch, C. R. Roeding, G. Purkert and S. K. Lindner, "Secrets of Software Success", Harward Business School Press, 2000.

[Höst97] M. Höst, C. Wohlin, "A Subjective Effort Estimation Experiment", International Journal of Information and Software Technology, Vol. 39, No. 11, pp. 755-762, 1997.

[Höst98] M. Höst, C. Wohlin, "An Experimental Study of Individual Subjective Effort Estimations and Combinations of the Estimate", Procs. 20th International Conference on Software Engineering, Kyoto, Japan, pp. 332-339, 1998.

[Hughes96] R. T. Hughes, "Expert Judgement as an Estimating Method", Information and Software Technology, Vol. 38, pp. 67-75, 1996.

[Kachigan86] S. K. Kachigan, "Statistical Analysis – An Interdisciplinary Introduction to Univariate & Multivariate Methods", Radius Press, 1986.

[Karlsson97] J. Karlsson, K. Ryan, "A Cost-Value Approach for Prioritizing Requirements", IEEE Software, September/October, pp. 67-74, 1997.

[Khoshgoftarr94] T. M. Khoshgoftaar, D. L. Lanning, "Are the Principal Components of Software Complexity Stable Across Software Products?", Procs. International Symposium on Software Metrics, London UK, Oct. 94, pp. 61-72.

[Khoshgoftarr96] T. M. Khoshgoftaar, E. B. Allen, N. Goel, A. Nandi, J. McMullan, "Detection of Software Modules with High Code Debug Churn in a Very Large Legacy System", Procs. International Symposium on Software Reliability Engineering '96, White Plains, NY, p. 364-371.

[Mayrhauser90] A. von Mayrhauser, "Software Engineering: Methods and Management", Academic Press, 1990.

[Meyer91] M. A. Meyer and J. M. Booker, "Elicitating and Analyzing Expert Judgement. A Practical Guide", Academic Press, Ltd., 1991.

[NASA-SEL92] NASA-SEL; Software Engineering Laboratory Database Organization and Users Guide, Revision 2; NASA-SEL Series, SEL-89-201, Goddard Space Flight Center, Greenbelt, MD, Oct. 1992.

[Ohlsson98] M. C. Ohlsson, C. Wohlin, "Identification of Green, Yellow, and Red Legacy Components", Procs. International Conference on Software Maintenance '98, Washington, DC, pp. 6-15.

[Ohlsson99] M. C. Ohlsson, A. von Mayrhauser, B. McGuire, C. Wohlin, "Code Decay Analysis of Legacy Software through Successive Releases", Procs. IEEE Aerospace Conference 1999, IEEE Press: Piscataway, NJ, 1999.

[Paulk95] M. C. Paulk, C. V. Weber and B. Curtis, “The Capability Maturity Model: Guidelines for Improving the Software Process, SEI Series in Software Engineering, 1995.

[Ropponen00] J. Ropponen and K. Lyytinen, “Components of Software Development Risk: How to Address Them? A Project Manager Survey”, IEEE Transactions on Software Engineering, Vol. 26, No. 2, pp. 98-112, 2000,

[Saaty80] T. Saaty, “The Analytic Hierarchy Process”, McGraw-Hill, 1980.

[Shepperd01] M. Shepperd and M. Cartwright, “Predicting with Sparse Data”, Procs. IEEE International Software Metrics Symposium, London, UK, 2001.

[Wohlin00] C. Wohlin and A. von Mayrhauser, “Assessing Project Success using Subjective Evaluation Factors”, Software Quality Journal, Vol. 9, No. 1, pp. 43-70, 2001.

Appendix A1: Analytic Hierarchy Process

This method is based on the analytic hierarchy process (AHP), which is described in detail in [Saaty80]. The description here is intentionally kept short, since the method is described in detail in the references. AHP consists of four basic steps to determine the relative importance of factors. A fifth step to analyses the consistency of the results. We use the results of Saaty's method to compute values for the aggregate success variable.

Step 1: Create an $n \times n$ matrix (denoted *N*), where *n* is the number of success variables. In the diagonal in the matrix the value "1" is inserted. The matrix is referred to as the comparison matrix. Element n_{ij}, when *i* is not equal to *j,* records the relative importance of success variable *i* versus success variable *j*.

Step 2: Perform a pairwise comparison of the success variables with respect to the importance of the success variables. An example of a scale for comparing the success variables pairwise is illustrated in Figure 4. The size of the scale varies depending on the granularity needed to differentiate importance of factors. Each pairwise comparison means that it is necessary to determine which success variables of a given pair is more important and how much more important it is. For example, a marking *x* to the left on the scale means that variable *i* is more important than variable *j*. How far to the left determines how much more important variable *i* is than variable *j*. This ranking *x* also determines the values in the matrix *N*: $n_{ij} = x$, $n_{ji} = 1/x$.

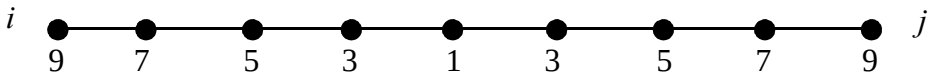

FIGURE 4. Scale for AHP.

Step 3: Compute the eigenvector of the $n \times n$ matrix. A simple method is proposed by Saaty [Saaty80] to do this; the method is known as averaging over normalized columns, and the procedure is as follows:

- Calculate the sum of the columns in the matrix, $n_j = \sum_{i=1}^{n} n_{ij}$.
- Each element in a column is divided by the sum of the column, $m_{ij} = n_{ij} / n_j$. This results in a new matrix, denoted *M*, with elements m_{ij}.
- The sum of each row in the new matrix is calculated, $m_i = \sum_{j=1}^{n} m_{ij}$.
- The sums of the rows are normalized by dividing by the number of success variables (*n*), $P_i = m_i / n$. This results in an estimation of the eigen values of the matrix, and it is referred to as the priority vector. The vector is denoted *P* with elements P_i for $i = 1 \ldots n$.

Step 4: Assign a relative importance to the success variables. The first success variable is assigned the element in the priority vector. It is said that the first success variable accounts for P_i percent of the importance of the success variables. The second success variable is assigned the second element in the priority vector and so on. Let P_1 to P_n be the percentage values for the importance of success variables *1* to *n.*

Step 5. Because AHP conducts more comparisons than minimally necessary, it is possible to evaluate the consistency of the ranking. This consistency ratio captures how consistently the pairwise comparison has been conducted. The consistency check is particularly important when a large number of pairwise comparisons are necessary, making it easier to make errors and hence become inconsistent.

The consistency ratio is computed in two steps.

- First, a consistency index (CI) is computed as $CI = (\lambda max - n)/(n-1)$, where λmax is the maximum principal eigen value of the $n \times n$ matrix. The closer λmax is to *n* the smaller is the error in the comparison. λmax is calculated by first multiplying the comparison matrix, i.e. matrix *N*, with the priority vector. Let the resulting vector be denoted *R* with elements R_i, $R = N \times P$ For the resulting vector, each element in the vector is divided by the corresponding element in the priority vector, $\lambda_i = R_i/P_i$. λmax is now computed as the average of the elements in the resulting vector, $\lambda max = \left(\sum_{i=1}^{n} \lambda_i \right) / n$. CI can now be calculated.

- The consistency ratio (*CR*) is determined by dividing the consistency index (*CI*) by a random index (*RI*). The random index has been generated to take into account randomness and it is used to normalize the consistency index. Thus, $CR = (CI)/(RI)$, where *RI* is determined from Table 8, where the first row shows the order of the matrix (*n*) and the second row shows the corresponding *RI* value. The smaller *CR*, the more consistent is the comparison.

TABLE 8. Matrix order and corresponding RI value [Saaty80].

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.45	1.51	1.48	1.56	1.57	1.59

According to [Saaty80], a consistency ratio of 0.10 or less is considered acceptable. However, [Karlsson97] points out that in practice higher values are often obtained. This indicates that 0.10 may be too hard, but it indicates an approximate size of the expected consistency ratio.

Step 6. If the consistency ratio indicates that the results are trustworthy (consistent), it is now possible to compute an aggregate success variable (sv_{agg}) for each project.

$$sv_{agg} = \sum_{i=1}^{n} sv_i \times P_i$$

In the formula, sv_i denotes the value of success variable *i* for a specific project and P_i is the relative importance given by the priority vector.

From a measurement theory perspective, this operation is problematic since the success variables are measured on an ordinal scale. For example, the distances between different values on an ordinal scale are not necessarily the same and a given value on one scale may be better or worse than the same value on another scale. A discussion on scales and use of different statistical methods in software engineering is provided in [Briand96].

Given that we map the ratio-level scale values for sv_{agg} obtained by AHP into a ranking, this should only pose a problem when scales are widely different for the individual success variables or when there are concerns about using rank order scales with AHP.

Appendix A2: Lexicographic Ordering

Lexicographic ordering (LO) is simpler than Method AHP, but it does not allow for quantifying the relative importance of the success variables with weights; it only ranks the relative importance of the success variables.

Step 1: Determine the relative importance of the success variables as a rank order. The output from this step is a ranked list of the success variables.

Step 2: Sort the projects, for which data is available, in descending order based on the most important success variable. This results in a list with the projects with the highest values of the most important success variables at the top. Within each value of the first success variable (for example all projects with a value of 5 for the first success variable), sort the projects in descending order based on the second most important success variable. This sorting is continued until all success variables have been processed. The outcome from this step is a sorted list of projects where the sorting is based on the importance of the success variables.