Using Factor Analysis to Generate Clusters of Agile Practices

(*A Guide for Agile Process Improvement*)

Noura Abbas
University of Southampton
School of Electronics and Computer Science
Southampton, UK, SO17 1BJ
n.abbas@ecs.soton.ac.uk

Andrew M Gravell
University of Southampton
School of Electronics and Computer Science
Southampton, UK, SO17 1BJ
amg@ecs.soton.ac.uk

Gary B Wills
University of Southampton
School of Electronics and Computer Science
Southampton, UK, SO17 1BJ
gbw@ecs.soton.ac.uk

Abstract— **In this paper, factor analysis is applied on a set of data that was collected to study the effectiveness of 58 different agile practices. The analysis extracted 15 factors; each was associated with a list of practices. These factors with the associated practices can be used as a guide for agile process improvement. Correlations between the extracted factors were calculated, and the significant correlation findings suggested that people who applied iterative and incremental development and quality assurance practices had a high success rate, that communication with the customer was not very popular as it had negative correlations with governance and iterative and incremental development. Also, people who applied governance practices also applied quality assurance practices. Interestingly success rate related negatively with traditional analysis methods such as Gantt chart and detailed requirements specification.**

Keywords: agile software development, agile process improvement, empirical research, factor analysis, agile practices

I. INTRODUCTION

Probably the most noticeable change to software development methodology in the last 15 years has been the introduction of the word "agile". As any area matures, there is a need to understand its components and relations, as well as the need of empirical evidence about how well agile methods work in real life settings and what are the best ways to use these methods.

Organizations apply agile software development differently, depending on their needs, resources and goals. Also, each team will choose the set of agile practices that is suitable for them. A long list of agile practices is available; each practice is related to one aspect or more of the development process. Software development teams in general and agile teams in particular need help choosing the right combination of agile practices based on their needs.

In order to group the list of agile practices into a more focused one; we decided to collect data about the effectiveness of agile practices from projects in real word. A survey will be the best way to collect as much data as possible. Before carrying out the survey, we thought of first explore existing surveys, so we do not repeat questions that were asked before. Agile adoption surveys that were conducted since 2006 (Ambler 2006) were available with their raw data so other researchers can reanalyze them.

The surveys received good number of responses (4232 responses in 2006, 781 in 2007, 642 in 2008) and they included questions that can be useful for our research, we decided to further analyze these surveys data for our research purpose.

II. AGILE ADOPTION SURVEY 2007

The 2007 survey was performed in March 2007 and received 781 responses (Ambler 2007). We chose to use this survey because it collected information not only about agile projects, success rate and iteration length, but it also included a section about the effectiveness of different agile practices. In July 2007, the results were published in Dr. Dobb's Journal and they indicated that agile techniques have been successfully adopted within the majority of organizations and often at scale. The results showed high success rate as 77% of the respondents indicated that 75% or more of their agile projects were successful.

The majority of agile teams had short iterations between one and four weeks (1 week: 17%, 2 weeks: 32.6%, 3 weeks:12.5%, 4 weeks: 21%). Regarding the effectiveness of agile practices, the high scoring practices were iterative development, regular delivery of working software, and simple design. Pair programming did not score very well. Ambler argued that this might be because many organizations do not give it enough time or because he had to distinguish between promiscuous pairing where pairs are swapped regularly and nonpromiscuous pairing when he asked the question.

III. APPLYING FACTOR ANALYSIS ON AGILE ADOPTION SURVEY 2007

Although Ambler presented the effectiveness of different practices, we needed to further explore how these practices are grouping together and how they are relating to success rate. The survey asked about 58 practices categorized in five categories: development practices, modeling and documentation practices, testing and quality practices, management and organizational practices and work product. In order to understand the structure of these variables we needed to reduce the huge data set to more manageable size while retaining as much of the original information as possible. Factor analysis (Field 2005) can be to reduce the data set (58 practices) into a set of factors by explaining the maximum amount of common variance in a correlation matrix using the smallest number of explanatory concepts.

The data was recoded using SPSS. SPSS was used as a tool for applying the analysis. First, because the software is provided by the University with introductory training, many books are available for self training, and most importantly it is a well respected tool among statisticians. In order to apply statistical methods on the current data we had to recode it into numbers using SPSS. Each practice had a 5 points scale with 5 being very effective and 1 less effective and options of "do not know" and "not applicable" which were coded as missing. This was done using a simple syntax that has to be applied on all columns we need to recode. The result is a new set of column with coded data. The frequencies of the emerging data were compared against the original ones to make sure that the recoding was done correctly. In the next section, we will explain how the factor analysis was applied and we will interpret its results.

A. Initial Considerations

Sample Size: The reliability of the factor analysis is dependent on sample size. (Kass et al. 1979) recommended having between 5 and 10 participants per variable up to total 300. (Tabachnick et al. 2001) agreed that it is comforting to have at least 300 cases for factor analysis. So a sample of 300 or more will probably provide a stable factor solution. Another way is to measure the Kaiser-Meyer-Olkin measure of sampling adequacy (KMO), which represents the ratio of the squared correlation between variables to the squared partial correlation between variables. According to (Kaiser 1974) a KMO value that is greater than .5 is acceptable, values between .5 and .7 are mediocre, values between .7 and .8 are good, values between .8 and .9 are great, and values above .9 are superb. With our sample size and a KMO of .87 as measured by SPSS, we are confident that factor analysis is appropriate for the agile adoption survey data.

Data Screening: Before running the analysis, we had to screen the data to eliminate any variables that should be excluded before the analysis is run. We can do that using the correlate procedure to create a correlation matrix of all variables. We use this matrix to eliminate variables that do not correlate with any other variables or that correlate very highly with other variables ($r<.9$) (Field 2005). In our example, we could not find any variable that fits the previous description therefore; we included all the variables in the analysis.

B. Running the Analysis and Interpreting the Results

We started with selecting the variables we need to include in the analysis. Also we calculated a number of important measures, such as KMO which is .87 in our case.

Factors Extraction: There are several methods for unearthing factors in the data. The method choice depends on the analysis purpose. When factor analysis was originally developed it was assumed that it would be used to explore the data in order to generate future hypotheses. As such, it was assumed that this technique would be applied to the entire population of interest. Such techniques assume that the sample used is the population. Principal component analysis is an example of one of these techniques. Other techniques are available for other purposes, such as the maximum likelihood method and Kaiser's alpha factoring for results generalization and the confirmatory factor analysis for testing a specific hypothesis (Field 2005).

The factor extraction gave us the component matrix were we can see that most variables load highly onto the first factor. At this stage, SPSS had extracted 15 factors. Statisticians recommend not to leave the final decision to SPSS regarding the number of extracted factors but to use its results as a guide. With a sample size over than 200 participants, the screen plot provides a fairly reliable criterion for factors selection (Stevens 1992). The screen plot shown in figure 1 is a graph of each eigenvalue against the factor which it is associated with, where the eigenvalues represents the amount of variation explained by a factor. (Kaiser 1974) recommended retaining all factors with eigenvalues greater than 1 which is a substantial amount of variation. These factors can be seen in the component matrix which contains the loading of each variable onto each factor which depends on the variable's correlation to the factor. Blank spaces can be seen see for some variables because we requested SPSS to show suppress loadings that are less than .4 to make interpretation simpler.

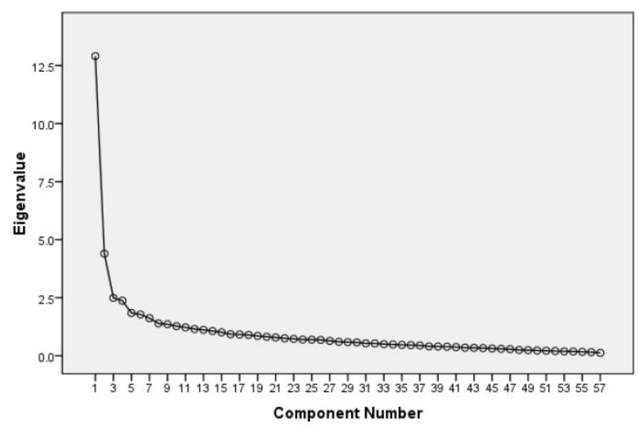

Figure 1. Screen plot for factor analysis

Factors Rotation: The interpretability of factors can be improved through rotation. Rotation maximizes the loading of each variable on one of the extracted factors which minimize the loading of the other variables. Therefore, this process makes it much clearer which variables are related to which factors. In order to decide which rotation method is more appropriate to our data, we tried to run both methods: the orthogonal rotation (varimax) and the oblique rotation. The late one produced a correlation matrix between the factors. If the components were independent then we would expect the oblique rotation to provide an identical solution to the orthogonal rotation and the component correlation matrix should be an identity one. The fact that these correlations existed told us than we could not assume independence and therefore the results of the orthogonal rotation should not be trusted and the obliquely rotated solution is more meaningful. The oblique rotation produced two matrices: the pattern matrix (Table II in Appendix) and the structure matrix (Table III in Appendix). The pattern matrix contains the factor loadings that are calculated after rotation. We can see that the rotation of the factors has clarified things considerably. The structure matrix takes into account the relationships between factors. At this stage we can look at the practices that load onto the same factor and try to identify common themes, then we double check with the structure matrix by doing the same thing (Field 2005).

TABLE I. THE EXTRACTED FACTORS AND THEIR RELATED VARIABLES

Factor1: architecture modeling • initial agile architectural modelling • initial agile requirements modelling • evolutionary design • proved architecture early	**Factor2: traditional analysis** • Gantt chart details • Gantt chart high-level • case tool modelling • architecture specification detailed • requirements specification details	**Factor3: process/governance** • burn down chart • velocity • planning game • daily stand up meeting • iteration task list • *regular status report* • *defect trend metrics*
Factor4: database practices • continuous database integration • database testing • database refactoring • data naming conventions	**Factor5: communication (team) – whiteboard Practices** • whiteboard sketches • whiteboard sketching modelling	**Factor6: agile quality assurance** • continuous code integration • test driven development • code refactoring • developer tests • *flexible architecture* • *evolutionary design* • *simple design* • *collective ownership*
Factor7: communication (team) • paper based modelling • paper models • pair programming	**Factor8: code analysis and inspection** • static code analysis • code inspection	**Factor9: lightweight testing and review** • independent confirmatory exploratory testing • customer acceptance tests • model document reviews
Factor10: architecture and configuration • architecture specification high-level • configuration management • architecture specification detailed	**Factor11: traditional quality assurance** • test plan • source code • defect reports • *regular status report*	**Factor12: coding standards** • coding standard • data naming conventions
Factor 13: lightweight requirements • requirements specification high-level • use cases light	**Factor14: incremental and iterative development** • incremental delivery • small releases • iterative development • sustainable pace • *active stakeholder participation* • *working demoable software*	**Factor15: communication (customers)** • co located team • active stakeholder participation

After studying both pattern and structure matrices, we were able to recognize the extracted factors. The identified factors are shown in table 1 were we can see each factor and the associated practices that load highly on that factor, The practices in italic have been added after considering the structure matrix. For example, the practices that load highly on factor 15 are: iterative development, incremental delivery, small release and sustainable pace which are the core of agile software development. We can call this factor iterative and incremental development. Also, the practices that load highly on factor 6 are all agile quality assurance practices: continuous code integration, test driven development, code refactoring and developers' tests. We can call this factor agile quality assurance practices. We can see that the factor analysis had re-categorized the 58 agile practices so we can study a smaller set of variables (15 compare to 58).

We can see that many practices are related to more than one factor, which is not surprising. The extracted factors can be used as a checklist in case a company or organization wants to focus on improving one aspect of the development process. If we consider the factor governance for example, the practices that formed this factor such as burn down chart, velocity, and planning game can be used as a guide for the company in order to focus on governance. An interesting factor is the agile quality assurance factor which includes all agile practices that relate to quality assurance such as continuous integration, refactoring and test driven development, where traditional quality assurance practices formed a different factor.

Factor Scores: The factor scores are another important output of the factor analysis. A factor can be described in terms of the variables measured and the relative importance of them for that factor. Therefore, it should be possible to estimate a person's score on a factor based on their scores for the constituent variables. The most use of factor scores is to reduce a large set of data into a smaller subset of measurable variables where the factor scores tell us an individual score on this subset of measures. Furthermore, we can carry out future analysis on the factor scores rather than the original data.

There are several techniques for calculating factor scores, of which the regression method preferred as it is the most easily understood one. However, the problem with this method is that it produces factor scores that are biased as they can correlate with other factor scores. There are two methods to solve this problem; the Barlett Method which produces scores that are only correlated with their own factors, and the Anderson-Rubin method that produces uncorrelated scores. In our example correlation scores are not a problem therefore the Barlett method is used. The factor scores will be added to the original data were we will have 15 new columns for the 15 new factors and now we can apply different types of analysis on the new factors (Field 2005).

IV. CORRELATIONS BETWEEN THE EXTRACTED FACTORS

In order to study any existing relationship between the different variables, correlation was used to analyze the data. Correlation is a measure of the relationship between variables, however, in order to know what type of correlation is more appropriate; we need to explore the data. Screening the data showed that that our data are not normally distributed. Therefore Spearman's correlation coefficient (rs) will be used, this correlation is nonparametric and it can be used when the data is not normally distributed. The correlation coefficient has to lie between -1 and +1, where a coefficient of +1 indicates a perfect positive relationship and a coefficient of -1 indicates a perfect negative relationship. A correlation coefficient value of ±.1 represents a small effect, ±.3 is a medium effect and ±.5 is a large effect. We have to be careful about correlation coefficients interpretation because they give no indication of the direction of causality (Field 2005).

When applying correlation between the extracted factors and success rate which was collected in the survey, we got the correlation matrix (Table V in Appendix), below we present the significant correlations for the extracted factors:

Success rate has a positive relationship with the followings other factors:

- agile quality assurance practices, rs =.16, ($p < 0.01$)
- iterative and incremental development, rs =.25, ($p < 0.01$)

Success rate has a negative relationship with the followings other factors:

- traditional analysis practices, rs = -.12, ($p < 0.05$)
- communication within the team (whiteboard practices), rs = -.16, ($p < 0.01$)
- coding standards practices rs = -.16, ($p < 0.01$)

Governance practices have a positive relationship with the followings other factors:

- architecture modeling, rs =.12, ($p < 0.05$)
- agile quality assurance, rs =.20, ($p < 0.01$)
- iterative and incremental development, rs =.21, ($p < 0.01$)
- Communication with the team, rs =.17, ($p < 0.01$)

Governance practices have a negative relationship with the followings other factors:

- Traditional quality assurance, rs = -.13, ($p < 0.05$)
- communication with the customers, rs = -.19, ($p < 0.01$)

Agile quality assurance has a positive relationship with the followings other factors:

- architecture modeling, rs =.14, ($p < 0.05$)
- iterative and incremental development, rs =.32, ($p < 0.01$)
- Communication with the team, rs =.16, ($p < 0.01$)

Agile quality assurance has a negative relationship with the followings other factors:

- communication with the customers, rs = -.11, ($p < 0.05$)
- communication within the team (whiteboard practices), rs = -.20, ($p < 0.01$)

Iterative and incremental development has a positive relationship with architecture modeling, rs =.26, (p <0.01)

Iterative and incremental development has a negative relationship with communication with customers, rs = -.11, (p <0.01)

Communication with customers has a positive relationship with communication within the team (whiteboard practices), rs =.19, (p <0.01)

According to the previous results, we can argue that people who applied iterative and incremental development and agile quality assurance practices had a high success rate. In addition, people who applied governance practices also applied agile quality assurance practices but there was not much emphasis on high communication with the customers. We have to be careful here as only two practices; co-location and active stakeholder participation contributed to the communication with the customer factor. Communication with the team factor had a positive relation with governance and agile quality assurance practices. A negative but not significant relation was found between traditional quality assurance and agile quality assurance. This maybe because agile projects have tended to abandon more traditional quality assurance practices as they move more towards agile quality assurance. Interestingly, success rate related negatively with traditional analysis methods such as Gantt chart and detailed requirements specification.

V. VALIDITY ISSUES

In this paper, we re-analyzed data from existing surveys. Although the authors did not collect the data, this survey was conducted by a well-known and respected researcher within the agile community. However, the data still has the same limitations as any survey, mainly, the collected data is self-reported, and poor memory or misunderstanding of the questions can all contribute to inaccuracies in the data (Nardi 2002). One important issue to discuss is that as the data is based on the respondent's opinions; one threat to the factor analysis results could be that people may have rated agile practices based on how effective they think they are rather than reporting their real experience. Finally, when using correlation we have to keep in mind that it gives no indication about the direction of causality. Also, none of the correlation coefficients is very close to 1 (or -1) so the correlations, though statistically significant, are relatively week.

VI. RELATED WORK

This section will look at the related work conducted by other researchers. The 2006 survey was reanalyzed by Parsons and Lal (Parsons et al. 2007). The analysis compared the impact on outcomes when using no agile methods with the outcomes when using at least one agile method. The analysis findings suggested that the adoption of at least one agile method improves the outcomes of quality, satisfaction, and productivity over the use of non-agile methods, without a statistically significant increase in cost. We analyzed the data differently as we can argue that when a company is not using any named agile method, this does not mean that they are not using agile software development. The survey results support our claim as the number of responses who said that they are not using any agile method (59%) is larger than the number of respondents who did not use any agile technique (34%).

Factor analysis was used in a study conducted by So and Scholl (So et al. 2009). The paper presented a measurement instrument to study the social-psychological effect of eight agile practices. The practices were chosen by the researchers, and then qualitative methods were used to produce a set of items for each practice which formed a questionnaire. The factor analysis, namely principal component analysis, was used to test the validity of the existed factors structure. In other words, the analysis was used to check whether the extracted factors will be the same factors (practices) introduced by the researcher. In our case, the analysis as used for a different purpose, as we did not have an initial list of factors, instead the analysis extracted 15 new factors that were identified and named by us. This restructured a large set of practices into a smaller set of factors, which made applying further analysis much easier.

VII. CONCLUSIONS

Applying the factor analysis on agile practices effectiveness data from a survey conducted in 2007 (Ambler 2007) resulted in reducing 58 practices to 15 factors presented below. Each factor is associated with a list of agile practices that can be used as a checklist when improving the related factor.

Factor1: architecture modeling
Factor2: traditional analysis
Factor3: process/governance
Factor4: database practices
Factor5: communication (team) – whiteboard practices
Factor6: agile quality assurance
Factor7: communication (team)
Factor8: code analysis and inspection
Factor9: lightweight testing and review
Factor10: architecture and configuration
Factor11: traditional quality assurance
Factor12: coding standards
Factor 13: lightweight requirements
Factor14: incremental and iterative development
Factor15: communication (customers)

The relationships between the extracted factors were studied using correlations. The results suggested that people who applied iterative and incremental development and agile quality assurance practices had a high success rate. Also, people who applied governance practices also applied agile quality assurance practices but there was not much emphasis on high communication with the customers. We have to be careful here as only two practices; co-location and active stakeholder participation contributed to the communication with the customer factor. Communication with the team factor had a positive relation with governance and agile quality assurance practices. Interestingly, success rate related negatively with traditional analysis methods such as Gantt chart and detailed requirements specification.

One way of a practical application of the previous results could be as a guide to be used by agile teams, this guide suggests that:

a) In order to improve success rate, the team can use the positively correlated factors and their associated practices. These factors are : Factor 6: agile quality assurance practices which is associated with continuous code integration, test driven development, code refactoring, developer tests, flexible architecture, evolutionary design, simple design and collective ownership. The second factor was Factor 14. Iterative and incremental development, which is associated with incremental delivery, small releases, iterative development, sustainable pace, active stakeholder participation, and working demoable software

b) The team can consider avoiding factors which are negatively correlated with success including Factor 2 Traditional analysis practices which is associated with Gantt chart details, Gantt chart high-level, case tool modeling, architecture specification detailed, and requirements specification details. Also, success correlated negatively with Factor 5 Communication within the team (whiteboard practices) including whiteboard sketches and whiteboard sketching modeling. The final factor that is correlated negatively with success is Factor 12. coding standards practices including two practices coding standard and data naming conventions.

ACKNOWLEDGMENT

We would like to acknowledge Scott Ambler for making the raw data of his surveys available for reuse by other researchers.

REFERENCES

Ambler, S. (2007). "Results from Scott Ambler's 2007 Agile Adoption Rate Survey " Retrieved 28/07/2009, from www.ambysoft.com.

Field, A. (2005). Discovering Statistics Using SPSS, Sage.

Kaiser, H. F. (1974). "An Index of Factorial Simplicity." Psychometrika **39**: 31-36.

Kass, R. A. and H. E. A. Tinsley (1979). "Factor Analysis." Journal of Leisure Research **11**(120-138).

Nardi, P. M. (2002). Doing Survey Research: A Guide to Quantitative Research Methods Allyn & Bacon.

Parsons, D., H. Ryu and R. Lal (2007). The Impact of Methods and Techniques on Outcomes from Agile Software Development Projects. IFIP International Federation for Information Processing, Springer Boston**:** 235-249.

So, C. and W. Scholl (2009). Perceptive Agile Measurement: New Instruments for Quantitative Studies in Pursuit of the Social-Psychological Effect of Agile Practices. Agile Processes in Software Engineering and Extreme Programming**:** 83-93.

Stevens, J. P. (1992). "Applied Multivariable Analysis of Variance Tests." Psychological Bulletin **88**: 728-737.

Tabachnick, B. G. and L. S. Fidell (2001). Using Multivariate Statistics, Bosten:Allyn and Bacon.

Appendex

TABLE II. Pattern Matrix

	Factor														
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
initial agile architectural modeling	.758														
initial agile requirements modeling	.756														
evolutionary design	.501														
proved architecture early	.467														
flexible architecture															
Gantt chart details		.883													
Gantt chart high-level		.845													
case tool modeling		.571													
architecture specification detailed		.502								-.449					
requirements specification details		.485													
burn down chart			.734												
velocity			.718												
planning game			.629												
daily stand up meeting			.528												
iteration task list			.514												
defect trend metrics															
regular status report															
continuous database integration				-.826											
database testing				-.777											
database refactoring				-.771											
data naming conventions				-.480								-.445			
whiteboard sketches					-.752										
whiteboard sketching modeling					-.741										
working demoable software															
continuous code integration						.654									
test driven development						.587									
code refactoring						.581									
developer tests						.465									
simple design															
collective ownership															
paper based modeling							.701								
paper models							.624								
pair programming							.427								
static code analysis								.710							
code inspection								.664							
independent confirmatory exploratory testing									.585						
customer acceptance tests									.555						
model document reviews									.454						
architecture specification high-level										-.569					
configuration management										.566					
test plan											-.678				
source code											-.541				
defect reports											-.506				

use cases details															
coding standard												-.702			
UI refactoring															
requirements specification high-level													.535		
use cases light													.460		
architectural spikes															
UI testing															
incremental delivery														.796	
small releases														.786	
iterative development														.718	
sustainable pace														.554	
self organizing teams															
co located team															-.794
active stakeholder participation															-.476

Rotation converged in 74 iterations.

TABLE III. STRUCTURE MATRIX

	Factor														
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
initial agile architectural modeling	.831														
initial agile requirements modeling	.817														
evolutionary design	.600					.464									
proved architecture early	.583													.404	
flexible architecture	.527			-.430		.446									
Gantt chart details		.875													
Gantt chart high-level		.827													
case tool modeling		.657													
architecture specification detailed		.634								-.511					
requirements specification details		.606													
burn down chart		.532									-.495		.425		
velocity			.763												
planning game			.746												
daily stand up meeting			.696												
iteration task list			.641												
defect trend metrics			.580		-.401										
regular status report			.512								-.463				
continuous database integration			.405												
database testing				-.834											
database refactoring				-.819											
data naming conventions				-.817											
whiteboard sketches				-.577								-.527			
whiteboard sketching modeling				-.520								-.478			
working demoable software					-.800										
continuous code integration					-.799										
test driven development						.689									
code refactoring						.675									
developer tests						.649									
simple design						.557									
collective ownership						.533								.461	
paper based modeling						.508			.411					.464	
paper models							.757								
pair programming					-.407		.675								
static code analysis						.449	.534								
code inspection								.748							
independent confirmatory exploratory testing								.714							
customer acceptance tests									.639						
model document reviews									.614						
architecture specification high-level	.414								.558						
configuration management	.497									-.648					
test plan										.481					
source code											-.731				
defect reports											-.607				

use cases details											-.566				
coding standard			.454								-.535				
UI refactoring												-.730			
requirements specification high-level													.596		
use cases light													.529		
architectural spikes				-.408									.440		
UI testing														.852	
incremental delivery						.403								.814	
small releases														.805	
iterative development														.670	
sustainable pace			.424											.457	
self organizing teams															-.786
co located team														.528	-.592
active stakeholder participation					-.496									.484	-.523

TABLE IV. CORRELATION COEFFICIENT BETWEEN THE EXTRACTED FACTORS AND SUCCESS RATE

	SR	**F1**	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**	**F11**	**F12**	**F13**	**F14**	**F15**
SR	1.000	.069	-.125*	.064	-.046	-.164**	.169**	-.020	-.013	.021	-.035	-.072	-.163**	.062	.257**	-.053
F1		1.000	.235**	.120*	-.273**	-.171**	.149*	.153**	.134*	.208**	-.175**	-.169**	-.123*	.096	.265**	-.105
F2			1.000	.059	-.138*	-.016	.002	.103	.214**	.176**	-.118*	-.232**	-.042	.138*	-.038	.046
F3				1.000	-.173**	-.231**	.205**	.179**	.150**	.172**	-.049	-.135*	-.043	-.023	.216**	-.192**
F4					1.000	.091	-.197**	-.186**	-.130*	-.166**	.105	.147*	.125*	-.023	-.208**	.053
F5						1.000	-.207**	-.085	-.091	-.102	.112	.126*	.124*	-.063	-.205**	.192**
F6							1.000	.164**	.141*	.062	.002	-.063	-.129*	.025	.320**	-.117*
F7								1.000	.158**	.098	-.118*	-.080	-.066	.013	.128*	-.080
F8									1.000	.151**	-.066	-.153**	-.060	.094	.067	-.042
F9										1.000	-.060	-.236**	-.042	.103	.130*	-.118*
F10											1.000	.078	-.018	-.030	-.030	.037
F11												1.000	.121*	-.128*	-.095	.021
F12													1.000	-.061	-.028	.031
F13														1.000	.104	-.042
F14															1.000	-.245**
F15																1.000

* Correlation is significant at the 0.05 level (2-tailed).

** Correlation is significant at the 0.01 level (2-tailed).

SR: Success Rate

F#: Factor#