# AUTOMATED SOFTWARE REQUIREMENTS MANAGEMENT TOOLS: A METHODOLOGY FOR PROJECT SUCCESS

[1] **Faisal Adnan,** [2] **Imran Haider Naqvi,**
[1] COMSATS Institute of Information Technology, Lahore.
[2] CIF, COMSATS Institute of Information Technology, Lahore
ei.netsolian@gmail.com
drimranhaider@ciitlahore.edu.pk

***ABSTRACT:*** *This paper provides an insight of automated software requirements management and its role in project success (PS). Different features of automated software requirements management tools were critically reviewed. The underlying associations among software requirements management, software requirements traceability, changing requirements, using automated software requirements management tools and rework with PS were explored through a survey conducted among the software houses. This study found a lack of proficiency in automated SRM skills and practices among the software projects which caused rework in software development life cycle. This study is a novel contribution in exploring the role of automated software requirements management tools as an effective methodology for project success.*

**Keywords:** *Software Requirements Management Tools, Project success, Rework, Software Requirements Management, Software Requirements Traceability, Changing Requirements.*

## 1. INTRODUCTION

Project success (PS) is achieved through effective requirements management process in scope management knowledge area. PS should be measured in terms of completing the project within scope, time, cost, quality, resource and risk [1]. PS could be ensured via project management (PM) tools and techniques. PM ten knowledge areas include integration, scope, time, cost, quality, human resource, communications, risk, procurement and stakeholder management. The ten knowledge areas and five process groups (initiating, planning, executing, monitoring and controlling along with closing) provide adequate guidelines for ensuring PS. A well-structured requirements engineering (RE) process improves the overall software productivity [2]. PS is ensured by RE which is a legitimate phase of software development life cycle (SDLC) which consists of requirement definition and SRM phases [3]. Software requirements definition phase leads to software requirements specification (SRS) document. Software requirements management process consisted of requirements documentation; requirements change management and requirements traceability [4]. SRM controls changing requirements and requirements traceability based on the SRS document given as an input to SRM process. SRS document help as an agreement of understanding between clients and project team members.

### 1.1 RATIONALE OF STUDY

Rework emerged as the most frequent burning issue which adversely affected PS. Major cause of project failure was poor SRM [5]. Literature does not provide adequate guidelines for rework reduction through the SRMT. Literature showed that SRM, SRT, CR were the rework factors closely linked with PS. An empirical research for exploring the associations with factors of rework was hence intended to quantify the role of automated Software requirements management (SRMT) in PS.

### 1.2 PROBLEM STATEMENT

PS is adversely affected by rework phenomenon. [6] Depicted that 40-100% rework was present in requirements gathering phase and the cause of rework in software projects was lack of a structured approach for SRM in SDLC.

### 1.3 RESEARCH QUESTIONS

Based on literature review the following research questions were proposed for this study.

1).What is the impact of SRM on Project Success?
2).Whether SRT on Project Success?
3).Whether CR impact Project Success?
4). How SRMT impact Project Success?
5).What is the underlying relationship of rework with Project Success.
6).How much Project Success could be ensured by SRMT.
7).How much rework could be reduced by SRMT.

## 2. LITERATURE REVIEW

Poor requirement gathering was primary cause in 37% of software projects failure during the year 2014[7]. Poor SRM was the primary cause of project failure almost half of the time, when software projects do not meet their original goals and business objectives. Majority of organizations lacked maturity in the SRM process due to lack of availability of necessary skilled workforce. The executive management, project sponsors and other project stakeholders were found reluctant to achieve excellence in SRM process. PS could be ensured by capturing valid, reliable, concise, feasible, consistent, verifiable, traceable and maintainable requirements in the scope management knowledge area. It was found that 5.1% of every dollar spent on software projects was wasted due to poor SRM which means that US$51 million was wasted for every US$1 billion spent on software projects [8]. The effect of poor SRM was even worst for low performance organizations (which completed 60% or fewer projects on time, within budget and meet original goals) in which half of the software projects were unsuccessful. High performance organizations stressed on effective SRM as a core competency for PS. Aaron et al. also found that poor communication among the project stakeholders was the primary cause of project failure which negatively affected SRM process in 75% projects more than any other area, like schedule or budget. [9] Suggested that poor SRM caused 48% of problems in SDLC. Data collected from high performance organizations (which achieved 80% or more projects on time, within budget and meet original goals), confirmed that only 11% of the projects were unsuccessful. The waste of money due to poor SRM remained within bearable limits of just 1 cent for every dollar spent in high performing organizations as compared to 10 cents loss by low performing organizations. SRM as part of project management managed constantly CR in SDLC [10]. Complete, concise and well-structured SRM process was critical for PS [11]. Poor SRM remained one of the top three critical factors of project failure. SRM helped in project team collaboration to harness innovation. [12] Reported that lack of SRM lead to project failure. Effective RM helped software teams in reducing project schedule and budget overruns. Software teams were unhappy in assigning adequate resources for SRM. A study in telecommunications and banking sectors indicated that successful projects allocated 28% overall resources and 38.6% of schedule for SRM process, while on an average 15.7% of project resources and effort was used for SRM [13].

Putting more effort and resources for SRM process increased the likelihood to meet the stakeholder demands and ensured PS. NASA projects data found that the projects which invested more than 10% resources on SRM resulted in low project cost and less schedule overruns compared to the projects which invested less effort to SRM processes and methodologies [14]. Automated SRM through SRMT facilitated in SRT which enhanced the productivity of software projects. SRT is a necessary part of effective RM in SDLC [15]. Clearly visible requirements through the SRMT improved team's communication. Centralized requirements repository provided by automated SRM through SRMT supported project teams to streamline the SDLC process. SRMT facilitated in managing the verifiability / quality of CR. Critical features which made SRMT most important for PS, varied with project nature and industry requirements. Software industry used various SRMT depending on the nature, complexity and the specific needs of the software product. Commonly used SRMT in software industry and their features are listed in Table 1.

High performance organizations recognized the importance of automated SRM processes and practices for PS. Features provided by SRMT like SRT, changing requirements impact analysis, requirements validation and coverage analysis provided a road map for PS. SRMT helped software project teams in rapid application development to stay competitive in the industry and provided fastest access to the market. SRMT helped in streamlining communication gap among project stakeholders and in tailoring the rework [16].

Effective SRM during the initial stages of the project life cycle doubled chances of PS and reduced project overruns by almost 87% [17]. Software projects faced rapidly CR throughout SDLC which caused schedule delays and budget overruns. Software teams faced lack of requirements visibility & were unable to determine rework required due to rapidly changing software requirements in the projects which caused rework in SDLC. Data showed that RM defects caused 70-85% of rework cost [18]. Rework cost upraised as software headed towards completion. [19] Suggested that during a specific reporting period in SDLC, 10-20% of rework effort were commonly accepted. Software projects faced a lot of rework which required up to 80% of the total work effort [20]. Literature showed that rework during the programming/coding phase caused 200 times more as compared with rework performed during the requirements analysis phase [21]. SRMT helped project teams to estimate CR impact on overall PS. Customer's satisfaction & effective communication among project stakeholders was found critical for PS.

SRMT facilitated in tracking both the projects as well as requirements current completion status to inform the project stakeholders about the most up to date status of requirements implementation. [22] Showed that automated SRM through the SRMT ensured 75% increase in productivity and 69% net reduction in rework cost. [23] Found that devoting more schedule and effort to SRM process yields quick and efficient delivery of software projects.

**Table 1: Commonly used SRMT features adopted from [30]**

**A.** Clear visibility of requirements to all stakeholders.
**B.** Dynamically linked requirements with different artefacts.
**C.** Permanent and secure storage location for requirements management.
**D.** Live requirements traceability/ prioritization/ addition/deletion/modification.
**E.** Requirement change management and upward/downward change impact assessment.
**F.** Integration with other tools for improved communication.
**G.** Checklist for requirements quality verification and testing.
**H.** Collaborative development of the software product.
**I.** Scalability to facilitate more end users if project team size grows.
**J.** Online repository for project related glossary terms and references.
**K.** Requirements secure import/export from other tools.
**L.** Secure system with different privileges for various stakeholders.

| **SRMT Name** | **SRMT features which helped in PS** | **SRMT features having security drawbacks** |
|---|---|---|
| **Requisite Pro** | A, B, C, D, E, F, H | G, I, J, K, L, I |
| **Case Complete** | A, B, C, D, F, J | G, H, I, K, L, I |
| **Analyst Pro** | A, B ,C, D, E, F, H, L | G, J, K, I |
| **Optimal Trace** | A, B, C, D, F, H | G, I, J, K, L, I |
| **DOORS** | A, B, C, D, E, F, I, K, L | G, J, K |
| **GMARC** | A, B, C, D, E | F, G, H, I, J, K, L |
| **Objective** | A, B, C, D, E, H, J | F, G, I, K, L |
| **RDT** | A, B, C, D, E, F, I, L | G ,H ,J ,K |
| **RTM** | A, B, C, D, E, H, | F ,G ,I ,J |

## 2.1 RESEARCH HYPOTHESES

H1: SRM is positively related with PS & negatively related with rework.
H2: SRT is positively related with PS & negatively related with rework.
H3: CR is negatively related with PS & positively related with rework.
H4: SRMT is positively related with PS & negatively related with rework.
H5: Project Success is negatively related with Rework.

## 3. THEORETICAL FRAMEWORK

[15] Said that SRT was a necessary part of SRM. [17] Found that effective SRM enhance chances of PS. While [22] found that using an internal website for automated SRM increased productivity and streamlined communication among project stakeholders. This study used SRM, SRT, CR and SRMT as independent variables and PS as dependent variable to explore their effect on the PS. Theoretical framework implied that SRMT was negatively associated with rework as shown in Figure 1.

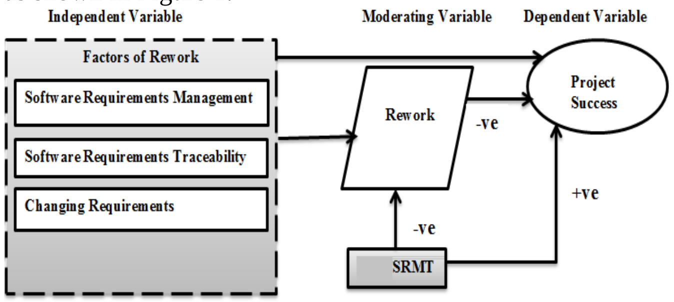


**Figure 1: Theoretical Framework**

## 4. METHODOLOGY

The research following a pilot study was carried out at the eighteen software houses. Self-administered questionnaire was distributed among randomly selected project team members. It was a correlational study. The study design was cross sectional. The research subjects were the software project team members of both accomplished and near to completion software projects of previous 5 years with documented evidence of rework and SRMT in software projects.

The study population included project team members of CMMI Level II and above or software houses with more than 15 project team members. A random sample of 224 project team members working on various software projects was selected from an estimated population of 500 [24]. The study adopted valid, pretested measurement scales from the existing literature i.e. [25] used for SRM [26] used for SRT and CR [27] used for rework while [28] scales were used for PS. The responses were collected on a 5 point Likert scale ranked from 1-5 as depicted in Table 2.

**Table.2: Coding of data for analysis and interpretation**

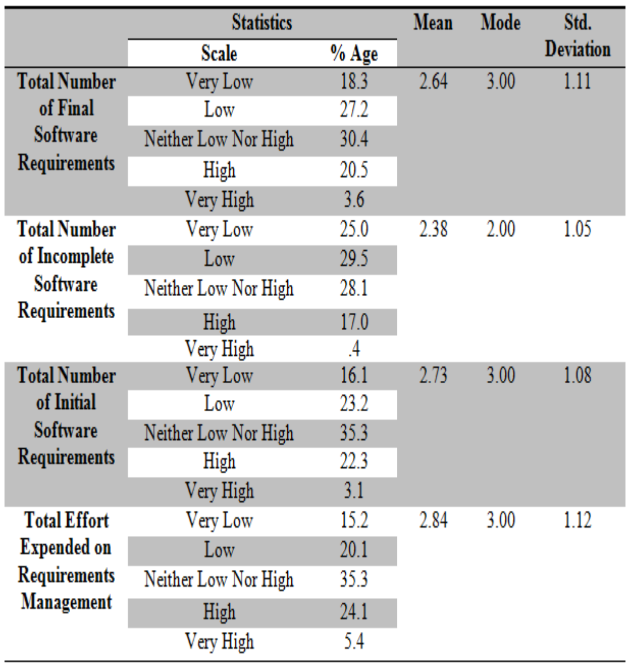
| | |
|---|---|
| Strongly Disagree/ Very Little | 1 |
| Disagree/ Little | 2 |
| Neither Agree Nor Disagree/Neither Little Nor Large | 3 |
| Agree/Large | 4 |
| Strongly Agree/Very Large | 5 |

**Table. 3: Reliability Analysis**

| | No. of Items | Cronbach's Alpha |
|---|---|---|
| Changing Requirements | 5 | 0.731 |
| Software Requirements Traceability | 2 | 0.620 |
| Software requirements Management | 4 | 0.612 |
| Rework | 5 | 0.798 |
| Project Success | 7 | 0.601 |

IBM SPSS statistics 20 was used for data analysis. Mean $\pm$ S.D was reported for SRM, SRT, CR, rework and PS. Reliability of questionnaire was checked through Cronbach's alpha. Correlation and regression analysis were used to test the research hypothesis. Table 3 showed that Cronbach's alpha values for SRM, SRT, CR, rework and PS were within the acceptable range & p-value < 0.05 was considered statistically significant.

**Table 4: Descriptive Statistics of Software Requirements Management**



| Statistics | Scale | % Age | Mean | Mode | Std. Deviation |
|---|---|---|---|---|---|
| Total Number of Final Software Requirements | Very Low | 18.3 | 2.64 | 3.00 | 1.11 |
| | Low | 27.2 | | | |
| | Neither Low Nor High | 30.4 | | | |
| | High | 20.5 | | | |
| | Very High | 3.6 | | | |
| Total Number of Incomplete Software Requirements | Very Low | 25.0 | 2.38 | 2.00 | 1.05 |
| | Low | 29.5 | | | |
| | Neither Low Nor High | 28.1 | | | |
| | High | 17.0 | | | |
| | Very High | .4 | | | |
| Total Number of Initial Software Requirements | Very Low | 16.1 | 2.73 | 3.00 | 1.08 |
| | Low | 23.2 | | | |
| | Neither Low Nor High | 35.3 | | | |
| | High | 22.3 | | | |
| | Very High | 3.1 | | | |
| Total Effort Expended on Requirements Management | Very Low | 15.2 | 2.84 | 3.00 | 1.12 |
| | Low | 20.1 | | | |
| | Neither Low Nor High | 35.3 | | | |
| | High | 24.1 | | | |
| | Very High | 5.4 | | | |

## 5. RESULTS & DISCUSSION

PS required a high level of SRM effort. This study supplemented the findings of [17] and found that a medium level of SRM effort and project resources was used for the initial SRM in 39% software projects. Medium level of total effort and project resources were allocated for SRM process in 35% projects. This study found that medium numbers of final software requirements were effectively managed in 45% projects. Poor management of incomplete software requirements caused extensive rework in SDLC. Results of this study confirmed that little magnitude of incomplete software requirements was effectively managed by software teams in 55% of surveyed projects (Mean:2.38$\pm$1.05) as shown in descriptive statistics of Table 4.

Pearson correlation coefficient data analysis of SRM, SRT and SRMT showed moderate positive correlation **(r=0.576, r=0.557, r=+0.478, p < 0.01). Si**gnificant low negative correlation was present between CR & PS. Significant moderate negative correlation was found between PS & rework (**r= -0.485, p < 0.01)** shown in Table 5.

**Table 5: CORRELATIONS**

| Correlations | Project Success | Rework |
|---|---|---|
| Software Requirement Management | .576 | -.296 |
| Software Requirement Traceability | .557 | -.346 |
| Changing Requirements | -.202 | .231 |
| SRMT | .478 | -.345 |
| Project Success | 1 | -.485 |

PS was ensured by assigning more effort and resources for CR process. This study found that addition/update related CR were common in software projects which caused extensive rework in SDLC. Survey statistics of this study found that CR related to deletion of lines of code were present in little magnitude (Mean: 1.81±0.83) which were not the major source of rework in SDLC. This study found that addition of adaptive changes in SDLC caused extensive unavoidable rework. This study found that in 77% projects overall little effort was used for requirements change process. The magnitude of correct software requirements was also reported very low in 78% projects (Mean: 1.82±0.92) as shown in Table 6.

**Table 6: Descriptive statistics of Changing Requirements**

| | Statistics | | Mean | Mode | Std. Dev |
|---|---|---|---|---|---|
| | Scale | % Age | | | |
| Software Requirements Added to Project | Very Low | 35.3 | 2.05 | 1 | 0.97 |
| | Low | 32.6 | | | |
| | Neither Low Nor High | 24.1 | | | |
| | High | 7.6 | | | |
| | Very High | .4 | | | |
| Software Requirements Deleted from Project | Very Low | 42.0 | 1.81 | 1 | 0.83 |
| | Low | 37.5 | | | |
| | Neither Low Nor High | 18.8 | | | |
| | High | .9 | | | |
| | Very High | .9 | | | |
| Software Requirements Modified in Project | Very Low | 40.6 | 1.91 | 1 | 0.93 |
| | Low | 34.4 | | | |
| | Neither Low Nor High | 19.2 | | | |
| | High | 4.9 | | | |
| | Very High | .9 | | | |
| Correct or Unchanged Software Requirements in Project | Very Low | 46.0 | 1.82 | 1 | 0.92 |
| | Low | 31.7 | | | |
| | Neither Low Nor High | 17.4 | | | |
| | High | 4.0 | | | |
| | Very High | .9 | | | |
| Effort Expended On Requirements Changed Process | Very Low | 43.8 | 1.85 | 1 | 0.93 |
| | Low | 34.8 | | | |
| | Neither Low Nor High | 14.7 | | | |
| | High | 6.3 | | | |
| | Very High | .4 | | | |

**Table 7: Descriptive statistics of Software Requirements Traceability**

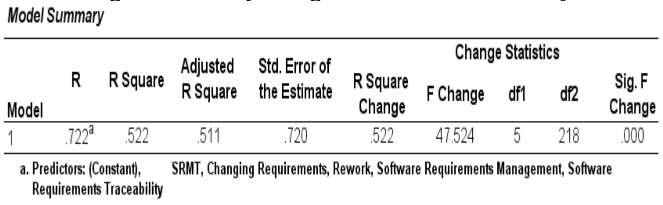
| | Statistics | | Mean | Mode | Std. Dev |
|---|---|---|---|---|---|
| | Scale | % Age | | | |
| Number of Requirements Traced Throughout SDLC | Very Low | 20.1 | 2.67 | 3 | 1.11 |
| | Low | 21.4 | | | |
| | Neither Low Nor High | 32.1 | | | |
| | High | 24.6 | | | |
| | Very High | 1.8 | | | |
| Effort Expended to Ensure that Requirements are Traceable | Very Low | 21.4 | 2.61 | 3 | 1.15 |
| | Low | 25 | | | |
| | Neither Low Nor High | 28.1 | | | |
| | High | 21.9 | | | |
| | Very High | 3.6 | | | |
| Software Requirements Traceability | Very Low | 7.1 | 2.83 | 3 | 0.96 |
| | Low | 32.1 | | | |
| | Neither Low Nor High | 34.4 | | | |
| | High | 23.7 | | | |
| | Very High | 2.7 | | | |

This study found that a low to medium level of software requirements were traceable in SDLC. SRT helped in rework reduction and projects putting less effort on SRT faced extensive rework in SDLC. Statistics of this study concluded that medium numbers of software requirements were traced throughout SDLC in 42% projects with high intensity of rework (Mean: 2.67±1.11).

Statistics clinched that in 46% projects a medium level of effort was spent to ensure that the software requirements remained traceable to all the project stakeholders. Low to medium level of effort for SRT caused high rework in software projects (Mean: 2.61±1.15) as shown in Table 7.

The multiple regression analysis model summary in Figure 2 showed that almost 52% of PS variation was ensured by SRM, SRT, CR, rework & SRMT.

**Figure 2: Multiple Regression Model Summary**



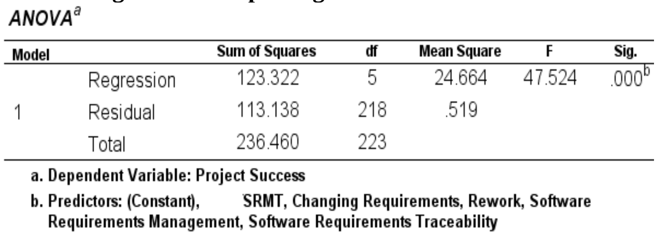
*Model Summary*

| Model | R | R Square | Adjusted R Square | Std. Error of the Estimate | Change Statistics: R Square Change | F Change | df1 | df2 | Sig. F Change |
|---|---|---|---|---|---|---|---|---|---|
| 1 | .722[a] | .522 | .511 | .720 | .522 | 47.524 | 5 | 218 | .000 |

a. Predictors: (Constant), SRMT, Changing Requirements, Rework, Software Requirements Management, Software Requirements Traceability

This study contributed that effective SRMT ensured a moderate level of PS & rework reduced significant chances of overall PS. Figure 3 found that the regression model was a good fit of the data where F (5,218) = 47.524, p<0.001.

**Figure 3: Multiple Regression Anova Statistics**



*ANOVA[a]*

| Model | | Sum of Squares | df | Mean Square | F | Sig. |
|---|---|---|---|---|---|---|
| 1 | Regression | 123.322 | 5 | 24.664 | 47.524 | .000[b] |
| | Residual | 113.138 | 218 | .519 | | |
| | Total | 236.460 | 223 | | | |

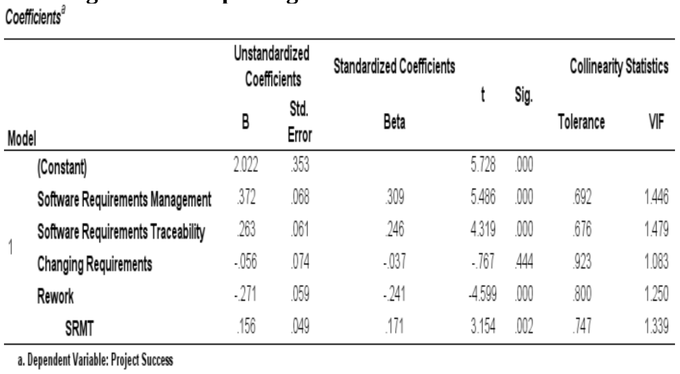
a. Dependent Variable: Project Success

b. Predictors: (Constant), SRMT, Changing Requirements, Rework, Software Requirements Management, Software Requirements Traceability

Figure 4 found that SRM, SRT, CR, rework & SRMT statistically significantly (p<0.05) predicted PS. Thus the model equation to predict PS is:

**PS = 2.02 + 0.372(SRM) + 0.263(SRT) - 0.56(CR) – 0.271(Rework) + 0.156(SRMT).**

**Figure 4: Multiple Regression Coefficients Statistics**



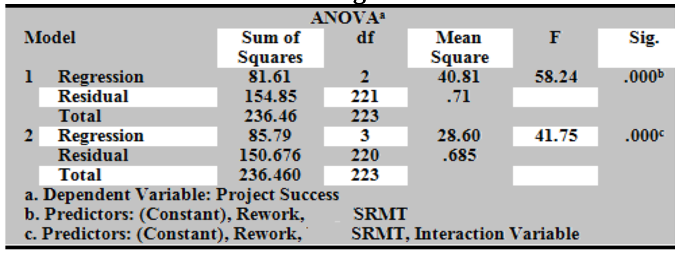
Coefficients[a]

| Model | | Unstandardized Coefficients | | Standardized Coefficients | t | Sig. | Collinearity Statistics | |
|---|---|---|---|---|---|---|---|---|
| | | B | Std. Error | Beta | | | Tolerance | VIF |
| 1 | (Constant) | 2.022 | .353 | | 5.728 | .000 | | |
| | Software Requirements Management | .372 | .068 | .309 | 5.486 | .000 | .692 | 1.446 |
| | Software Requirements Traceability | .263 | .061 | .246 | 4.319 | .000 | .676 | 1.479 |
| | Changing Requirements | -.056 | .074 | -.037 | -.767 | .444 | .923 | 1.083 |
| | Rework | -.271 | .059 | -.241 | -4.599 | .000 | .800 | 1.250 |
| | SRMT | .156 | .049 | .171 | 3.154 | .002 | .747 | 1.339 |

a. Dependent Variable: Project Success

SRM, SRT & SRMT was positively related with PS while CR & rework was found negatively related with PS. This study found that SRM helped in 08% rework reduction, SRT helped in 12% rework reduction & SRMT helped in 12% rework reduction while CR enhanced rework magnitude by 5% and decreased PS chances 4%. This study concluded that rework reduced 23% chances of PS. This study quantified that with effective SRM, SRT & SRMT up to 33%, 31% & 23% chances of PS could be ensured. Table 8 statistics concluded that the model was found significant without introducing product term of rework and SRMT. Where F (2,221) =58.24 and p-value < alpha (0.000<0.05). The model was also found significant after introducing the product term of rework and SRMT. Where F (3,220) =41.75 and P-value < alpha (i.e. 0.000<0.05).

**Table.8: Moderating role of SRMT**



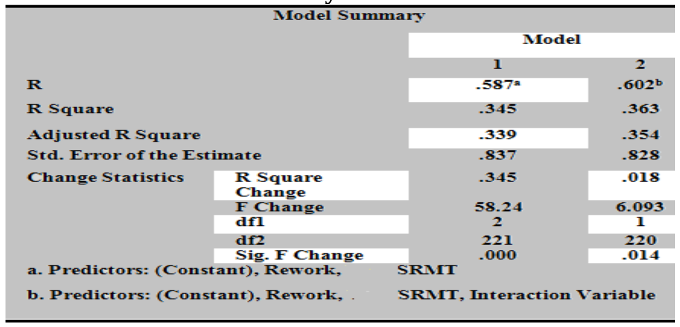
ANOVA[a]

| Model | | Sum of Squares | df | Mean Square | F | Sig. |
|---|---|---|---|---|---|---|
| 1 | Regression | 81.61 | 2 | 40.81 | 58.24 | .000[b] |
| | Residual | 154.85 | 221 | .71 | | |
| | Total | 236.46 | 223 | | | |
| 2 | Regression | 85.79 | 3 | 28.60 | 41.75 | .000[c] |
| | Residual | 150.676 | 220 | .685 | | |
| | Total | 236.460 | 223 | | | |

a. Dependent Variable: Project Success
b. Predictors: (Constant), Rework, SRMT
c. Predictors: (Constant), Rework, SRMT, Interaction Variable

The Table 9 statistics depicted that the interaction between rework and SRMT accounted for more variance. The coefficient of determination, $R^2$ change =0.018 with p-value 0.014 at 95% confidence interval. The model showed that SRMT potentially significantly moderated the relationship between rework and PS.

**Table 9: Model summary of SRMT as moderator**



Model Summary

| | | Model | |
|---|---|---|---|
| | | 1 | 2 |
| R | | .587[a] | .602[b] |
| R Square | | .345 | .363 |
| Adjusted R Square | | .339 | .354 |
| Std. Error of the Estimate | | .837 | .828 |
| Change Statistics | R Square Change | .345 | .018 |
| | F Change | 58.24 | 6.093 |
| | df1 | 2 | 1 |
| | df2 | 221 | 220 |
| | Sig. F Change | .000 | .014 |

a. Predictors: (Constant), Rework, SRMT
b. Predictors: (Constant), Rework, SRMT, Interaction Variable

## 6. CONCLUSIONS

This study is in agreement with [22] & further contributed that SRM & SRT through SRMT ensured high chances of PS. This study concluded that one per cent increase in SRM, SRT & SRMT increased almost 37%, 26 & 16% chances of overall PS while one per cent increase in the magnitude of CR & rework decreased 6% & 27% chances of PS respectively. This study also contributed that SRMT has significant moderating role between rework and PS. This study is in agreement with [16] & revealed that SRMT helped project teams to analyse the impact of CR and in streamlining communication gap between project stakeholders. This study contributed that high PS was ensured by allocating more effort and resources to CR process. This study further concluded that frequent insert/update related CR in SDLC caused extensive rework while the little magnitude of CR related to deletion of lines of code required less rework effort in SDLC. It was found that moderate level of effort and resource allocation for SRM process was insufficient for PS and rework reduction. This study conferred the findings of [17] & contributed that moderate level of PS was ensured with effective SRM, SRT and SRMT. This study conferred that a low to moderate level of SRT was insufficient for PS and caused extensive rework in SDLC. This study is in agreement with [15] & contributed that higher level of SRT through SRMT helped in rework reduction and lead to PS.

## 7. FUTURE RECOMMENDATIONS

1. Current research model determined effect of SRMT in overall PS. Future research could see the effect of SRMT among individual phases of SDLC.
2. The current research focused on analysing the overall rework role in PS. The future research could be more focused in exploring the role of various rework types in PS.
3. Future research could help in quantifying the exact amount of rework present at various stages of the SDLC.

## 8. REFERENCES

[1] Guide, A. (2013). Project Management Body of Knowledge (PMBOK® GUIDE). In Project Management Institute.

[2] Damian, D., Chisan, J., Vaidyanathasamy, L., & Pal, Y. (2005). Requirements engineering and downstream software development: Findings from a case study. Empirical Software Engineering, 10(3), 255-283.

[3] Hennicker, R., & Koch, N. (2000). A UML-based methodology for hypermedia design. In ≪ UML≫ 2000 - the Unified Modeling Language (pp. 410-424). Springer Berlin Heidelberg.

[4] Gorschek, T. (2006). Requirements Engineering supporting technical product management.

[5] Zaineb, G., & Manarvi, I. A. (2011). Identification And Analysis Of Causes For Software Bug Rejection With Their Impact Over Testing Efficiency. *International Journal of Software Engineering & Applications (IJSEA)*, *2*(4).

[6] MicroFocus. (2010). Successful Projects start with high quality requirements, 1-10.

[7] A. L. Mark. (2014). PMI's Pulse of the Profession: The high cost of low performance, A core competency for

project and program success. *Project Management Institute*.

[8] S. Aaron. P. B. David. & S. C. Tricia. (2014). PMI's Pulse of Profession: Requirements Management, A Core Competency for Project and Program Success. Project Management Institute.

[9] Hall, T., Beecham, S., & Rainer, A. (2002). Requirements problems in twelve software companies: an empirical analysis. *IEE Proceedings-Software*, *149*(5), 153-160.

[10] Shahid, M., Ibrahim, S., & Mahrin, M. N. R. (2011). An Evaluation of Requirements Management and Traceability Tools. *World Academy of Science, Engineering and Technology, WASET*.

[11] Verner, J., Cox, K., Bleistein, S., & Cerpa, N. (2007). Requirements engineering and software project success: an industrial survey in Australia and the US. *Australasian Journal of Information Systems*, *13*(1).

[12] StandishGroup. (2011). Chaos Manifesto: A Recipe for Success, Standish group international.

[13] Hofmann, H. F., & Lehner, F. (2001). Requirements engineering as a success factor in software projects. IEEE software, 18(4), 58-66.

[14] Hooks, I. F., & Farry, K. A. (2001). *Customer-centered products: creating successful products through smart requirements management*. AMACOM Div American Mgmt Assn.

[15] Nuseibeh, B., & Easterbrook, S. (2000, May). Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering* (pp. 35-46). ACM.

[16] Cass, A. G., Osterweil, L. J., & Wise, A. (2009). A pattern for modeling rework in software development processes. In *Trustworthy Software Development Processes* (pp. 305-316). Springer Berlin Heidelberg.

[17] K. Ellis. (2009). the path to success, IAG Consulting, Business Analysis Benchmark.

[18] M. T. Gail. (2014). Requirements analysis and management of cots based systems, a success story. Cots based software systems, lecture notes in computer science, 211-215.

[19] Fairley, R. E., & Willshire, M. J. (2005). Iterative rework: The good, the bad, and the ugly. Computer, 38(9), 34-41.

[20] S. E. Cross. (2002). Annual report, Carnegie Mellon University, Software Engineering Institute.

[21] Boehm, B. W. (1988). Understanding and controlling software costs. *Journal of Parametrics*, *8*(1), 32-68.

[22] IBM. (2009). Reducing rework through effective requirements management, 1-6.

[23] Blackburn, J. D., Scudder, G. D., & Van Wassenhove, L. N. (1996). Improving speed and productivity of software development: a global survey of software developers. *Software Engineering, IEEE Transactions on*, *22*(12), 875-885.

[24] Sekaran, U. Research Methods for Business: A Skill Building Approach. 2003. *John Willey and Sons, New York*.

[25] Loconsole, A. (2001, April). Measuring the requirements management key process area. In *Proceedings of the 12th European Software Control and Metrics Conference (ESCOM'2001)*.

[26] Shahid Iqbal, S. I., & M. Naeem Ahmed Khan, M. N. A. K. (2012). Yet another Set of Requirement Metrics for Software Projects. *International Journal of Software Engineering and Its Applications*, *6*(1), 19-28.

[27] Barry, M. R. (2011, March). CertWare: A workbench for safety case production and analysis. In *Proceedings of the 2011 IEEE Aerospace Conference* (pp. 1-10). IEEE Computer Society.

[28] Naqvi, S. I. H. (2007). Developing a Framework for effective IT Project Management and Best HR Practices.

[29] Bokhari, M. U., & Siddiqui, S. T. (2011, March). Metrics for Requirements Engineering and Automated Requirements Tools. In *Proceedings of the 5th National Conference*.