# Detailed Evaluation Results of RARGen[1]

*Evaluation Overview*

In the evaluation, we investigate three research questions. (1) What are the performance and scalability of RARGen in generating RARs? (2) What is quality of these mined RARs and do these RARs mined by RARGen represent real rules in practice? (3) Whether these automatically generated RARs can be used for the improvement of risk management in a "new" project from a certain domain?

## *A. Evaluation Setup*

We conduct the evaluation of RARGen on the risk-analysis documents from student term projects in the USC-CSSE[2] CS577a/b graduate software engineering course in 2007. In USC CSSE, CS577 is a project-oriented graduate software engineering course lasting for one year. All projects are real-client e-services projects collected from real world. The course involves about 20 teams per year in service learning by negotiating and developing useful software applications for USC campus and USC neighborhood community-service and small-business clients. We emphasize that the clients are real, the applications are real, the deliverables are real, and the DEN-student Independent V&Vers are full-time working professionals. Software developers are students who work on the projects part-time and they are MS/PhD-level students frequently with some work experience. The tailored management guidelines and estimation models enable them to work as professional teams. Doing so is the only way to get a critical mass of comparable projects upon which to experiment since we have not been able to find industrial companies yet that are willing or able to run 15-20 comparable projects on comparable timelines using comparable development teams and processes. All evaluations were conducted on a machine with Intel Duo Core 2.0 GHz with 2 GB memory (4 GB virtual memory).

## *B. Scalability and Performance*

We here address the first question on the performance and scalability of RARGen. To evaluate the performance of RARGen, we have used the **L**eave **O**ne **O**ut **C**ross **V**alidation (LOOCV) [1]. Based on LOOCV, we left one out of 20 projects as the "new" project and used the risk-analysis documents from remaining 19 projects as the input of RARGen. Table 1 summarizes our evaluation results. In the column "Evaluations", "Eval1" represents the 1st evaluation in which we leave Project 1 out as the "new" project. In "Eval1", RARGen learns RARs via mining the risk-analysis documents of the remaining 19 projects, that is, Project 2~20. Because we focus on text mining, Column "Input Docs Size (k)" records the number of words in the input risk-analysis documents in each evaluation. Column "# of Words" and "# of Rows" indicates the number of distinct works and number of rows/patterns in the constructed risk repository. Based on the risk repository, Latent Semantic Analysis [2][3] constructs the matrix *M* of which the size ranges from 1317*826 to 1465*889. Column "# of cluster**"** is one of the input parameters of the K-means cluster algorithm [4]. In this study, it ranges from 208 to 277.

After clustering, we employ the FP-Closed mining [5] to mine the closed frequent itemsets. The threshold is configured as 18. Column "# of CFIs" shows the number of closed frequent itemsets

[1] SMU Technical Report #09CSE01

[2] USC-CSSE: Center for Systems and Software Engineering, University of Southern California.

mined by the FP-Closed algorithm. The number of closed frequent itemsets mined from 19 out of 20 projects in each evaluation ranges from 163 to 230. There are some closed frequent itemsets containing little information about specific risk reduction information. For example, the closed frequent itemset {Backend Interface, Communication Interface} informs stakeholders about the potential risks between Backend Interface and Communication Interface of some system components. However, it does not provide any information to stakeholders of what system components (e.g., MS SQL, Apache or others specific COTS products) need to be involved in the interface analysis between their Backend and Communication Interface. This type of closed frequent itemsets becomes the noise to stakeholders.

During the generation of RARs, RARGen eliminates these noises and generates RARs based on the Risk Term File (RTF). The column "# of RTs" shows the number of risk terms in the RTF during each evaluation. "# of RARs" shows the number of learned RARs. Our results show that more than one hundred RARs can be effectively learned from these closed frequent itemsets in each evaluation without any prior knowledge or specifications from the domain experts. Compared with directly providing all these mined CFIs to the stakeholders, RARGen eliminates over 20% noise and provides stakeholders a set of RARs, which contain specific information and produce substantial value for risk reduction. The supports of all these RARs are all larger than 18, which validate the fact that risk reduction process follows many implicit rules in a project domain. In addition, RARGen learns these RARs efficiently. Column "Time(S)" shows the time of automatically learning the RARs in each evaluation. It takes no more than 3 seconds to generate all RARs for a "new" project in each evaluation with total 19 projects. RARGen is also space-efficient. It takes no more than 22 MB memory space for learning RARs as shown in the "Memory (M)" column in Table 1.

**Table 1. Characteristics of subjects used in evaluating RARGen**

| **Evaluation** | **Input Docs Size (k)** | **Latent Semantic Analysis** | | **K-means Cluster** | **FP-Closed mining** | **Risk Association Rulers Learner** | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | ***# of Words*** | ***# of Rows*** | ***# of Cluster*** | ***# of CFIs*** | ***# of RTs*** | ***# of RARs*** | ***Time(S)*** | ***Memory(M)*** |
| Eval1 | 183 | 1317 | 826 | 232 | 191 | 31 | 126 | 2.5 | 20 |
| Eval2 | 187 | 1389 | 851 | 255 | 195 | 36 | 150 | 2.7 | 21 |
| Eval3 | 179 | 1330 | 833 | 216 | 193 | 35 | 137 | 2.3 | 19 |
| Eval4 | 181 | 1429 | 871 | 239 | 198 | 33 | 129 | 2.6 | 20 |
| Eval5 | 180 | 1410 | 860 | 230 | 210 | 31 | 153 | 2.5 | 20 |
| Eval6 | 179 | 1450 | 883 | 277 | 213 | 32 | 137 | 2.2 | 18 |
| Eval7 | 180 | 1330 | 822 | 225 | 189 | 33 | 133 | 2.4 | 20 |
| Eval8 | 181 | 1401 | 841 | 208 | 192 | 30 | 146 | 2.2 | 21 |
| Eval9 | 183 | 1467 | 875 | 262 | 197 | 32 | 135 | 2.3 | 22 |
| Eval10 | 183 | 1465 | 889 | 231 | 163 | 31 | 129 | 2.3 | 21 |
| Eval11 | 181 | 1433 | 866 | 249 | 226 | 33 | 136 | 2.3 | 21 |
| Eval12 | 185 | 1340 | 827 | 223 | 192 | 35 | 122 | 2.1 | 19 |
| Eval13 | 186 | 1359 | 833 | 258 | 190 | 36 | 140 | 2.1 | 19 |
| Eval14 | 185 | 1433 | 869 | 263 | 179 | 33 | 136 | 2.3 | 22 |
| Eval15 | 181 | 1382 | 850 | 243 | 191 | 37 | 149 | 2.3 | 21 |
| Eval16 | 182 | 1377 | 836 | 268 | 220 | 33 | 157 | 2.1 | 19 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Eval17 | 183 | 1428 | 895 | 272 | 207 | 30 | 140 | 2.1 | 20 |
| Eval18 | 181 | 1431 | 872 | 239 | 221 | 31 | 121 | 2.1 | 20 |
| Eval19 | 187 | 1389 | 847 | 249 | 230 | 32 | 139 | 2.0 | 19 |
| Eval20 | 183 | 1441 | 860 | 260 | 227 | 35 | 157 | 2.1 | 21 |

*C. Quality of Mined Risk Association Rules*

This section addresses the second question on what is the quality of these RARs and do the RARs mined by RARGen represent real rules in practice. Because the selection of threshold in FP-Closed mining directly influences the quality of mined RARs, we have evaluated the quality of mined RARs based on different thresholds. We use Recall and Precision as two metrics:

$$(\mathrm{I})\,RECALL = \frac{|R_{\mathrm{min}ed} \cap R_{man}|}{|R_{man}|} \qquad (\mathrm{II})\,PRECISION = \frac{|R_{\mathrm{min}ed} \cap R_{man}|}{|R_{\mathrm{min}ed}|}$$

$R_{man}$ is the set of manually identified RARs from the risk-analysis documents in historical projects (for this evaluation purpose). In this study, we manually identified 135 risks and their reduction actions in these historical projects as the RARs. $R_{\mathrm{min}ed}$ is the set of mined RARs by RARGen. $R_{\mathrm{min}ed} \cap R_{man}$ (True Positive) indicates how many real rules in historical projects are mined. As shown in Figure 1, our approach achieves a Precision of 67% - 97% and Recall of 71% - 85% under 7 different threshold settings.

In this study, we prefer a threshold that achieves both high Precision and Recall rates (e.g., the threshold 18). However, in practice $R_{man}$ is usually unknown in prior, so Precision and Recall rates are unavailable as well. In this case, stakeholders may start with a large threshold: starting from a larger one can make users know the most frequent risk association rules in the historical projects and then verify if the risk reduction process in the "new" poject is consistent with these most frequent rules, and then decrease the threshold. Because RARs mined under a larger threshold are a subset of the ones mined under a small threshold, users just need to verify the newly mined rules as the threshold decreases.

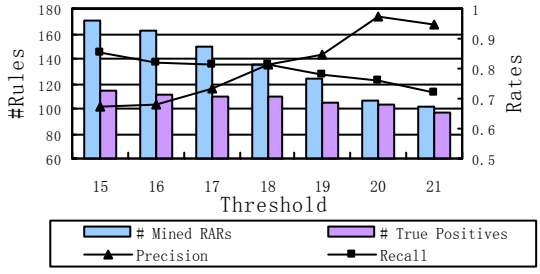


**Fig. 1. # of mined RARs, True Positives, Precision-Recall rates.**

*D. Utility of Risk Association Rules*

We next address another question on "can these automatically generated RARs be used for the improvement of risk management in a "new" project?" We discuss the utility of RARs for both top-level and detailed risks with their associated reduction actions in risk management, respectively.

**Top-level risks and reduction.** Automatically generated RARs build an evolvable risk reduction knowledge base for a certain project domain. RARs suggest a set of potential risks and

their corresponding reduction actions for future projects belonging to this domain. Even common top-level risks shared by various project domains may have domain specific reduction actions. For instance, top-level risks such as "Unstable Requirements" and "Unrealistic Budgets/zero budgets" are shared among different project domains. In our study, all projects have both. Although Boehm [1] proposed a general guideline on reduction actions for each of these two top-level risks, we found actual reduction actions adopted in these 20 e-service projects were usually more specific to the domain. Based on our mining results, two most frequently adopted reduction actions for the "Unstable Requirements" risk include "Review win conditions with clients" and "Manage clients' expectations through win-win process", which are different from the reduction actions for the similar risk in the general guideline. Therefore, the generated RARs provide stakeholders a more tangible guide to mitigate the top-level risks in a specific domain.

**Specific risks and reduction.** Because the majority of 20 projects are COTS[3]-based projects, we conduct a case study that compares the generated RARs for COTS Interoperability risks with those manually produced risk reduction plans for a "new" project and demonstrates these generated R➔R and R➔A rules can be used in tandem to detect system design defects.

We performed 20 evaluations via Jackknifing on 20 projects. In Eval14, Project 14 "Conference Room Reservation System" is selected as a "new" project. It is an e-Mentoring system for users to electronically view the availability of conference and training rooms, and to request reservations and allow staff to approve or decline reservation requests. It is a database and web-based application. Table 2 shows the reduction actions related to the interoperability risk between SQL Server and Web Application Server in Project 14.

**Table 2. Manually generated risk reduction decisions for SQL Server and Web Application Server interoperability risk in Project 14**

| | |
|---|---|
| 2.1 | Backend Interface (SQL Server) ←→ Web Interface (Web Application Server); |
| 2.2 | SQL Interface (SQL Server) ←→ Backend Interface (Web Application Server); |

**Table 3. Sample RARs mined from 20 e-service projects**

| | |
|---|---|
| 3.1 | {SQL Server, Coldfusion }➔{ Backend Interface (SQL Server) ← →Web Interface (Coldfusion)}; |
| 3.2 | {SQL Server, Coldfusion }➔{ SQL Interface (SQL Server) ←→ Backend Interface (Coldfusion)}; |
| 3.3 | {Coldfusion, IIS Server}➔{Backend Interface (Coldfusion) ←→ Web Interface (IIS Server)}; |
| 3.4 | {Web application Server}➔{Coldfusion, IIS Server}; |

The *RARs* (3.1-3.4) in Table 3 were mined from 20 student e-service projects. The *R➔A* rules 3.1-3.2 address the interoperability analysis between the MS SQL and Coldfusion. If the support-based ranking had been used, the interoperability analysis between the IIS Server and Coldfusion specified by rule 3.3 (whose support was less than the predefined threshold 18), would have not been mined as a rule. However, its subset {Coldfusion, IIS Server} (whose support is greater than 18) is a frequent closed itemset. Thanks to the FP-Closed mining, {Coldfusion, IIS Server} was mined as a closed itemset. As a clue, it can inform us that the interoperability analysis between Coldfusion and IIS Server is a risk to be addressed, which led us to include rule 3.3 as a *RAR*, whose support was 15. However, the manually specified risk reduction decisions (decisions 2.1 and 2.2 in Table 2) for project 14 did not take rule 3.3 into consideration because the *R➔R* rule 3.4 in Table 3 was not appropriately identified during the manual risk analysis process. As a result, the Web

[3] Commercial off-the-shelf (COTS) products are the software products that can be used "as-is". They are designed to be easily installed and to interoperate with other system components.

Application Server was replaced by only Coldfusion without IIS Server (see Fig. 2) in the detailed design. Hence, the interface analysis between Coldfusion and IIS Server was missing accordingly. This problem was caused by incomplete interoperability risk analysis between the *Web Server* and *Application Server* (see Fig. 3). Fig. 3 shows a design completely addressing *RARs* 3.1-3.4 based on our mined RARs.

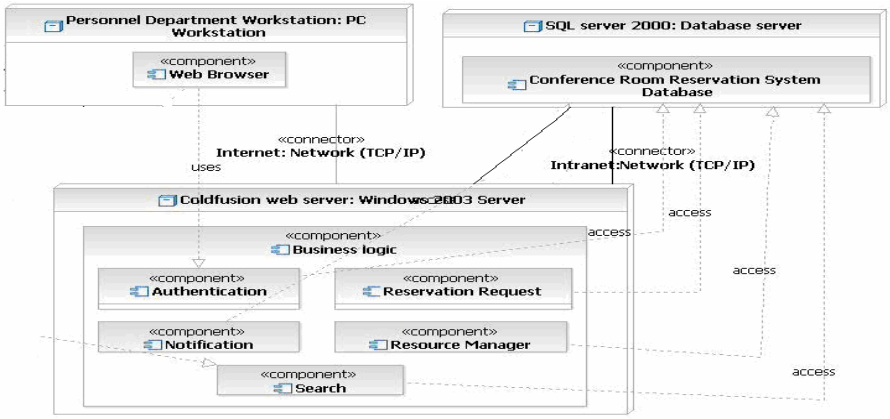


**Fig. 2. Incomplete design in Project 14.**

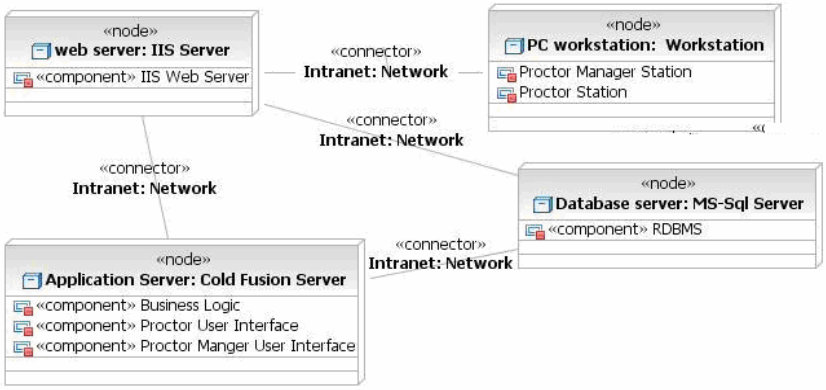


**Fig. 3. Complete design addressing RARs (3.1-3.4) in Table 3.**

In summary, through this example, we show that RARs mined from historical projects provide a useful reference on how similar risks get reduced in historical projects to verify the completeness and correctness of the current risk reduction process in future projects within the same domain.

**Table 4. System defects caused by incomplete risk reduction process.**

| Project # | # of Incomplete/Incorrect Risk Reduction Decisions | # of Design Defects | # of FP |
|---|---|---|---|
| 3 | 5 | 1 | 4 |
| 8 | 17 | 5 | 16 |
| 9 | 7 | 2 | 5 |
| 14 | 11 | 3 | 8 |
| 19 | 10 | 2 | 8 |
| 20 | 8 | 2 | 6 |

Table 4 shows the system design defects detected in the entire 20 evaluations. Column “# of incomplete/incorrect risk reduction decisions” shows the total number of manually made risk reduction decisions that do not confirm to our automatically generated RARs. Given a set of manually identified risks and their reduction actions in the “new” project, we check whether each manually conducted risk reduction decision is consistent with RARs mined from historical projects. If not, we consider it as an incomplete/incorrect risk reduction decision. Column “# of Design Defects” shows the number of system design defects detected due to the incomplete/incorrect risk reduction decisions.

There are mainly two causes resulting in incomplete /incorrect risk reduction decisions identified in 20 e-service projects through RARGen. One is due to the incomplete system component identification. The prior case study (missing the IIS Server in interface analysis) describes this issue. Many of our mined RARs contain more than three system components. For example, the mined RAR {MS SQL, Apache, Windows Server 2003}➔{Safari, JVM}, contains 5 system components and is used for communication dependency analysis in order to reduce the system component interoperability risks. It is easy for system analysts or designers to miss some components for the interface analysis with large amount of system components invovled. Such examples include the missing of Safari in project 9 and the missing of JVM in project 19. The other is due to the incomplete system component assessment so that specific interface analyses between two components are missing. In total 20 evaluations, we also detected other incomplete interface analysis based on our mined RARs, which have incurred system design defects. Such examples include missing interface analysis between Apache and Safari in Project 20.

Column “# of FP” is number of false positive, which indicates that even though the manually made reduction decision does not confirm to our generated RARs, it will not cause any design defects. When generating RARs, RARGen elaborates all RARs identified from historical project. However, not all these mined RARs need to be adopted for risk reduction in future projects. For instance, when performing interface analysis between Apache and other system components in Eval3 (Table 1), RARGen generates all RARs addressing interface analysis related to Apache including the ones related Apache and Linux, and the others related Apache and Windows in other 19 “historical” projects. However, because Project 3 is developed on a Windows-based platform, any interface analysis related to the Linux is not applicable. Thus these RARs addressing the interface analysis between Apache and Linux become the false positive. How to reduce these false positives and optimize the output of RARGen is our future work.

## References

[1] Ron. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection", Proceedings of the 14th International Joint Conference on Artificial Intelligence 2 (12): 1137–1143, 1995.

[2] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. “Indexing by latent semantic analysis” Journal of the American Society of Information Science, 41(6):391-407, 1990.

[3]T. K. Landauer, P. W. Foltz, and D. Laham. Introduction to latent semantic analysis, Discourse Processes, 25:259-284, 1998.

[4]J. B. MacQueen: "Some Methods for classification and Analysis of Multivariate Observations," Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, UC Berkley Press, 1:281-297, 1967.
[5] G. Grahne and J. Zhu. Efficiently using prefix-trees in mining frequent itemsets. In Proc. 1st IEEE ICDM Workshop on Frequent Itemset Mining Implementations, 2003.
[6] B. W. Boehm, "Software risk management: Principles and practice". IEEE Software, 8(1): 32-41.