International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.1, Issue1, pp-01-06 ISSN: 2249-6645

Software Products Risk Assessment (SPRA) Tool for Determining Source Code Risks

Armen Keshishian[1], **Hasan Rashidi**[2]

[1]*Computer science Dept, Qazvin Azad University*
Iran, Tehran

[2]*Hasan Rashidi, Qazvin Azad University*
Iran, Tehran

Abstract— **Lack of the management of software risks is one of the main reasons of software project failure. In order to implement proper risk management processes, it is necessary to evaluate risks, based on the specified criteria. The process of the assessment of risks is a time-consuming process in software engineering. So Tools for automated the risks assessment is needed. In this paper, a method for automatic evaluation of some important measures of risk management is provided. Getting the physical address of the project and analyzing line by line is how this method works. In this analysis, the risks between classes and the internal risks of any class discovered using some criteria. These criteria that are used in this method are based on three-tier architecture. Finally, proposed method, provide some quantities which are represent the impact of the risks. For showing the efficiency of this method, a tool named SPRA is implemented. At the end of this paper, a comparison between two out puts is represented, one output is based on the manual method and the second one is the SPRA tools output. These results indicate that the proposed automation method can increase the accuracy of the assessment while it is optimizing the time and avoiding human errors.**

Keywords— **Software risks, risk management, risk assessment automation**

I. Introduction

In recent years growing of the requirements in the industries, leads to increasing the complexity of the software and, in turn, it leads to amplification of the failure probability. As the proper software can guarantee the success of an industry, inefficient software can leads to the failure of the industry then. The reasons that are leading the failure of the software are called risk. Analyzing and assessment of risks can help to reduce the failure of the projects. [1] – [3]

The first step for analyzing the risks is the determination of the probability and the impact of the risk. The best way to determine the impact of risks is quantitative measurement, if the gathering of information from different resources was available. The most common way is calculating the expected monetary value. For the calculation of the EMV, the equation number 1 is used, where the EMV is stand for Expected Monetary Value. The Impact can also be calculated from the maximum of impact which is shown in equation 2 where P_i represented the probability of the maximum value. [4]

$$EMV = Probability \times Impact \qquad (1)$$
$$Impact = maxImpact \times P_i \qquad (2)$$
$$EMV = P_e \times P_i \times maxImpact \qquad (3)$$

Given the equations above, it's possible to obtain the EMV using equation 3, where P_e represented the probability of an event. There is another parameter which is called Management Reserve or MR in short. MR is summation of relative EMV's for all anticipated threats and the equation 4 is used to calculate it. This parameter used to reduce the risks. [4]

$$MR = \sum (probability_i \times impact_i) \qquad (4)$$

When threats identified and classified, the answer of the risks can be formulated and in that moment the Risk management plan will be complete. Figure 1 shows the risk management plan.

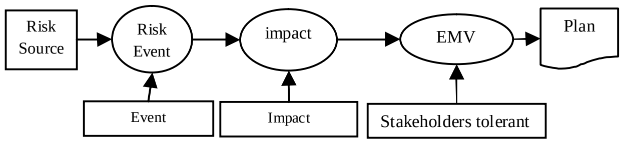


Fig. 1: Risk management plan [4]

Using some metrics for quantitative measurement is one of the most important methods in risk assessment. For that end, some important metrics that are lead to success or failure of a project should be gathered helping statistical information collection. Then for measuring the risks, the metrics should be calculated for any of the considered risks in the project. With comparison of the obtained values from the project with the values that are discovered in statistical method, the impact of any of the risks could be calculated. [4]

Now it is possible to use the impact parameter according to calculated impacts for each metric in equations 2, 3 and 4. Using these impacts and equations made it possible to give a value to each layer for assessment risks. With support of these quantitative measurements, the risk management plan will be so accurate and methodical. [4]

This paper introduces a new method that measures risk metrics in any software projects. At the end, the impacts of risks will be produced. For this goal, a tool that discovers all of the risks has been developed. With using this tool, all of the impacts of the risks will be calculated.

There is different categorization for risks in various areas. In one of these categories that represented by Hoodat and Rashidi [7] risks have been divided into two different classifications that are Internal and external risks. Internal risks take place into an organization but externals occur out of it. For external risks some issues are proposed such as the market behavior, competitions, prices, widespread failure of the product and etc. Internal risks are classified into three different categories that are product, process and project. Project risks concern the performance of the project. The product risks include technical risks, pending faults and possible shortcomings. The process risks could be the outcome of product risks. It is important to determine the relations among the risks on theses three categorize. [5]–[7]

In this paper one of the product risks is concerned that appears on the source code of the software. For avoiding some extra efforts, in the next section some similar works will be reviewed and then the new method will be introduced. At the end, the outputs of different models will be compared.

II. SIMILAR WORKS IN AUTOMATIC RISK ASSESSMENT

In this section some of the most important works that focus on discovering source codes risks will be reviewed. These efforts can be divided into two main approaches. These two approaches will be exhibited with their advantages and disadvantages.

A. DATRIX APPROACH

The name Datrix identifies a project, a set of tools and a team of engineers within Bell Canada. This approach tries to analysis the source code of the project. The aim of such an analysis is to assess the maintainability of these software products from a source code perspective. This model is based on the concept of an ASG, which stands for Abstract Semantic Graph. For generating this graph, the source code is parsed and the Abstract Syntax Tree (AST) is produced. The AST is then processed in order extract semantic information such as identifier scope, variable type and etc. This information is added to the AST as node attributes, edges or any other kind of annotations, that results in ASG production. In order to detect most risks, the ASG graph must be understandable. [8]

B. RISK ASSESSMENT USING SOURCE CODE APPROACH

For determining the risks, this approach divides the process into two phases. The information that is extracted in the first phase is generated using the automatic analysis of the source code. This information called primary information. Second phase results are called secondary information that are emerged from documents and the developers. A tool in Java language is developed to fetch the primary information using the source code analysis. The emerged information of the first phase then inserted into corresponding tables in database. In next stage, after the tool task, the analysis of the risks is on analyzers to write appropriate queries to get the required information. [9]

Table 1 shows these two approaches advantages and disadvantages.

TABLE 1
comparison between two approaches [8],[9]

Approach	Properties	Advantages	Disadvantages
Datrix	-using ASG -generating a graphical structure	-reducing reviewing process -representing a tree schema -the first graphical approach	-lack of the global view - lack of the version determining tool -lack of representing the sub graphs -tight dependency between the results and the analyzers -lack of some metrics for risk assessment
The source code analysis approach	-using database for storing the data	-reducing reviewing process -supporting the querying on the analyzed data	-high dependency between the results and the documents -high dependency between the results and the analyzers -lack of some metrics for risk assessment

III. DISCOVERING PROJECT RISKS METHOD

This paper represents a new method which uses flexible parsing [10] and source code analyzer [11]. After the analyzing phase, the risky patterns are discovered using some methods in resources [11, 12]. Then the impacts of the projects risks would be calculated. At the end of the method, with using resource [13] some documents will be generated.

The source code risks are categorized in 2 different classes:

- Intercommunicated class risks
- Single class risks

A. INTERCOMMUNICATED CLASS RISKS

For assessing the risks in a project it is necessary to discover the communicated classes and exhibit the interaction with numbers. Discovering and assigning these relations could be helpful in predicting the propagation of the changes among the classes of the project. These values can be used to determine the percentage of the changes, according to the variations percentage of the classes. This may be useful in decision support system (DSS). The accessibility of these percentages can assist managers to decide whether to accept or reject the changes. In case of acceptance, the classes that will change are determined. Two different kinds of dependencies are considered for analyzing the changes:

- Hard Dependencies
- Soft Dependencies

In this article the hard dependencies are the inheritances between two classes. These kinds of dependencies are more important than the soft dependencies, because the most tightly relation is created between these two classes when a class inherits from the other class. Therefore, the hard dependency value is higher than the soft dependency. The dependency value shows the propagation changes domain. In other words, it shows the number of affected classes. For discovering the hard dependencies in this article, the parent class is found using code analyzing method and then an inheritance relation is established among parent and the child classes.

The other kind of dependencies is soft dependency which contains the call of other class methods and usage of public members of the other classes. In this case a soft relation is established between two dependent classes. For this aim, first the public members of all classes are discovered and a list of those members is generated. After that, all of the classes will be analyzed with using the source code reviewing method. If any of items of the list is found in the class body, then a soft relation between two classes (*member owner class and user class*) will be created.

Discovering both dependencies has *O(n*m)* time complexity. First of all the whole codes are parsed, and the public members are fetched from the source code and a list is generated, this step has *O(n)* time complexity where *n* is the number of total code lines. At the second step, the code will be reviewed again and the generated list will be parsed simultaneously for each line. So the total time complexity will be *O(n*m)* where *n* is the number of code lines and *m* is the summation of the public members and the number of all classes.

B. SINGLE CLASS RISKS

As the inefficiency of a single class can threaten the whole project, it is important to assess a class without considering other classes and their relations. For automatic assessment of this kind of risks, some of metrics have to be prepared. These metrics have been gathered from previous articles. These metrics are identified by some keywords. With linear parsing and comparing each statement with these keywords is the routine of this method. In this analyzing the usage percentage of the keywords will be obtained. With these percents and using some boundary values that are represented in other articles, values which are the impacts of all risks will be calculated.

The metrics that are used in this paper are derived from resources [15], [16] and [17] where each one has its own boundary values. These boundary values are obtained from the statistically gathering data method. Every metric has its own value which represents the risk impact. Comparing the calculated value of each metric with the corresponding statistical values could lead to new value retrieval which represents the risk impact of that metric. Table 2 shows the metrics and the boundary values and the scale of each one in ten.

TABLE 2
some important metrics and the boundary values

#	Metric Name	Boundary Value	Ten Scale
1	Exception Handling Structures	Less than 5%	8
		5% to 8%	7
		8% to 12%	5
		More than 12%	1
2	The percent of using comments	Less than 10%	6
		10% to 15%	5
		More than 15%	1
3	The percent of using global variables	Less than 10%	5
		10% to 20%	6
		20% to 40%	7
		More than 40%	10
4	Number of methods	Less than 20	1
		20 to 40	3
		More than 40	5
5	The percent of using standard objects	Less than 20%	5
		20% to 40%	3
		40% to 60%	5
		More than 60%	3

IV. SOFTWARE PRODUCTS RISK ANALYSER TOOL

Software products risk analyzer or SPRA in short, is an application that is developed to automatically discover risks. The main goal of this tool is to avoid the analyzers from reviewing massive source codes. This tool identifies the whole risky patterns with given source codes then a document is created with results of this identification. This document includes all of the risky metrics of the three-tier architecture projects with their impact values. The output of this application can be generated in a short time without human errors.

The SPRA is represented, according to section 2 and the weaknesses of mentioned methods. This tool is developed by C# programming language. With obtaining the physical address of the project as an input, the parser starts the analyzing with proposed method. The parser analyzes the whole code in linear manner.

The parser creates a class object for any file that contains a class. This object is a defined class and its properties are specified based on the common characteristics between classes. For example all of classes can inherit and they have some methods, and etc. Fig. 2 shows the schema of class object and its interaction with parent class.

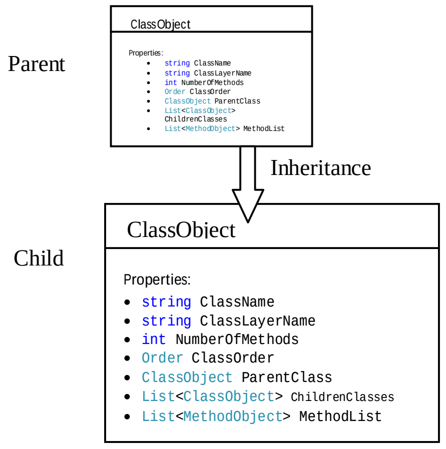


Fig. 2: the Risk Object and interaction with parent class

In this section, the risks of the layers are discussed. The purpose of using three-tier architecture is to apply the advantages of modular software development. In this architecture, any layer has its own duties; thus, there are some risks which threaten the related layer and the threat doesn't scatter to the other layers. So it is much easier to concentrate only on a layer risks instead of monitoring all risks of all layers. The SPRA tool applies this advantage and some part of it only deals with the risks of a layer.

A. Hard dependencies

There are lots of classes for any software which is developed by Object Oriented model. These classes usually have tight interconnections. One of the interconnections is inheritance relation that can cause some risks. Requirement changes or malfunctioning of a parent class could lead to the propagation of changes among wide variety of classes. For this reason, it is so vital to discover and monitor this kind of connection among classes. Figure 3 shows the inheritance between some classes and the propagation of changes in that set.

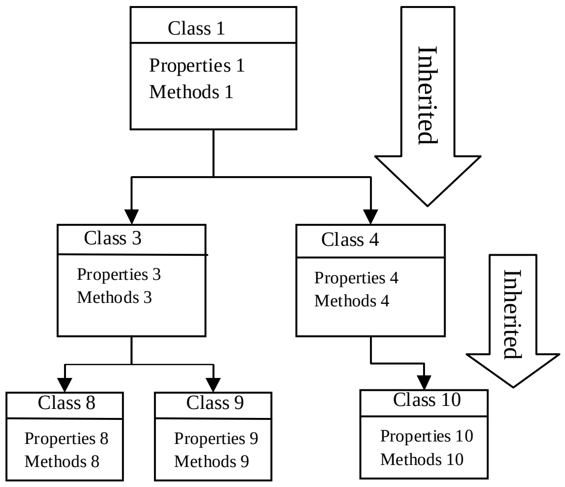


Fig. 3: the propagation of changes in inherited classes

At the beginning of class analysis, the parser identifies all [illegible] lasses is determined. For this property a field with integer type has been considered. Value zero exhibits the protected type and value one shows the public type of the class modifier. With using this value, the scale of the class could be determined. The parent of classes is determined in parallel manner while the parser checks the modifier type, and if the parent class is one of the internal classes, then two classes will be connected to each other. This connection would be as an inherited type. There is a certain property for class object which is the parent class. This property will be initialized with the parent class object. Using this information could help to the prediction of the required changes in other classes. Table 3 shows the required fields for discovering the hard dependencies.

TABLE 3
required fields for discovering the hard dependencies

#	Type	Field Name
1	Int	Modifier
2	ClassObject	ParentClass
3	List<ClassObject>	ChildrenList

B. Soft dependencies

Inheritance relations are not the only way in associating the classes. There is another kind of dependencies which is called soft dependency. For tracking this kind of connections, it has to have some certain fields in class object. To this end the UsedClasses field is considered as a list of class objects. For tracking the usage of other class methods, another field is considered as OtherClassesMethods. For used members, OtherClassesVars field is intended. With using these fields, the SPRA tool could track the propagation of changes of any class through the connected classes. This has *O(n)* time complexity. Table 4 shows the related fields in class object.

TABLE 4
related fields in class object

#	Type	Field Name
1	List<ClassObject>	UsedClasses
2	List<MethodObject>	OtherClassesMethods
3	List<Variable>	OtherClassesVars

V. SPRA COMPONENTS

For a better understanding of SPAR operation, it is needed to review its components. The core of this tool is based on three classes. In the next section these classes will be presented in details, after that a comparison between the SPRA and the manual outputs will be made.

A. BOUNDARY CLASS

According to the similar researches, the lack of the risk assessment is one of the main inadequacies. With using the measured criteria, risk assessments can be reduce the duplicated processes and even can help analyzers to manage the risks in a better way. This facility is considered in SPRA tool. SPRA uses the output of other researches in which the data is gathered from very different projects. After the comparison and value selection for any metric, the SPRA tool creates a document and reports all of the risky parameters.
In SPRA tool a class is considered for maintaining the boundary values. This class is a data structure that contains all of the keywords and corresponding values. Having this class and other components used in SPRA, makes it possible to obtain the risks of any arbitrary set of class. Table 5 shows the boundary values and some of the most important fields of this class.

TABLE 5
The boundary values and some of the most important fields of this class

Field Name	Field Type	Values
ExceptionUnder5	int	8
Exception5to8	int	7
ExceptionUpto12	int	1
CommentUnder10	int	6
Comment10To15	int	5
NOMUnder20	int	1
NOM20to40	int	3

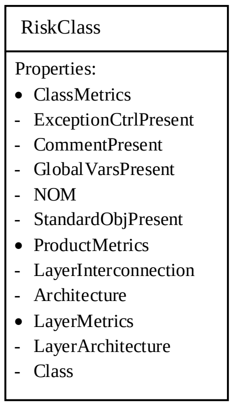
B. AVERAGE CLASS

Average class is a static class which is used for determining the average of the risks of any set of classes. This arbitrary set could be a single class or all of certain layer classes or any other combination of classes. With helping the boundary value class, the average class calculates an average value of the given classes risks impacts.

There are several methods defined in this class to make it easy to get all type of risks. For example a method is developed for determining the risks related to exception handling. With helping the boundary values and using the corresponding class, the impacts can be calculated. The average class calculates an average of these values and the result is stored in Risk Class.

C. RISK CLASS

For storing the assessed values, the risk class is introduced. This class has several fields for storing the values that will be used by managers and risk analyzers. To this end, this class uses three other subclasses. These classes are ClassMetrics, ProjectMetrics and LayerMetrics. In ClassMetrics some fields are considered for storing the assessed risks of any class object. The ProjectMetrics Class has the appropriate fields for maintaining the impacts of project level risks. The LayerMetrics class has some other fields for storing the impacts of any Layer risks. Figure 4 shows this class fields.



RiskClass

Properties:
- ClassMetrics
  - ExceptionCtrlPresent
  - CommentPresent
  - GlobalVarsPresent
  - NOM
  - StandardObjPresent
- ProductMetrics
  - LayerInterconnection
  - Architecture
- LayerMetrics
  - LayerArchitecture
  - Class

Fig. 4: Risk Class Fields

D. THE COMPONENTS INTERACTION

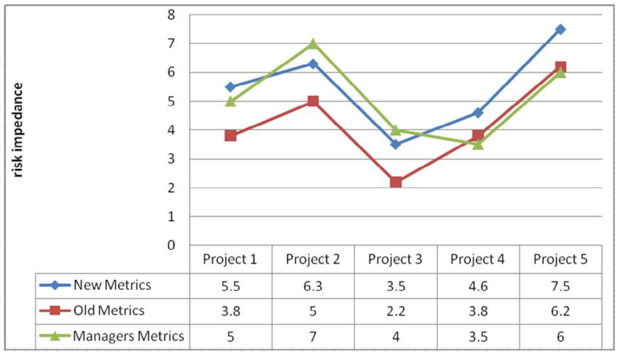
For better understanding to how this method works, the pseudo code 1 is represented. As pseudo code 1, at the first phase of the tools execution all of the required classes are added to AllClassesList object. After initializing this object a Risk Class object is created. Then for each item on AllClassesList the criteria of risks is calculated and the corresponding field is initialized. This initialization is handled by the Average class. The Average Class applies the boundary value class. Furthermore, using boundary values for each metric, the impact can be determined and stored in the corresponding subclass. Finally, using the documenting methods, a document is stored in the physical address of the project. This document contains all of the information that can be usefully in risk assessment and analyzing.

```
List<ClassObject> AllClassesList = initializing the favorite set of classes;

RiskClass riskClass = new RiskClass();

foreach (ClassObject cls in AllClassesList)
{
   riskClass.ClassMetric.ExeptionCtrlPresent =
                                   Average.ExeptionCtrlPresent(AllClassesList);
   riskClass.ClassMetric.CommentPresent =
                                   Average.CommentPresent(AllClassesList);
   riskClass.ClassMetric.GlobalVarPresent =
                                   Average.GlobalVarPresent(AllClassesList);
   riskClass.ClassMetric.NOM = Average.NOM(AllClassesList);
   riskClass.ClassMetric.StandardObjPresent =
                                   Average.StandardObjPresent(AllClassesList);
   riskClass.ProductMetric.LayerInterconnection =
                                   Average.LayerInterConnection(AllClassList);
   riskClass.LayerMetrics.LayerArchitecture =
                                   Average.LayerArchitecture(AllClassList);
   riskClass.LayerMetrics.Class = Average.Class(AllClassList);

}
```

Pseudo code 1: the interaction between components

VI. Comparison between SPRA and other methods

For measuring the performance of the SPRA tool, some sample projects of Kaloob Engineering Corporation is selected. The Kaloob Corporation is a software development company that develops GIS based systems. For comparison, three different methods are considered. First method is the SPRA method output and the other one is based on the old methods that were mentioned, and the last one is the manual method that is implemented by the developers and managers. The old methods are obtained from references [11] and [12]. The SPRA metrics and boundary values are obtained from reference [10] and the manual metrics are utilized by managers and the risk analyzers. These comparisons are made using five different projects. Projects number 1, 2 and 3 are web-based and the others are windows-based. The best method is the one that its results are closer to the manual outputs. Figure 1 shows the result.



	Project 1	Project 2	Project 3	Project 4	Project 5
New Metrics	5.5	6.3	3.5	4.6	7.5
Old Metrics	3.8	5	2.2	3.8	6.2
Managers Metrics	5	7	4	3.5	6

Fig. 5: some different methods outputs

According to this Diagram, the outputs of the SPRA tool are accurate enough to be used in real environments. This means that the SPRA could avoid the human errors while it saves the assessment time. It can also be replaced with the manual assessing methods.

VII. Conclusion

In this article a method is introduced that can estimate the risks of software projects. The SPRA tool is developed based on the represented method to automate the risk assessments processes. The outputs show, SPRA could be replaced with the time-consuming manual methods. The output diagram shows that the type of the project could play an important role in risk assessment; thus, gathering the boundary values of impacts based on the project type could be one of the future researches.

References

[1] B. Xinlong, "Software Engineering Failures: A Survey". Oregon State: University Corvallis, School of EECS, 2001.
[2] B. Lawhorn. (2009) Software Project Failure Costs Billions. Better Estimation & Planning Can Help. [Online]. http://www.standish.com,
[3] W. Humphrey, "Five reasons why software projects fail". Addison-Wesley, 2002.
[4] J. Kontio, "The riskit method for software risk management", Institute for Advanced Computer Studies and Department of Computer science, University of Maryland, 1999.
[5] R. Pressman, *Software Engineering: A Practitioner's Approach*. 7th ed., McGraw-Hill Science Engineering, 2009.
[6] H. Ronald, P. Haimes and Y. Yacov, "Software Risk Management". University of Virginia: Software Engineering Institute, Center for Risk Management of Engineering, 1996.
[7] H. Hoodat, H. Rashidi. "Classification and Analysis of Risks in Software Engineering" . World Academy of Science, Engineering and Technology 56, 2009.
[8] Lapierre, Sebastien, Lague, Bruno and Leduc, Charles. "Datrix Source Code Model and its Interchange Format: Lessons Learned and Considerations for Future Work. Montreal", Canada: Bell Canada, Quality Engineering and Research, 2002.
[9] van Deursen, Arie and Kuipers, Tobias. "Source-Based Software Risk Assessment". Netherlands: CWI and Delft University of Technology the Netherlands, 2004.
[10] G. Knapen, B. Laguë, M. Dagenais,E. Merlo, "Parsing C++ Despite Missing Declarations", International Workshop on Program Comprehension, May 99, Pittsburgh, PA, USA.
[11] M. Kuhnemann, T. Rauber, G. Runger, "A source code analyzer for performance prediction", Parallel and Distributed Processing Symposium, 18th International, 2004.
[12] Li, Z. Lu, S. Myagmar, S. Zhou, "finding copy-paste and related bugs in large-scale software code", Software Engineering, IEEE Transactions on, March 2006.
[13] Victor R. Basili and Salwa K. Abd-El-Hafiz, "A method for documenting code components", Journal of Systems and Software, Volume 34, Issue 2, August 1996, Pages 89-104.
[14] H. Koziolek and F. Brosch, "Parameter Dependencies for Component Reliability Specifications", Sixth International Workshop on Formal Engineering approaches to Software Components and Architectures, FESCA, 2009.
[15] A. Keshishian, H. Rashidi, "Assessing several Metrics for risk management in software", Sixteenth conference of Computer science, Sharif University of Tehran., 2010.
[16] A. Stale, "Risk-based testing: Risk analysis fundamentals and metrics for software testing including a financial application case study", Systems and Software, pp. 287-295. 2000.
[17] L. Rosenberg, H. Stapko and G. Albert. "Risk-based Object Oriented Testing", SATC NASA, Unisys, 2000.