ARMOR: Analyzer for Reducing Module Operational Risk

Michael R. Lyu
Jinsong S. Yu
Elaine Keramidas
Siddhartha R. Dalal
Bell Communications Research
Morristown, New Jersey 07960

Abstract

ARMOR (Analyzer for Reducing Module Operational Risk) is a software risk analysis tool which automatically identifies the operational risks of software program modules. ARMOR takes data directly from project database, failure database, and program development database, establishes risk models according to several risk analysis schemes, determines the risks of software programs, and display various statistical quantities for project management and engineering decisions. Its enhanced user interface greatly simplifies the risk modeling procedures and the usage learning time. The tool can perform the following tasks during project development, testing, and operation: (1) to establish promising risk models for the project under evaluation (2) to measure the risks of software programs within the project, (3) to identify the source of risks and indicates how to improve software programs to reduce their risk levels, and (4) to determine the validity of risk models from field data.

1: Introduction

Risks and problems are strongly related. The relationship between risks and problems is similar to the relationship between faults and failures. A risk is a potential problem and a problem is a risk that has manifested. As the system gets more complex, the software failure behavior becomes more intricate, and the criticality of software component and its internal modules raises. In order to reduce the risk of software operations, the software modules which have the potentials to cause system problems have to be identified early[1].

Software risk is expressed by the potential number of failures that may be caused by a module, as well as the severity and intensity of these failures. Before software failures happen, there is a need to model and predict the level of risks contained in software systems or modules. To model software risk the quality indicators of software modules, usually in the form of metrics, are required. The key elements in software risk analysis, therefore, are the implementation of software metrics, the application of software measurement, and the validation of the measurement results to establish pertinent metrics [2], [3]. The validated software metrics could be applied to form various models for the early detection of software risk and risky modules. Based on these approaches, much research work has been performed in identifying error-prone software and managing risk early in software development [4], [5].

To date there is a lack of validated software risk models and systematic approaches to identifying and reducing the operational risk of software modules. As a result, it is urgent to acquire appropriate software tools which can automate the procedure for the collection of software metrics, the selection of risk models, and the validation of established models. For this purpose, we are prototyping a software risk analysis tool, called Analyzer for Reducing Module Operational Risk (ARMOR), for an automatic and systematic approach to software risk management.

2: Objective and Overview of ARMOR

Many commercial tools are currently available for the measurement of software metrics and establishment of quality index of software programs. However, few tool can both perform sophisticated risk modeling and validate risk models against software failure data by various statistical techniques. ARMOR is designed to provide the missing link.

The objectives of ARMOR are summarized as follows:

(1) *To access and compute software data deemed pertinent to software characteristics.* ARMOR accesses

three major databases: project source code directory (for *product*-related metrics), program development history (for *process*-related metrics), and the Modification Request (MR) database (a failure report system at Bellcore).

(2) *To compute product metrics automatically whenever possible.* By measuring the project source files, ARMOR directly computes software code metrics related to the software product.

(3) *To evaluate software metrics systematically.* A preliminary analysis of the effectiveness of the computed and collected metrics is obtained by studying the correlation of these metrics to the software failure data in the MR database. This study provides information about the candidate metrics for the establishment of risk models.

(4) *To perform risk modeling in a user-friendly and user-flexible fashion.* Metrics are selected with appropriate weighting to establish risk models, which can compute quantitative risk measures (i.e., risk scores) of each software module. Several modeling schemes are provided in ARMOR. Risk models could be defined, removed, and executed easily at the user's discretion.

(5) *To display risks of software modules.* Once computed, risk scores computed the risk models could be used to highlight each software module by different colors. Risk distribution can be demonstrated in various forms.

(6) *To validate risk models against actual failure data and compare model performance.* Using several validation criteria, the risk models are compared with actual failure data to determine their predictive accuracy. Model validation results are provided in a summary table. Validated models could be saved for a later reusage.

(7) *To identify risky modules and to indicate ways for reducing software risks.* Once a valid model is established, the risk score computed for each module can be compared with the risk score contributed by the individual metric components. This process is iterated to identify the dominanting metrics which need to be addressed for the reduction of module operational risk.

ARMOR is designed as a software risk modeling and analysis tool that addresses the ease-of-use issue as well as other issues. ARMOR is currently implemented in a UNIX X-windows environment, using Extended Tcl/Tk as its interface builder. By enabling pull-down menu options, ARMOR allows users to apply the software metrics deemed important to software risks, to establish various risk models, and to compute module risks. After the risk models have been established and executed, the predicted risk of each module is displayed with a color to represent the risk of the module. Users can also display various statistics on the distribution of software risks. Finally users can apply regression analysis and other statistical techniques to determine the validity of the risk models. The validated risk models in turn can be saved in a model repository for their applications to another project release or a completely different project.

Figure 1 shows the high-level architecture for ARMOR.

3: ARMOR Context

ARMOR is composed of seven major functional areas:

1. File Operations ("File" menu)
2. Selecting Scope ("Scope" menu)
3. Computing and Selecting Metrics ("Metrics" menu)
4. Model Definition and Execution ("Models" menu)
5. Risk Evaluation ("Evaluation" menu)
6. Model Validation ("Validation" menu)
7. Help System ("Help" menu)

Although the complete functionality of ARMOR is not yet implemented, the major portion of ARMOR is available to demonstrate its capability.

3.1: File operations

There are four operations that can be performed by selecting items from the **File** pull-down menu. These are: "Project Open", "Load Process History", "Load Failure Data", and "Exit". "Project Open" navigates through the directories of projects to select the project for risk analysis. Upon selecting a particular project directory, ARMOR automatically computes the structure diagram of the project and displays the calling graph (dependency graph) relationships among program modules within the project directory. Module development history, including program module names, build date, inspection status, application function area, number of extensions, etc., is brought up by clicking "Load Process History" button. The MR failure database is displayed by the selection of "Load Failure Data". Finally "Exit" provides the normal exit to the risk analysis procedure.

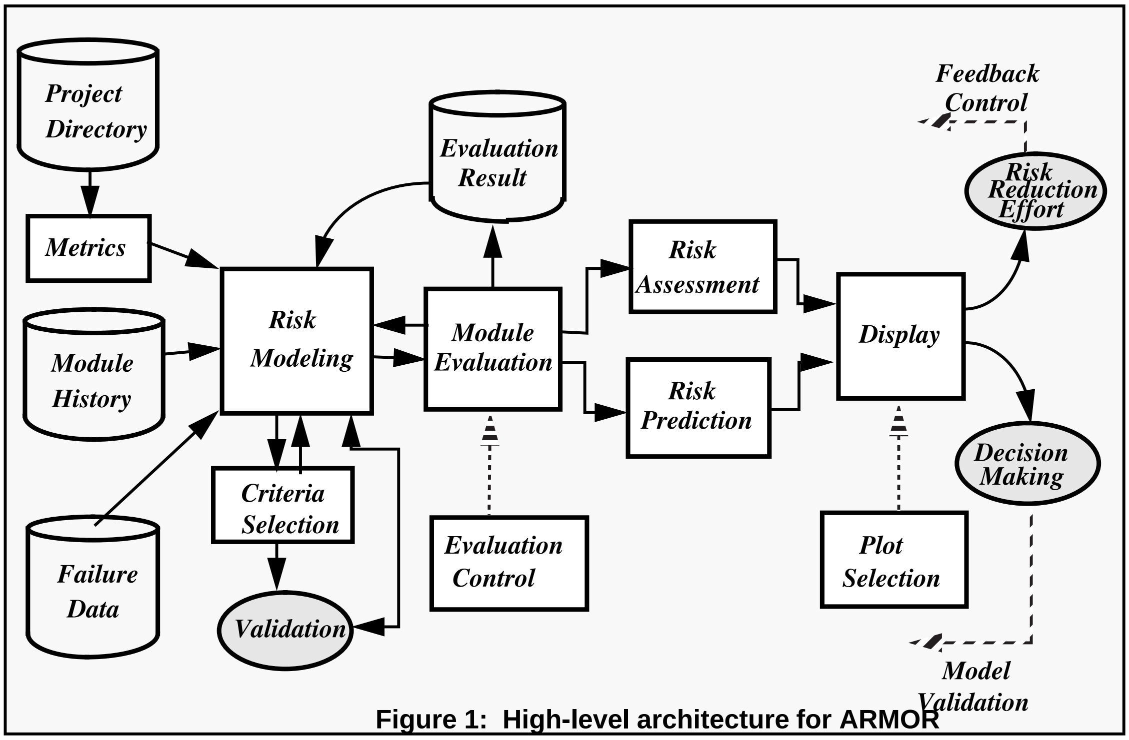

Figure 1: High-level architecture for ARMOR

3.2: Selecting Scope

Users have available to them four **Scope** operations which allow them to determine the scope to which risk analysis is performed. These are:

1. Project – the risk analysis is performed to all the modules within the project.

2. Subsystem – the risk analysis is performed to a particular subsystem of the project. A subsystem is typically identified as a sub-directory within the project directory.

3. File – the risk analysis is performed to the modules within a file.

4. Module – this risk analysis is only performed to a particular module.

Selection of "Project" and "Subsystem" scope involves global application of risk analysis to a large number of modules, where statistical validation of risk models could be achieved. Selection of "File" and "Module" level scope, on the other hand, is usually made when users want to focus the validated risk models on some particular modules (e.g., newly patched modules) in their mind.

3.3: Computing and Selecting Metrics

There are two types of operations in the **Metrics** pull-down menu: The first type of operations, including "Metrics Computation" and "Metrics Selection", involves metrics calculation alone. These operations allow users to compute product-related metrics, and select available product and process metrics for preliminary validation. The second type of operations, including "MR Selection", "Regression Schemes", "Correlation Computation", and "Summary", involves correlation study between metrics and failure data (if available), or among the metrics themselves. These operations allow users to compute the correlation between the selected metrics and a chosen subset of the software failure data, under certain regression scheme and evaluation criteria. They also allow users to apply statistical discriminant analyses to detect redundant metrics (i.e., metrics that are strongly correlated among themselves). The comparisons between different metrics in their predictive capability are summarized.

3.4: Model Definition and Execution

The **Models** operation allows users to choose metrics with appropriate weighting schemes, to construct various forms of risk models, to select and apply the established models to the project modules, and to display the risk analysis results.

1. Selection and Weighting – allows users to select and weight one or more candidate metrics as the basis to form risk models.

2. Model Definition– allows users to define risk models and save the models in an inventory, which might have contained previously validated models. These historical or user-defined models remain available during the current and subsequent sessions. Three modeling schemes are available for users to construct risk models: (1) "Summation Form" establish a risk model by using the sum of the selected metrics; (2) "Product Form" constructs a risk model by using the product of the selected metrics; and (3) "Tree Form" uses a classification tree to classify the risk of a software module according to the value range of the metrics.

3. Model Edit/Model Removal – allows users to change or remove descriptions of risk models that were previously created using the "Model Definition" capability.

4. Model Execution – allows users to select risk models from the model inventory to compute risk scores of the modules.

5. Display – allows users to display computation results from various risk models.

6. Automation – allows users to automatically search for the best risk models under a particular modeling scheme. This operation contains iterative and intensive computations involving the following two main menu items, **Models** and **Validation**.

3.5: Risk Evaluation

Four operations are available for displaying the risk evaluation results in the **Evaluation** menu. "Ranking" displays the risk scores and the order of risks among the software modules within the selected scope for analysis. "Highlight Risky Module" paints colors to each module to show its associated risk level. "Data Table" lists all the computed metrics values and risk scores among the modules under selected risk models. "Distribution Plot" plots various distributions of the risk scores for an overall project risk analysis.

3.6: Model Validation

Similar to the **Metrics** menu, the **Validation** menu include two types operations. The first type of operation is "Model Selection", which allows user to select computed risk models for overall validation. The second type of operations, including "MR Selection", "Regression Schemes", "Correlation Computation", "Validation Criteria", and "Summary", involves correlation study between risk models and failure data (if available). These operations allow users to compute the correlation between the selected models and a chosen subset of the software failure data, under certain regression scheme and evaluation criteria. The comparisons between different models in their predictive capability are summarized in a summary table. Model Validation operations are available only when the software failure data are accessible.

3.7: Help System

The **Help** system provides context-sensitive online assistance to users by allowing them to search for and read descriptions of the major ARMOR functional areas.

4: ARMOR On-Screen Appearances

Figures 2-7 show a series of screen dumps for the described ARMOR tool. It can be seen that the application of risk modeling and analysis to software modules is a straightforward process. Users are also given a considerable amount of choices in constructing and applying risk models. This combination of simple operations and variety in risk models makes it easy for users to identify an appropriate risk model for a particular development effort.

4.1: Open Databases

This screen is shown in Figure 2. To choose any of the three available databases, users select the "File" menu with the mouse, upon which a dialogue box for selecting a project directory appears on the screen (not shown here). After selecting the project, the structure chart of the project is computed and displayed in the main window as shown. Similarly, "Load Process History" and "Load Failure Data" options display failure data (shown at the bottom window) and process history data (shown in the middle window), respectively. The main window can be zoomed in or zoomed out to view different part of a large project structure.

4.2: Select and Compute Metrics

The screen is shown in Figure 3. After selecting the scope of risk application from the "Scope" menu, users can select metrics to form risk models. Upon clicking "Metrics Computation" option, a dialogue box appears for the users to choose product-related metrics for computation (shown in the middle). These metrics are later combined with the process-related metrics (extract from the project history database) to form a list under "Metrics Selection" menu for the users to use in risk modeling. These metrics are subject to statistical discriminant analyses to detect redundant metrics.

4.3: Select Metrics and Criteria

The screen in Figure 4 shows the selection of metrics and weighting criteria before constructing risk models. Users then can click on "Selection and Weighting" option from the "Models" menu to select metrics and their weighting criteria. A dialogue box appears for the users to choose the percentage of each metric when they are used to form risk models. The selections of weighting criteria are easily controlled by scale buttons.

4.4: Construct Risk Models

The screen is shown in Figure 5. Now the users can construct risk models according to different schemes. Shown in the figure is the selection of "Tree Form" from the "Models" menu, upon which a separate window appears on the lower left-hand corner. In this case the users have constructed a classification tree to determine risk modeling criteria which will be applied to the software modules. The risk modeling could lead to a binary decision ("positive" for no risk, "negative" for risk as shown here), a multiple-level decision ("very high risk" to "no risk"), or a risk score index value. Since the classification tree could be large and complicated, zoom-in and zoom-out facilities are provided for better viewing of the model.

4.5: Evaluate Models

The screen is shown in Figure 6. In this figure the users have clicked on "Highlight Risky Module" to display the risk level associated with each module by coloring it. The users also have selected "Distribution Plot" option to plot the distribution of the number of modules with respect to various risk categories, which could be either risk levels or risk scores (shown here).

4.6: Validate Models

The screen is shown in Figure 7. Finally users can select the risk models for validation against actual data recorded in the MR failure database. After selecting appropriate "Regression Scheme" and "Validation Criteria", the users then click on "Correlation Computation" to compare the predictive risk models and the actual number of MRs (or MR density) in each module. It is assumed that a good risk model should produce a module risk score which is strongly correlated with the number of MRs in that module. After the correlation is computed, the regression plot can be displayed for each model, as shown in the middle right-hand side of the figure. Users can also select the "Summary" option to display the overall comparisons among the selected models for validation, as seen in the model validation result window.

5: Conclusions and Extensions

The Tree Model component of ARMOR has been applied to a telecommunications system for risk analysis of its 1254 program modules, with 4.6 million lines of code in total. Upon its full implementation, ARMOR will provide extensive support to software risk modeling and measurement in a flexible and friendly user interface. ARMOR is designed as a tool that is easy to learn and to use. In addition, ARMOR can also be linked with reliability modeling tools to measure module-level reliability and construct system-level reliability. In contrast to these black-box reliability tools, however, ARMOR is perceived as a gray-box modeling tool which can perform reliability and risk analyses according to the structure of software systems.

References

1. R. Chillarege and S. Biyani, ‘‘ Identifying Risk Using ODC Based Growth Models,’’ *Proceedings of 1994 International Symposium on Software Reliability Engineering* , Monterey, California, November 1994.
2. S.L. Pfleeger, ‘‘Lessons Learned in Building A Corporate Metrics Program,’’ *IEEE Software*, May 1993, pp. 67-74.
3. N. Fenton, ‘‘Software Measurement: A Necessary Scientific Basis,’’ *IEEE Transactions on Software Engineering*, Vol. 20, No. 3, March 1994, pp. 199-206.
4. R.W. Selby and V.R. Basili, ‘‘Analyzing Error-Prone System Structure,’’ *IEEE Transactions on Software Engineering*, Vol. 17, No. 2, February 1991, pp. 141-152.
5. J.C. Munson and T.M. Khoshgoftaar, ‘‘The Detection of Fault-Prone Programs,’’ *IEEE Transactions on Software Engineering*, Vol. 18, No. 5, May 1992, pp. 423-433.