Change Impact Analysis for the Software Development Phase:
State-of-the-art

Nazri Kama
Advanced Informatics School, Universiti Teknologi Malaysia, Malaysia
nazrikama@ic.utm.my

Abstract

Impact analysis predicts the parts of the software system that can be affected by changes in the system. Selecting the approach to perform impact analysis in the software development phase requires consideration of the class artifact status. Static analysis approaches do not require class artifacts to be fully developed. However, the analysis tends to overestimate the number of affected classes. So it is of little practical use. On the other hand, dynamic analysis approaches expect all the class artifacts to be fully developed. So the analysis tends to be imprecise when not all class artifacts are fully developed. This paper reviews current impact analysis techniques’ capability from the perspective of supporting the software development phase implementation. Based on this review, the needs of a new impact analysis technique for the software development phase are then constructed.

Keywords: Impact Analysis, Software Development Phase, Traceability

1. Introduction

Impact analysis is the activity of estimating the effect either before or after making a set of changes to a software system [1, 2]. If the estimation is conducted before making changes, the impact analysis helps software developers to determine components of software to be affected. This estimation is also known as a predictive impact analysis. Conversely if the estimation is conducted after making changes, the impact analysis guides software testers to run regression test efforts by reducing the set of test cases to be run to those cases that traverse the change.

Conducting IA activity requires different approaches and implementations [3] for different software phases such as software maintenance and software development. Most existing IA work in the software maintenance phase are CI analyses [4-7] where the main artifact of analysis is the class. Unfortunately, in the software development phases, where the software is still under development, not all the classes are completely developed [4]. Therefore, the existing IA approaches are not suitable for software development phases.

There is a major challenge for the current impact analysis techniques to be applied for the software development phase which is the existence of the partially developed class in the class artifacts. Therefore, this paper provides a comprehensive review on the current impact analysis techniques capabilities. This review identifies which capabilities can be used and which capabilities can be improved from the current techniques to support impact analysis implementation for the software development phase.

This paper is laid out as follows: Section 2 justifies related work. Next, Section 3 proposes a set of reviewing criteria. Then, Section 4 analyses current change impact analysis
techniques capability and proposes a set of criteria of a new impact analysis technique for the software development phase. Finally, Section 5 concludes the study.

2. Related Work

To describe the impact analysis area, two main aspects are reviewed which are definition of impact analysis, impact analysis process and current impact analysis techniques.

2.1 Impact Analysis

One of the most referred definitions of impact analysis is “a process of identifying potential consequences of a change, or estimating what needs to be modified to accomplish a change [1]”. The motivation behind the impact analysis activity is to identify software artifacts (i.e., requirement, design, class and test artifacts) that are potentially to be affected by a change. The change can be in a form of addition, removal and modification of new or existing software artifacts. With information on potentially affected software artifacts, effective planning can be made on what action will be undertaken with respect to the change.

There are two main perspectives to impact analysis which are the dependency analysis and the traceability analysis. Typically, the dependency analysis is also known as a program analysis. The program analysis focuses on identifying relationships among class artifacts or source codes by exploring the internal structure of the codes [1]. This analysis aims to determine what elements in the source codes could be potentially affected by a change. There are many types of program analysis techniques that have been introduced, such as the control dependency and the data dependency [8]. The control dependency uses a program’s conditional structures for the analysis whereas the data dependency analyses the program’s variables.

Comparatively to the program analysis, the traceability analysis is the analysis of relationships between software artifacts across different software phases. Since this analysis involves various software artifacts across different software phases, some researchers use this analysis to support impact analysis activity for the software development phase [9-14]. The difference between this analysis and the program analysis is that this analysis focuses on the dependencies between software artifacts in different software phases instead of a single software artifact. There are two types of traceability analysis which are the Pre-traceability analysis and the Post-traceability analysis [15]. The pre-requirement traceability provides a mechanism to verify that all requirements have been described in a formal requirement specification document. On the other hand, the post-requirement traceability provides a mechanism to ensure all requirements in the formal requirement specification document have been implemented and how they have been implemented in the software.

Much of the work on impact analysis has been limited to source code analysis [3-6] using the dependency analysis approach. Relying on the source code analysis does not account for the overall impact to a software project. Software artifacts such as design and test artifacts should be kept up-to-date according to the change. This indirectly shows that these software artifacts are part of the impacted artifacts by the change. Thus, to identify thorough consequences of making a change in a software project, an effective combination between the traceability analysis and the dependency analysis is important.

2.2 Impact Analysis Process

There are three main steps in the impact analysis process [16] which are: analyzing change specification and software artifacts; tracing potential impacts; and implementing the requested changes. The first step identifies a set of impacted artifacts that is thought to be initially
affected by the change specification. The initial set of impacted artifacts is called the Starting Impact Set (SIS). After the SIS identification, the next step analyses the SIS to trace potential impacts by filtering the unnecessary or false artifacts. The filtered result from this analysis is the Candidate Impact Set (CIS) (or also called the Estimated Impact Set (EIS)). Finally, the last step identifies a set of impacted artifacts that is actually modified during the actual change implementation. This set is of impacted artifacts called the Actual Impact Set (AIS).

The structure of the impact analysis process is an iterative process. During actual change implementation, new impacted artifacts might be discovered along the process. The new impacted artifacts are called the Discovered Impact Set (DIS). The DIS is considered as the under-estimate impact set in the CIS. Since not all artifacts in the CIS are in AIS, the false artifacts in the CIS are called the False Positive Impact Set (FPIS). The FPIS is considered as the over-estimate impact set in the CIS. The combination of artifacts in CIS with artifacts in DIS and the elimination of FPIS artifacts in that combination should be in the AIS results, i.e.,

\[ AIS = CIS + DIS - FPIS. \]

There are several metrics that can be used to evaluate the accuracy of CIS produced by the impact analysis process [17-19]. The metrics aim to evaluate the closeness between CIS and AIS results. The closer the results between CIS and AIS are, the higher the accuracy of CIS will be. For example, in [19] evaluation metrics, they use three metrics to evaluate the effectiveness of CIS results produced by impact analysis process. The metrics are the Completeness, the Correctness and the Kappa Value [20]. The Correctness Metric represents the percentage of the actual predicting impacted classes from the overall actual impacted classes, the Completeness Metric represents the percentage of the actual impacted classes from the overall predicting impacted classes and the Kappa Value Metric is used to represent the level of agreement between the predicted set of potential impacted classes (CIS) and the actual set of impacted classes (AIS).

2.3 Current Impact Analysis Techniques

There are two categories of impact analysis techniques [21] which are the static analysis technique and the dynamic analysis technique. The static analysis technique develops a set of potential impacted classes by analyzing program static information that is generated from software artifacts (i.e., requirement, design, class and test artifacts). Conversely for the dynamic analysis technique, this technique develops a set of potential impacted classes by analyzing program dynamic information or executing code. The following sub-sections describe each impact analysis category.

2.3.1 Static Analysis: There are three most related current static analysis techniques to the new proposed approach which are the Use Case Maps (UCM) technique [3], the requirement interdependency technique [22], and the class interactions prediction with impact prediction filters (CIP-IPF) technique [10].

The UCM technique [3]: This technique uses the UCM model [23] to perform impact analysis on the functional requirements and the high level design model. This technique assumes that all the functional requirements and the high level design model are completely developed. This technique has two limitations which are: (1) there is no traceability technique used from the functional requirements and the high level design artifacts to the actual source codes. This technique only makes an assumption that the content of these two artifacts that is represented using the UCM model are reflected to the class artifacts. Any affected elements in the UCM model are indirectly reflected to the affected class artifacts; and (2) there is no dynamic analysis or source code analysis involved in this technique. Based on the precept that
some of the effect of a change from a class to other class(es) may only be visible through
dynamic or behavior analysis of the changed class, results from this technique tend to miss
some actual impacted classes.

The requirement interdependency technique [22]: This technique uses a combination of
requirement interdependency model and horizontal traceability analysis. For the requirement
interdependency model, the requirement interdependency detection technique from [24] is
used whereas for the horizontal traceability analysis, the Information Retrieval technique [25]
is employed. As with the UCM technique [3], this technique assumes that the requirements
and the class artifacts are completely developed prior to implementing the technique. One
main advantage of this technique compared to the UCM technique is that this technique has
included traceability link detection between the requirements artifacts and the class artifacts.
This detection is important as impact of changes can be navigating to the class artifacts
effectively. However, there are two limitations of this technique. (1) First, this technique
does not involve design artifacts. It is known that not all requirements can be directly mapped
to class artifacts [26]. Some requirements need design artifacts as a mediator to map to the
class artifacts. Thus, not all impacted classes can be detected based on the impacted
requirements as in some circumstances design decision is required to support the detection
[12-13]. (2) There is no dynamic analysis or source code analysis involved in this technique.
Based on the precept that some of the effect of a change from a class to other class(es) may
only be visible through dynamic or behavior analysis of the changed class [27-28], results
from this technique tend to miss some actual impacted classes.

The Class Interactions Prediction with Impact Prediction Filters (CIP-IPF) technique [10]:
This technique uses a class interactions prediction as a model for detecting impacted classes.
This technique is almost similar to the requirement interdependency technique [7] as changes
to a requirement will be mapped to the class artifacts. However, the differences between these
techniques are this technique: (1) develops its own requirement interactions detection
 technique; (2) uses the Rule-based technique [29] for the horizontal traceability analysis
detection and (3) assumes that some classes are partially developed and some of them are
fully developed. This technique has its strength compared to the UCM technique [3] and the
requirement interdependency technique [22]. First, comparing to the UCM technique, this
technique has traceability link detection between the requirements artifacts and the class
artifacts feature. This feature is used to navigate impact of changes at the requirement level to
the class artifacts. Next, comparing to the requirement interdependency technique, this
technique introduces a design artifacts analysis as part of its process to identify impacted class
artifacts.

2.3.2 Dynamic Analysis: There are two techniques which are the Influence Mechanism
technique [5] and the Path Impact technique [6]. Basically, these techniques predict the
impact set (classes or methods) based on method level analysis.

The Influence Mechanism technique: This technique introduces the Influence Graph (IG)
as a model to identify impacted classes. This technique uses the class artifacts as a source of
analysis and assumes that the class artifacts are completely developed. The advantage of this
technique can be considered when its uses a combination of the static analysis and the
dynamic analysis techniques during its impact analysis implementation. The static analysis
intends to give preliminary results of the impacted class artifacts prior to performing the
dynamic analysis. Only the preliminary affected classes will be further investigated to detect
more affected classes using the dynamic analysis. This ultimately improves the effectiveness
of the technique’s implementation. However, there is a limitation for this technique which is
there is no formal mapping or traceability process from requirements artifacts or design
artifacts to class artifacts. This process is important in impact analysis process as changes not only come from class artifacts but it also comes from design and/or requirements artifacts. Since design and requirements artifacts do interact among them vertically (between two different artifacts of a same type) and horizontally (between requirement and design artifacts), changes that happen to them could contribute to different affected class artifacts. In some circumstances, focusing on the source code analysis may not able to detect those affected classes.

The Path Impact technique: This technique uses the Whole Path DAG (directed Acyclic Graph) model [30] as a model to identify impacted classes. The concept of implementation for this technique is almost similar to the Influence Mechanism technique as this technique uses the class artifacts as a source of analysis and assumes that the class artifacts are completely developed. Also, this technique performs a preliminary analysis prior to performing a detail analysis. The advantage of this technique compared to the Influence Mechanism technique is that this technique performs impact analysis at the method level instead of class level analysis using the SEQUITUR algorithm [31]. The results of the analysis will ultimately give higher accuracy than the results produced at the class level analysis. However, there are two limitations of this technique. First, the implementation is time consuming as the technique opens to a huge number of data when the analysis goes to a large application. Next, there is no formal mapping process from requirements artifacts or design artifacts to class artifacts. As described earlier, this process is important in impact analysis process as changes not only come from class artifacts but also from design and/or requirements artifacts.

3. Reviewing Criteria

To review the capability of current impact analysis techniques to support impact analysis for the software development phase, we developed four reviewing elements [3,4]. These elements are considered as important elements to support impact analysis for the software development phase. The elements are: (1) the source of data to develop the impact analysis model; (2) impact analysis model development technique; (3) partially developed class consideration; and (4) impact analysis implementation.

Source of data element: This element is used to develop the impact analysis model. Some techniques use the class artifact or source code and some of them use the requirement artifact and the design artifact as the source of developing the model. In the impact analysis technique for the software development phase, the requirement artifact and the design artifact are considered to be the practical source of development since they are the most stable and completed forms of user requirements compared to class artifact or source code [3,4].

Impact analysis model development technique element: This element defines a technique that the impact analysis technique used to develop the impact analysis model. There are two techniques that can be used to develop the model which are the reverse engineering or the forward engineering (or predictive technique). Since the class artifact or source code is not practical as the source of model development for the software development phase, the reverse engineering technique is indirectly considered to be impractical as the model development technique [6, 9].

Partially developed class consideration element: This element defines capability of the impact analysis technique to include partially developed class analysis in its implementation. This consideration is important for the impact analysis technique as not all classes in the software development phase are fully developed [32].

Analysis technique element: This element defines how a technique analyses the impact analysis model to identify impacted artifacts. The analysis can be: (1) static analysis
technique; or (2) dynamic analysis technique. Some techniques use either the static analysis technique or the dynamic analysis technique only and some of them combine between the static analysis technique and the dynamic analysis technique. Since the static analysis technique has an advantage when partially developed class involved in the analysis [13] and the dynamic analysis technique has an advantage when all classes are fully developed compared to the static analysis technique, the combination of both techniques is considered as the best analysis technique for impact analysis in the software development phase. However, the dynamic analysis technique will only be implemented if some potential impacted classes produced by the static analysis technique are fully developed. In other words, the dynamic analysis technique will be implemented if possible.

4. Analysis Results

Table below summarizes current impact analysis techniques based on the fourth reviewing elements.

**Table 1. Current Impact Analysis Techniques’ Strengths and Weaknesses**

<table>
<thead>
<tr>
<th>Technique Name</th>
<th>Reviewing Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Maps</td>
<td>Source of analysis: Requirement artifact and design artifact</td>
</tr>
<tr>
<td></td>
<td>Impact analysis model development technique: Predictive technique</td>
</tr>
<tr>
<td></td>
<td>Partially developed class consideration: Not included</td>
</tr>
<tr>
<td></td>
<td>Analysis Technique: Static analysis only</td>
</tr>
<tr>
<td>Requirement Interdependency</td>
<td>Impact analysis model development technique: Predictive technique</td>
</tr>
<tr>
<td></td>
<td>Source of analysis: Requirement artifact only</td>
</tr>
<tr>
<td></td>
<td>Analysis Technique: Static analysis only</td>
</tr>
<tr>
<td>Class Interactions Prediction</td>
<td>Source of analysis: Requirement artifact and design artifact</td>
</tr>
<tr>
<td></td>
<td>Impact analysis model development technique: Predictive technique</td>
</tr>
<tr>
<td></td>
<td>Partially developed class consideration: Not included</td>
</tr>
<tr>
<td></td>
<td>Analysis Technique: Static analysis only</td>
</tr>
<tr>
<td>CoverageImpact</td>
<td>Source of analysis: Class artifact or source code</td>
</tr>
<tr>
<td></td>
<td>Impact analysis model development technique: Reverse engineering technique</td>
</tr>
<tr>
<td></td>
<td>Partially developed class consideration: Not included</td>
</tr>
<tr>
<td></td>
<td>Analysis Technique: Dynamic analysis only</td>
</tr>
</tbody>
</table>
Table 1: Comparison of current impact analysis techniques

<table>
<thead>
<tr>
<th>Technique</th>
<th>Source of analysis: Class artifact or source code</th>
<th>Impact analysis model development technique: Reverse engineering technique</th>
<th>Partially developed class consideration: Not included</th>
<th>Analysis Technique: Dynamic analysis only</th>
</tr>
</thead>
<tbody>
<tr>
<td>PathImpact</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Influence Graph</td>
<td>Analysis Technique: Static and dynamic analysis</td>
<td>Source of analysis: Class artifact or source code</td>
<td>Impact analysis model development technique: Reverse engineering technique</td>
<td>Partially developed class consideration: Not included</td>
</tr>
</tbody>
</table>

Looking at the above table, we could say that none of the current impact analysis techniques support all the important elements for impact analysis implementation in the software development phase. For example, the Use Case Maps technique only supports the source of analysis element through the use of requirement and design artifacts and the impact analysis model development technique element through the use of the predictive technique. However, this technique does not include the partially developed class consideration and only uses the static analysis technique for its implementation. Since, these two elements do not comply with the reviewing elements, this technique will not be effectively used for the software development phase. Therefore, we claim that a new impact analysis technique needs to be constructed.

The new impact analysis technique will exhibit four characteristics: (1) uses high level artifacts (i.e., requirement and design artifacts) as the source of developing the impact analysis model; (2) applies the predictive technique as the impact analysis model development technique; (3) includes partially developed class analysis in the dynamic analysis technique implementation; and (4) combines the static analysis technique and the dynamic analysis technique to implement impact analysis.

5 Conclusion

This paper presents a comprehensive analysis of current impact analysis technique from the perspective of supporting software development phase implementation. From the analysis, we propose that the new impact analysis technique needs to have four main characteristics. The characteristics are: (1) the new technique uses high level artifacts (i.e., requirement and design artifacts) as the source of developing the impact analysis model; (2) applies the predictive technique as a technique for developing the impact analysis model; (3) introduce a partially developed class analysis as part of its dynamic analysis technique; and (4) combines the static analysis technique and the dynamic analysis technique for impact analysis implementation. As for the future work, we intend to develop a new impact analysis technique that is based on the fourth characteristics.

Acknowledgements

We would like to thank to Universiti Teknologi Malaysia (UTM) for financial support under the Potential Academic Staff (PAS) grant Vot No 0K001. Also to final year postgraduate students of software engineering course at the Advanced Informatics School, Universiti Teknologi Malaysia who have been involved in the study.
References


Author

Nazri Kama received his Master Degree in Real-time Software Engineering and Bachelor Degree in Management Information System from Universiti Teknologi Malaysia in 2002 and 2000 respectively. He then obtained his PhD degree at The University of Western Australia in 2011. His research interests are in software development, software maintenance, impact analysis, traceability, and requirement interactions.