Change Impact Analysis

Martin Ward

Reader in Software Engineering

martin@gkc.org.uk

Software Technology Research Lab
De Montfort University

Change Impact Analysis

Impact analysis is a process that predicts and determines the parts of a software system that can be affected by changes to the system.

- **Change Set:** The parts of the software system that are to be changed
- **Impact Set:** The parts of the software system that are affected by the changes

Impact Analysis is Essential

Impact analysis is a systematic approach to understanding impacts of software changes and is essential to:

- Identify the parts that require retesting
- Improve estimation of time, labor and money required for maintenance
- Reduce potential errors due to unknown change impacts
- Improve overall efficiency in software maintenance

Change Impact Analysis

Impact analysis involves the following steps:

Change Impact Analysis

Impact analysis involves the following steps:

1. Identify the artefacts that are to be modified

Change Impact Analysis

Impact analysis involves the following steps:

1. Identify the artefacts that are to be modified
2. Trace forwards through any relationships that indicate a dependency on that artefact. These relationships will have the selected artefact as a source of the dependency.

Change Impact Analysis

Impact analysis involves the following steps:

1. Identify the artefacts that are to be modified
2. Trace forwards through any relationships that indicate a dependency on that artefact. These relationships will have the selected artefact as a source of the dependency.
3. If a change affects any of these “dependent artefacts”, then other artefacts that depend on them will also be affected. The impact analysis must therefore act recursively looking for relationships that have any of the “dependent artefacts” as sources.

Change Impact Analysis

Impact analysis involves the following steps:

1. Identify the artefacts that are to be modified
2. Trace forwards through any relationships that indicate a dependency on that artefact. These relationships will have the selected artefact as a source of the dependency.
3. If a change affects any of these "dependent artefacts", then other artefacts that depend on them will also be affected. The impact analysis must therefore act recursively looking for relationships that have any of the "dependent artefacts" as sources.
4. This process continues until a complete graph is obtained starting at the selected artefacts and finishing with artefacts on which nothing else depends. Each artefact in the graph could therefore be affected by the change.

Impact Analysis Scope Management

Any recursive analysis like this can produce large numbers of results which can often be meaningless to the user due to the sheer volume of information. In order to make the results more meaningful, the impact analysis needs to be scoped in some way. Mechanisms for scoping the impact analysis queries include:

Impact Analysis Scope Management

Any recursive analysis like this can produce large numbers of results which can often be meaningless to the user due to the sheer volume of information. In order to make the results more meaningful, the impact analysis needs to be scoped in some way. Mechanisms for scoping the impact analysis queries include:

- Identify the maximum number of iterations of relationship traversal in the graph. This limits the graph to the artefacts that are most closely related to the artefact being changed.

Impact Analysis Scope Management

Any recursive analysis like this can produce large numbers of results which can often be meaningless to the user due to the sheer volume of information. In order to make the results more meaningful, the impact analysis needs to be scoped in some way. Mechanisms for scoping the impact analysis queries include:

- Identify the maximum number of iterations of relationship traversal in the graph. This limits the graph to the artefacts that are most closely related to the artefact being changed.
- Restrict the query to specific relationships. This identifies the sort of relationships that are of interest in the analysis.

Static Impact Analysis

- Depends on analysis of source code
- Based on assumptions of all possible software runtime behaviors
- Results can include most of the software system in the impact set

Dynamic Impact Analysis

- Based on software runtime data and dynamic interactive behaviors of the software system
- Depends on a set of executions of the system
- Tend to produce more precise results than static approaches

Computing an Impact Analysis

An impact analysis within a single program is equivalent to a **forward slice**.

Computed by tracking control and data dependencies:

```
$i := 1$;
prod $:= 1$;
sum $:= 0$;
**while** $i \leqslant n$ **do**
    $a :=$ input$[i]$;
    sum $:=$ sum $+ a$;
    prod $:=$ prod $* a$;
    $i := i + 1$ **od**;
output1 $:=$ sum;
output2 $:=$ prod
```

Computing an Impact Analysis

An impact analysis within a single program is equivalent to a **forward slice**.

Computed by tracking control and data dependencies:

```
$i := 1$;
prod $:= 1$;
$\boxed{\text{sum} := 0;}$
**while** $i \leqslant n$ **do**
    $a :=$ input$[i]$;
    $\boxed{\text{sum} := \text{sum} + a;}$
    prod $:=$ prod $* a$;
    $i := i + 1$ **od**;
$\boxed{\text{output1} := \text{sum};}$
output2 $:=$ prod
```

Note: a forward slice is not usually a complete program.

Forward Slicing Example

If the condition in the loop is affected, then all variables assigned in the loop are included:

```
$i := 1$;
prod $:= 1$;
sum $:= 0$;
**while** $i \leqslant n \wedge$ sum $\leqslant$ max **do**
    $a :=$ input$[i]$;
    sum $:=$ sum $+ a$;
    prod $:=$ prod $* a$;
    $i := i + 1$ **od**;
output1 $:=$ sum;
output2 $:=$ prod
```

Forward Slicing Example

If the condition in the loop is affected, then all variables assigned in the loop are included:

```
$i := 1;$
prod $:= 1;$
$\boxed{\text{sum} := 0;}$
while $\boxed{i \leqslant n \wedge \text{sum} \leqslant \text{max}}$ do
    $\boxed{a := \text{input}[i];}$
    $\boxed{\text{sum} := \text{sum} + a;}$
    $\boxed{\text{prod} := \text{prod} * a;}$
    $\boxed{i := i + 1}$ od;
$\boxed{\text{output1} := \text{sum};}$
$\boxed{\text{output2} := \text{prod}}$
```

Forward Slicing Example

Slicing on the first assignment to sum:

```
$i := 1$;
prod $:= 1$;
sum $:= 0$;
while $i \leqslant n$ do
    $a :=$ input$[i]$;
    sum $:=$ sum $+ a$;
    prod $:=$ prod $* a$;
    $i := i + 1$ od;
output1 $:=$ sum;
sum $:= 0$;
output2 $:=$ sum
```

Forward Slicing Example

Slicing on the first assignment to sum:

```
$i := 1;$
prod $:= 1;$
$\boxed{\text{sum} := 0;}$
**while** $i \leqslant n$ **do**
    $a :=$ input$[i];$
    $\boxed{\text{sum} := \text{sum} + a;}$
    prod $:=$ prod $* a;$
    $i := i + 1$ **od**;
$\boxed{\text{output1} := \text{sum};}$
sum $:= 0;$
output2 $:=$ sum
```

Change Impact Analysis Tools

Chianti is a change impact analysis tool for Java that is implemented within eclipse.

- Analyse two versions of a Java program
- Decompose their difference into a set of atomic changes
- Calculate a partial order of inter-dependencies of these changes
- Report change impact in terms of affected (regression or unit) tests whose execution behavior may have been modified by the applied changes.
- For each affected test, determine a set of affecting changes that were responsible for the test's modified behavior.