**Terms:**

* **Issue** -   
  A unit of work to accomplish an improvement in a data system
* **Bug-issue** -   
  An issue that deals with fixing a bug. In this article we will deal only with bug-issues therefore we will use only the term 'issue'.
* **Version control system (VCS)** -  
  A software component for management of software configuration.
* **Commit** -  
  A version of a project that have been documented by a VCS.
* **Testcase** -   
  Executable software procedure that is defined by <*in*, *expected\_out*>.  
  Possible states:
  + **Passed** -   
    The execution of the testcase over *in* as an input yields *expected\_out* as an output.
  + **Failed** -   
    The execution of the testcase over *in* as an input not yields *expected\_out* as an output.
  + **Generated runtime error** -   
    The execution of the testcase over *in* as an input is generating runtime error
* **Valid testcase** –   
  Given Issue *i*, Commit *f*, Commit *p*, when *f* is fix-commit of *i*, and *p* is a parent-commit of *f*, we would way that Testcase *t* is a **valid testcase**  of *i* iff *t* pass in *f* and pass in *p*
* Valid issue -   
  We would say that Issue *i* is **valid** iff *i* has a valid testcase
* **Possible issue** – Issue that:
  + Is from type bug
  + Has diffs in source files
  + Has an associated fix-commit
* **Issue running** – Given Issue *i* Commit *f* and MavenModule *m* when , *f* is a fix-commit of *i* and *m* contains diffed source files of *f*, **issue running**is the process of finding valid testcases of *i* that are testing *m* when the project is checked out at *f*. **One issue can have few runnings.**
* **Complete issue running -**  An issue running that completes without any error.
* **Complete issue –** issue that has a complete issue running

**Experimental setup**

The abstract process of *Bugminer* is as follows:

Given Project *p* and Issue *i*:

1. Collect the *fix-commits* of *i*
2. For each <*fc*,*fcp*> when *fc*  is a *fix-commit* and *fcp* is *fc*'s parent:
   1. Generate testcases in *fc.* Denote these testcases as *gen\_testcases*
   2. Run *gen\_testcases* in *fc*
   3. Run *gen\_testcases* in *fcp*
   4. For each testcase *t* in *gen\_testcases* the passes in *fc* and fails *fcp*:
3. return

**Data**:

We looked at the following software projects: tika, commons-math, commons-lang, commons-compress.

In order to manage the project versions and to run the test we used the VCS git.

In order to run the manage the build-phases (compilation, testing, cleaning) of the projects we used Maven.

In order to get the issues of the project we used the issue tracker jira. For example, to get the relevant issues of the project tika we raised the query '' project = TIKA AND issuetype = Bug AND createdDate >= '2000/11/26' ORDER BY createdDate ASC " to the issue tracker.

In order to get the fix-commits for a given issue we used the commit messages. Our criteria to define commit c as a fix-commit of issue I is that the message of c contains the issue- key of i.   
**Metrics**:

* Hit rate:  
  (Valid issues)/(All issues)
* Threshold rate  
  (Valid )/(All generated testcases)