Comparison of Automated Unit Testing Tools

Research Project Presentation
CS9550A: Specifications, Testing, and Quality Assurance
Instructor: Prof. Kostas Kontogiannis

Aasminpreet Singh Kainth
Department of Computer Science
Middlesex College, Western University

Agenda

1. **What is Unit Testing?**
   A. **Why Bother Unit Testing?**
2. **Problem Description?**
3. **Motivation**
4. **Importance**
5. **Analysis Phases**
6. **Conclusion**
7. **LIMITATIONS AND FUTURE WORK**

Unit Testing

a level of software testing where individual units/components of a software are tested.

The purpose is to validate that each *unit* of the software performs as designed.

A *unit* is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

In procedural programming, a *unit* may be an individual program, function, procedure, etc.

In object-oriented programming, the smallest *unit* is a method, which may belong to a base/ super class, abstract class or derived/ child class.

Why Bother Unit Testing?

There are many reasons why unit testing should be adopted for any software engineering project. These reasons include:

1. **Faster Debugging** – Without unit testing, the time it takes to debug or resolve a functional test that may have failed takes a long time to track down.
2. **Better Design and Documentation** – Writing unit tests forces developers into thinking how their code is going to be used and has generally resulted in better design and even better documentation.

3. **Better Feedback Mechanism** – When all unit tests for a system are run as a whole, the state of the system as a whole can be measured.

4. **Reduce Future Cost** – Many studies have proved that it costs significantly more money to fix a bug found later in the release than earlier in the development. A good unit test suite will uncover basic bugs early on in the development cycle

Problem Description

Some developers still do not do much unit testing. The possible reasons are:

1. Some developers do not think the time and effort will be worthwhile.
2. Unit tests must be maintained, and maintenance for unit tests is often not budgeted.
3. A third possibility is that developers may not know how to design and implement high quality unit tests.

Importance

Using automated unit test tools instead of manual testing can help with all three problems.

1. Automated unit test tools can reduce the time and effort needed to design and implement unit tests

2. These tools can make it easier to maintain tests as the program changes

3. Tools can encapsulate knowledge of how to design and implement high quality tests so that developers do not need to know as much.

But an important question that developers must answer is **“which tool should I use?”**

Motivation

Automation support for unit testing, in the form of various automation tools, could significantly lower the cost of performing unit testing phase as well as decrease the time developer involved in the actual testing.

The problem is how to choose the most appropriate tool that will suit developer requirements consisting:

1. cost involved,
2. effort needed,
3. level of automation provided,
4. language support, etc.

Analysis: Phase 1

The total number of tools found is 22, which includes all the domain, type of availability, etc. Out of 22 tools, I found 4 prototype tools.

1. AutoGen
2. Godzilla
3. SimC
4. Symclat

Prototype tools are developed by some educational group or researchers.

Most of them are not available for public use; because these tools are not completely developed to launch in the market.

Analysis: Phase 2

CATEGORIZATION OF TOOLS

Once I collected all the automation supported unit-testing tools. It is time to categorize them and put them into groups.

A. Based on Domain
B. Programming Language Support
C. Testing Technique
D. Availability of Tools

Tool Name	Language supported	Testing Technique used	Domain belongs	Availability
Agitar	Java	Observation driven testing	Desktop	Commercial
Austin	C	Search based technique	Desktop	Open source
CUTE	C/C++	Concolic testing	Desktop	Free
Crest	C	Systematic dynamic test generation	Desktop	Open source
DART	C	Concolic testing	Desktop	Open Source
Eclat	Java	Classification technique	Desktop	Academic
EXE	C	Concolic testing	Desktop	Open Source
Findsbugs	Java	Static analysis	Desktop	Open source
Jtest	Java	White box testing	Desktop and Server edition	Commercial from Parasoft
JCrasher	Java	Robustness testing or random testing	Desktop	Open source
JUB (Junit test case builder)	Java	Builder pattern	Desktop	Open source
JWalk	Java	Specification based testing	Desktop	Academic
JCute	Java	Search based and Concolic testing	Desktop	Open source
PathCrawler	C	Concolic testing	Desktop	Online Server
PEX	C#	Dynamically Symbolic exec	Desktop	License from Microsoft
Randoop	Java	Feedback- directed random testing	Desktop	Open source
SMART	C	Concolic testing		Open Source
TestGen4j	Java	Boundary level testing	Desktop	Not Available

CATEGORIZATION OF TOOLS

Tool Name	Language supported	Testing Technique used	Domain belongs	Availability
Agitar	Java	Observation driven testing	Desktop	Commercial
Austin	C	Search based technique	Desktop	Open source
CUTE	C/C++	Concolic testing	Desktop	Free
Crest	C	Systematic dynamic test generation	Desktop	Open source
DART	C	Concolic testing	Desktop	Open Source
Eclat	Java	Classification technique	Desktop	Academic
EXE	C	Concolic testing	Desktop	Open Source
Findsbugs	Java	Static analysis	Desktop	Open source
Jtest	Java	White box testing	Desktop and Server edition	Commercial from Parasoft
JCrasher	Java	Robustness testing or random testing	Desktop	Open source
JUB (Junit test case builder)	Java	Builder pattern	Desktop	Open source
JWalk	Java	Specification based testing	Desktop	Academic
JCute	Java	Search based and Concolic testing	Desktop	Open source
PathCrawler	C	Concolic testing	Desktop	Online Server
PEX	C#	Dynamically Symbolic exec	Desktop	License from Microsoft
Randoop	Java	Feedback- directed random testing	Desktop	Open source
SMART	C	Concolic testing		Open Source
TestGen4j	Java	Boundary level testing	Desktop	Not Available

CATEGORIZATION OF TOOLS

Based on Domain

Domain defines as the type of application that is possible to test with the tool.

The most common domains identified are Desktop and server based.

From table, most of the tools support only desktop-based applications.

Tool Name	Language supported	Testing Technique used	Domain belongs	Availability
Agitar	Java	Observation driven testing	Desktop	Commercial
Austin	C	Search based technique	Desktop	Open source
CUTE	C/C++	Concolic testing	Desktop	Free
Crest	C	Systematic dynamic test generation	Desktop	Open source
DART	C	Concolic testing	Desktop	Open Source
Eclat	Java	Classification technique	Desktop	Academic
EXE	C	Concolic testing	Desktop	Open Source
Findsbugs	Java	Static analysis	Desktop	Open source
Jtest	Java	White box testing	Desktop and Server edition	Commercial from Parasoft
JCrasher	Java	Robustness testing or random testing	Desktop	Open source
JUB (Junit test case builder)	Java	Builder pattern	Desktop	Open source
JWalk	Java	Specification based testing	Desktop	Academic
JCute	Java	Search based and Concolic testing	Desktop	Open source
PathCrawler	C	Concolic testing	Desktop	Online Server
PEX	C#	Dynamically Symbolic exec	Desktop	License from Microsoft
Randoop	Java	Feedback- directed random testing	Desktop	Open source
SMART	C	Concolic testing		Open Source
TestGen4j	Java	Boundary level testing	Desktop	Not Available

CATEGORIZATION OF TOOLS

Based on Programming Language Support

Each tool has the support for at least one specific programming language.

The most common programming languages concluded form the table are Java and C.

Only one tool PEX has the support for Microsoft C#. All other's either support Java or C.

CUTE is only tool with the support of both C/C++.

Tool Name	Language supported	Testing Technique used	Domain belongs	Availability
Agitar	Java	Observation driven testing	Desktop	Commercial
Austin	C	Search based technique	Desktop	Open source
CUTE	C/C++	Concolic testing	Desktop	Free
Crest	C	Systematic dynamic test generation	Desktop	Open source
DART	C	Concolic testing	Desktop	Open Source
Eclat	Java	Classification technique	Desktop	Academic
EXE	C	Concolic testing	Desktop	Open Source
Findsbugs	Java	Static analysis	Desktop	Open source
Jtest	Java	White box testing	Desktop and Server edition	Commercial from Parasoft
JCrasher	Java	Robustness testing or random testing	Desktop	Open source
JUB (Junit test case builder)	Java	Builder pattern	Desktop	Open source
JWalk	Java	Specification based testing	Desktop	Academic
JCute	Java	Search based and Concolic testing	Desktop	Open source
PathCrawler	C	Concolic testing	Desktop	Online Server
PEX	C#	Dynamically Symbolic exec	Desktop	License from Microsoft
Randoop	Java	Feedback- directed random testing	Desktop	Open source
SMART	C	Concolic testing		Open Source
TestGen4j	Java	Boundary level testing	Desktop	Not Available

CATEGORIZATION OF TOOLS

Based on Testing Technique

It is interesting to see in table, that tools are based on many different type of testing technique, there is not any one testing technique that has dominated the list of tools.

Some testing techniques were found to be used by few tools such as “concolic testing” and symbolic execution.

Tool Name	Language supported	Testing Technique used	Domain belongs	Availability
Agitar	Java	Observation driven testing	Desktop	Commercial
Austin	C	Search based technique	Desktop	Open source
CUTE	C/C++	Concolic testing	Desktop	Free
Crest	C	Systematic dynamic test generation	Desktop	Open source
DART	C	Concolic testing	Desktop	Open Source
Eclat	Java	Classification technique	Desktop	Academic
EXE	C	Concolic testing	Desktop	Open Source
Findsbugs	Java	Static analysis	Desktop	Open source
Jtest	Java	White box testing	Desktop and Server edition	Commercial from Parasoft
JCrasher	Java	Robustness testing or random testing	Desktop	Open source
JUB (Junit test case builder)	Java	Builder pattern	Desktop	Open source
JWalk	Java	Specification based testing	Desktop	Academic
JCute	Java	Search based and Concolic testing	Desktop	Open source
PathCrawler	C	Concolic testing	Desktop	Online Server
PEX	C#	Dynamically Symbolic exec	Desktop	License from Microsoft
Randoop	Java	Feedback- directed random testing	Desktop	Open source
SMART	C	Concolic testing		Open Source
TestGen4j	Java	Boundary level testing	Desktop	Not Available

CATEGORIZATION OF TOOLS

Based on Availability of Tools

Availability of tool makes big difference for someone who wants to use.

In the table, it can be seen that there are 2-3 commercial tools available such as Jtest, Agitar, etc.

Most of the tools are open source but unfortunately, some of them are not even available.

There are few examples of academic and licensed tools also like Eclat, PEX.

Analysis: Phase 3

TYPE OF ERRORS

This phase dealt with finding the type of error(s) that each tool can find.

The aim was to find the type of error(s) for each tool by reading the research papers instead of by trying them on some source code.

The reason I could not find the type of errors for some tools was the lack of research papers read.

Tool Name	Type of Error(s)
CUTE	Error in mathematical calculation even inside nested statements; Unnecessary calls of same method; Results comparison with expected values; Null point exception and out of bound in array; Unused code Specialized rules (J2EE); Formatting in the code
DART	Program crashes; Assertion violation; Non termination; Memory allocation detect with collaboration with other run time checking tool
EXE	Constraint Solving; Independent constraint optimization; Bit-victor arithmetic
ECLAT	Out of bound Exception; Pre/post condition violation; Illegal inputs
Jcrasher	Robustness failure; Undeclared run time exception; Pre/Post condition violation Out of memory error; Analysis of exception for error
Jcute	Deadlocks; Uncaught Exceptions; Infinite loop
Jwalk	Unexpected interaction among methods such as dead ends on the fly, broken pre condition; Interleaved constructor
Path Crawler	Infinite loop detection; Pre/Post condition violation; Handling floating point numbers and arithmetic operations; Aliasing and pointer arithmetic
PEX	Argument exceptions; Out of bound in array; Arithmetic specification; Missing pre/Post conditions; Buffer overflow or Resource leak; Syntactic programming error
Randoop	Run time assertion violation; Runtime access violation; Missing messages in resource file and state of resource file; Memory management errors; Concurrency error such that related to test input

TYPE OF ERRORS

Tool Name	Type of Error(s)
CUTE	Error in mathematical calculation even inside nested statements; Unnecessary calls of same method; Results comparison with expected values; Null point exception and out of bound in array; Unused code Specialized rules (J2EE); Formatting in the code
DART	Program crashes; Assertion violation; Non termination; Memory allocation detect with collaboration with other run time checking tool
EXE	Constraint Solving; Independent constraint optimization; Bit-victor arithmetic
ECLAT	Out of bound Exception; Pre/post condition violation; Illegal inputs
Jcrasher	Robustness failure; Undeclared run time exception; Pre/Post condition violation Out of memory error; Analysis of exception for error
Jcute	Deadlocks; Uncaught Exceptions; Infinite loop
Jwalk	Unexpected interaction among methods such as dead ends on the fly, broken pre condition; Interleaved constructor
Path Crawler	Infinite loop detection; Pre/Post condition violation; Handling floating point numbers and arithmetic operations; Aliasing and pointer arithmetic
PEX	Argument exceptions; Out of bound in array; Arithmetic specification; Missing pre/Post conditions; Buffer overflow or Resource leak; Syntactic programming error
Randoop	Run time assertion violation; Runtime access violation; Missing messages in resource file and state of resource file; Memory management errors; Concurrency error such that related to test input

TYPE OF ERRORS

Most common errors that can be find by maximum tools are “out of bound” and “exception related errors”.

The pre/post violation is another common error that most tools can find.

PEX is the tool that can find maximum number of errors i.e. eight.

Conclusion

Major contribution of the research work:

1. The list of tools and categorization table could be very useful for someone who wants to choose automated unit testing tool.

2. The categorization of all found tools explain the type of error(s) that can be found by some tools.

3. The table for categorization would be very useful in selecting the tool for industrial use.

LIMITATIONS AND FUTURE WORK

The following are the limitations and future work in the research:

1. The table in the paper contains some commercial tools, because of accessibility problem to these tools I could not provide the detailed information about them and especially type of error(s) that they can find.

2. The aim to provide the detailed comparison of some selected tools by applying them on open source could not achieved because of time constraint.

3. The idea to develop new tool by merging two three tools of similar kind that can find more range of errors.

Questions?

Thank You.