PR:QA White Paper Series: WP1.1

# Static and Dynamic Testing Compared

**Abstract**

Many developers believe that dynamic testing is more effective than static testing. In fact the opposite is true. This paper compares the technical and economic effectiveness of static and dynamic testing. It shows that:

- static testing finds bugs before you compile.
- many bugs found by dynamic testing can be found earlier by static testing.
- the earlier you detect bugs the cheaper they are to fix.
- static testing is more cost effective than dynamic testing.
- quality-conscious developers use both static and dynamic testing tools.

PR:QA's static checking tools ***QA C***, ***QA C++*** and ***QA Fortran*** are the most comprehensive in the industry. Developers who use them achieve shorter delivery times and reduced development cost.

PR:QA White papers provide short discussions of a range of topics in software engineering. For a current list contact

| | |
|---|---|
| Programming Research Ltd. | Tel: 01932 888080 |
| Glenbrook House | Fax 01932 888081 |
| 1-11 Molesey Road | E-mail: support@prqa.co.uk |
| HERSHAM | WWW: http://www.prqa.co.uk |
| Surrey KT12 4RH | |

## Static or dynamic testing?

Software developers often think they must choose between static and dynamic testing. Some feel they cannot afford the tools to do both. Many prefer dynamic testing because they believe, mistakenly, that it finds real rather than potential bugs.

This note compares static and dynamic testing. It is based on the experiences of clients of Programming Research Ltd (PR:QA) who use both dynamic test tools and one or more of PR:QA's world-leading ***QA C***, ***QA C++*** or ***QA Fortran*** tools. All have reported that static testing is many times more cost-effective than dynamic testing.

***PR:QA users confirm that static testing beats dynamic testing by a wide margin.***

## Why is static testing more effective?

Static testing gives you comprehensive diagnostics for your code. ***QA C***, for example, warns you about:

- syntax errors
- code that will be hard to maintain
- code that will be hard to test
- code that does not conform to your coding standards
- non-portable usage
- ANSI violations

All this takes place before compilation. It takes roughly as long as compilation and checks every statement you have written.

***Static testing achieves 100% statement coverage in a relatively short time.***

Dynamic testing can take place only after compilation and linking. It may involve running several test cases each of which may take longer than compilation. **It finds bugs only in parts of the code that are actually executed.** Furthermore such testing often touches less than half the code. (See for example Woodward, Hedley et al. [1]).

***Typically dynamic testing takes longer than static testing yet finds fewer bugs.***

Of course static and dynamic testing find different classes of bugs. Some bugs are detectable only by static testing, some only by dynamic. Nevertheless about half of the bugs detectable by dynamic testing can be detected earlier by static testing.

Since static testing is faster and achieves 100% coverage, the unit cost of detecting these bugs by static testing is many times lower than by dynamic testing.

***If you are using neither static nor dynamic test tools, static tools offer greater marginal benefits.***

The essential difference between static and dynamic testing is this: static testing is about **prevention**, dynamic testing is about **cure**. Ideally you should use both static and dynamic testing tools. If you can only afford one kind of tool, remember:

***Prevention is always cheaper than cure.***

## Where is the evidence?

At PR:QA we know from experience, both our clients' and our own, that static testing outperforms dynamic testing. For example:

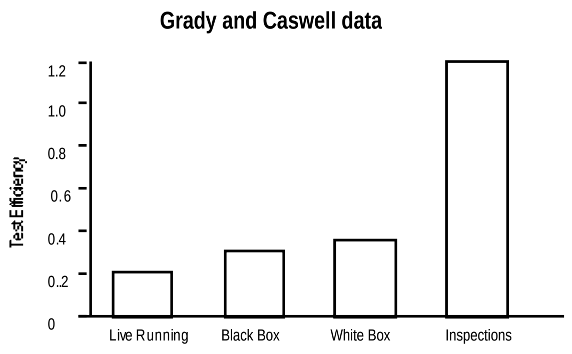
- Engineers at a large telecommunications company checked code with ***QA C*** and found a bug that their regression testing failed to detect. This bug would have cost over £6m to correct after release.

- One company took 18 months to achieve 100% dynamic statement coverage of 3% of their safety-critical code. In contrast, using ***QA Fortran*** for just one afternoon revealed over a hundred errors distributed throughout the code.

- At PR:QA it took us two days using an excellent, widely used dynamic testing tool to detect four errors that our static testing had missed. Our own static testing tools had previously found over 200 errors in under an hour.

These experiences are not isolated. Humphrey [2] cites comparable experiences at TRW. Grady and Caswell [3] give evidence suggesting that static testing was four times more effective than dynamic testing (see Figure 1).

***Figure 1***



Dutch associates of PR:QA relayed to us the following data reported by the *Dutch organisation for Structured design of Information Systems*. This data came from research done to compare the effectiveness of static testing against dynamic testing using a large number of test cases.

***Comprehensive Testing:***

| Type | Hours | Error Coverage |
|---|---|---|
| Static | 1 | 90% |
| Dynamic | 8 | 99% |

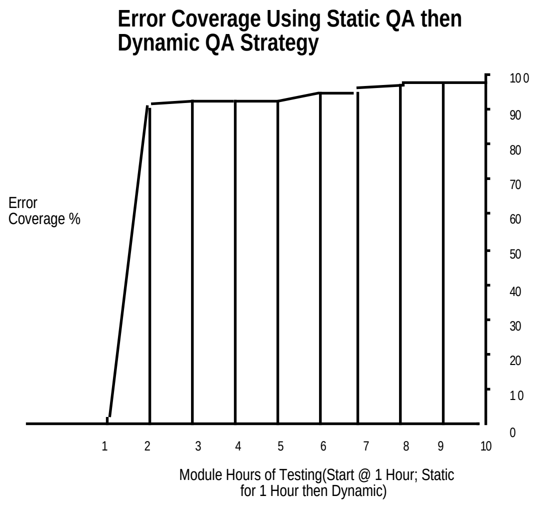
***Static Testing (of key or complex modules):***

| Type | Hours | Error Coverage |
|---|---|---|
| Static | 1 | 90% |
| Dynamic | 1 | 98% |

The system studied had functions with on average 50 lines of code each and taking on average 8 hours each to write. Coding adhered to a local coding standard.

Figure 2 summarises the results of using static testing followed by dynamic testing in the ratio 1:8, that is 1 hour of static testing followed by 8 hours of dynamic testing.

***Figure 2***



The evidence supports the following conclusions about comparative **time effectiveness**:

***Static testing is up to 100 times more effective. Even in selective testing, static testing may be up to 10 times more effective. The most pessimistic estimates suggest a factor of 4.***

***Dynamic testing detects fewer errors than static testing, but it does detect some that static testing misses.***

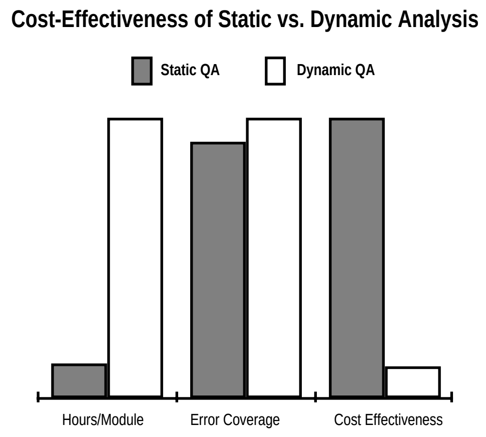
***If timescales are tight, use of dynamic testing tools might be omitted, but tool-supported static testing should never be omitted.***

***To eliminate as many errors as possible, both static and dynamic testing should be used.***

Figure 3 shows these results in terms of **cost-effectiveness**:

Vendors of the best dynamic testing tools argue convincingly that use of their tools result in a 30% increase in test efficiency. Such benefits repay user investment in a few months.

***Figure 3***



***With static testing tools the payback is measured in weeks, and sometimes in days.***

## Summary

- Static testing is a bargain compared with dynamic testing.
- Static and dynamic testing are complementary.
- To maximise software reliability, you should use both static and dynamic techniques supported by appropriate tools (Hatton [4]).

## References

[1] Woodward, M. R, Hedley, D. et al *Experience with path analysis and testing of programs*, IEEE Trans. Software Engineering **6**(3): 278-286, 1980

[2] Humphrey, W. S, *Managing the Software Process*, Addison-Wesley 1990

[3] Grady, R. B, and Caswell, D. L, *Software Metrics: Establishing a Company-Wide Program*, Prentice-Hall, 1987

[4] Hatton, L, *Safer C: Developing for High-Integrity and Safety-Critical Systems*, McGraw-Hill, 1994

## For more information contact:

**Programming Research Ltd**
**Glenbrook House**
**1-11 Molesey Road**
**HERSHAM**
**Surrey**
**KT12 4RH**

| | |
|---|---|
| **Tel:** | **01932 888080** |
| **Fax** | **01932 888081** |
| **E-mail:** | **support@prqa.co.uk** |
| **WWW:** | **http://www.prqa.co.uk** |