RAPITA SYSTEMS
A DANLAW COMPANY

Safety through quality

WHITE PAPER

Automating WCET analysis for DO-178B/C

Contents

1. Introduction

RapiTime

Rapi**Time** is a tool to determine the execution time of embedded software. It reports measured maximum, high-water-mark and calculated worst-case times. The WCET calculation is based on measured data, annotations and the source code structure.

To use Rapi**Time**'s calculated WCET in support of DO-178B or DO-178C objective 6.3.4f, it is necessary both to qualify the tool and to provide a corresponding analysis process.

Determination of worst-case execution time (WCET) is an activity that is recommended in DO-178B and DO-178C 6.3.4f. Here we consider the current state of practice for determining WCET and present the benefits available from automating the process. Thanks to its DO-178B/DO-330 qualification pack, RapiTime uniquely allows these benefits to be realised in DO-178B/C projects.

One approach to determining WCET that is widely used and currently accepted by certification authorities is manual analysis and measurement. Manual analysis and measurement is an effort-intensive task and requires an extremely high level of care to ensure measurements are correctly captured.

By using automation to reduce the level of effort required and increase the accuracy of results, Rapi**Time** represents an evolution of the manual analysis and measurement approach. With the introduction of the DO-178B/DO-330 qualification pack, Rapi**Time** can replace manual activities in projects requiring DO-178B/DO-178C certification. The development of the new qualification pack means Rapi**Time** is the only tool in the marketplace capable of measurement-based WCET analysis for DO-178B.

In this document, we describe:

- What is required for WCET in DO-178B projects.
- The current state of practice, and its limitations.
- How RapiTime represents an improvement on the state of practice.
- How to use RapiTime in a DO-178B environment.

2. DO-178B/C requirements for WCET

DO-178B (and the newer DO-178C) recommendations address many considerations in the development of embedded, real-time software. Timing of software is no exception.

In DO-178B/C, there is no single objective that is solely concerned with timing. However, two objectives include timing considerations, as shown in Table 1 (see pg 5).

In DO-248B ('Final Report for Clarification of DO-178B "Software Considerations in Airborne Systems and Equipment Certification"'), a frequently asked question addresses WCET:

> **3.73 FAQ #73: ARE TIMING MEASUREMENTS DURING TESTING SUFFICIENT OR IS A RIGOROUS DEMONSTRATION OF WORST-CASE TIMING NECESSARY?**
> Reference: ED-12B/DO-178B: Sections 6.3.4 and 11.20
> Keywords: timing; worst-case timing
>
> **Answer:**
>
> In addition to verifying that the software requirements relating to timing have been met, ED-12B/DO-178B states that the worst-case timing should be determined. Section 6.3.4f of ED-12B/DO-178B states that as part of meeting the verification objective of the source code being accurate and consistent, the worst-case timing should be determined by review and analysis for Levels A, B, and C software. The results of this review and analysis should be documented in the Software Accomplishment Summary as timing margins (reference Section 11.20d of ED- 12B/DO-178B).
>
> The worst-case timing could be calculated by review and analysis of the source code and architecture, but compiler and processor behavior and its impact also should be addressed. Timing measurements by themselves cannot be used without an analysis demonstrating that the worst-case timing would be achieved, but processor behavior (e.g. cache performance) should be assessed. Using the times observed during test execution is sufficient, if it can be demonstrated that the test provides worst-case execution time.

Worst-Case Execution Time

Simply put, the worst-case execution time (WCET) of a computational task is the maximum length of time the task could take to execute on a specific hardware platform. This excludes any time spent executing other tasks or interrupts.

It is important to note the last sentence: Timing measurements by themselves cannot be used without an analysis demonstrating that the worst-case timing would be achieved. In other words, testing alone is not adequate for demonstrating worst case execution times – some form of analysis is also required.

Table 1. Timing considerations in DO-178B/C objectives

	Objective		**Applicability (DAL)**				**Output**	
ID	**Description**	**Ref.**	**A**	**B**	**C**	**D**	**Description**	**Ref.**
A-5, 6	Source Code is accurate and consistent.	6.3.4f	●	❍	❍	❍	Software Verification Results	11.14
A-6, 5	Executable Object Code is compatible with target computer.	6.4.3a	❍	❍	❍	❍	Software Verification Cases and Procedures Software Verification Results	11.13 11.14

❍ Objective required at DAL
● Objective required with independence at DAL

Entry A-5, **6** is concerned with reviews and analysis of the source code. It is assumed that, although the items in table A-5 are typically review items, analysis may include some dedicated testing of the source code to substantiate its properties. The timing aspects of 6.3.4f are indicated as below:

6.3.4f: Accuracy and consistency: The objective is to determine the correctness and consistency of the Source Code, including [...] worst-case execution timing [...]

Entry A-6, **5** is concerned with requirements-based hardware/software integration testing. The software should be tested both for conformance to the requirements and for specific error sources associated with operation within the target computer environment. The timing aspects of 6.4.3a are indicated as below:

6.4.3a: Requirements-Based Hardware/Software Integration Testing: This testing method should concentrate on error sources associated with the software [...] Typical errors revealed by this testing method include:

[...]

- Failure to satisfy execution time requirements.

[...]

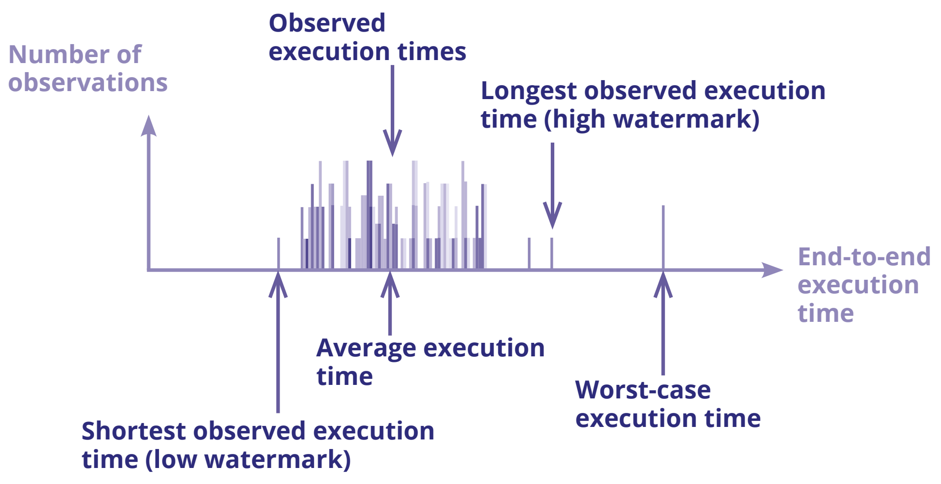

Number of observations
Observed execution times
Longest observed execution time (high watermark)
End-to-end execution time
Average execution time
Worst-case execution time
Shortest observed execution time (low watermark)

Figure 1 – Software timing characteristics

3. Current state of practice for determining WCET

Actual WCET or Estimate?

Despite research over the last fourty years, there is no way to find the exact worst-case execution time (the "actual WCET") for any reasonably large piece of code running on modern processor. Instead, the aim of WCET analysis when applied in the context of engineering projects must be to find a useful and valid approximation of the actual WCET.

Rapita Systems regularly talks to companies that develop avionics systems under DO-178B. We have come to recognize that, although there is variation in state of practice for determining WCET, there is a frequently used approach that is applied and certified in many avionic systems.

3.1 What exactly is the State of Practice?

The manual, measurement-based approach frequently used in the avionic industry typically follows these steps:

1. Put some form of instrumentation in at the start and end of the unit we're interested in finding WCET for:
    - Unit could be a task (i.e. put instrumentation in as first statement of task and last statement of task).
    - Unit could be a complete partition.
    - Unit might be a section of code.
    - Instrumentation is a mechanism that is added to the code running on the target that measures the time spent executing regions of code. Most commonly, instrumentation consists of specific instructions added to the code.
2. There are many approaches to implementing instrumentation. For example, instrumentation might:
    - Toggle an externally visible I/O pin – this could be observed with an oscilloscope or a logic analyzer.
    - Additional code might record the start time and stop time of specific code units. This could then be recorded in memory and retrieved subsequently to derive the high water mark, maximum and minimum execution times.

3. Manually review the code to attempt to identify worst-case paths through the source code.
   Note: this is important to meet the requirement from DO-248B FAQ 73.
4. Devise test cases to drive code through these paths.
5. Record the time taken to execute the code. Frequently, a "safety margin" is added to this value.

3.2 What are limitations of this approach?

- Identifying worst-case paths through code is difficult and effort-intensive:
  - Predicting which areas of code are responsible for large execution times isn't easy.
  - A simple assignment statement (especially in C++ or Ada) might result in a significant number of operations if copying a complex data structure.
  - Some complex-looking groups of statements might be aggressively optimised by the compiler.
- This approach is highly likely to lead to an optimistic WCET (see sidebar). On the positive side, this approach will never report a pessimistic WCET.
- If the running time of the application exceeds its timing budgets, this approach doesn't support the engineer in identifying exactly where to address the problem:
  - Most of the code will not affect the worst-case and can be ignored. If we don't know which code is in this region, we're wasting time looking at it.
  - Some code will affect the worst-case slightly – reducing execution time of this code will have a marginal effect.
  - A small part of the code will have a significant effect on WCET. Finding this code with a manualapproach is difficult.

3.3 Is it possible to improve on this approach?

There are three ways in which the manual approaches can be improved through automation:

1. Make the results much less optimistic/more realistic.
2. Reduce the effort required to identify good test cases. This reduces the risk of reporting a WCET value that is exceeded during actual use.
3. Provide assistance in identifying which parts of the code affect WCET the most.

Optimistic vs. Pessimistic Computed WCET

Techniques for calculating WCET estimates may be optimistic or pessimistic. An optimistic value is one that is less than the actual WCET. Likewise a pessimistic value is one that is greater than the actual value.

Typically, WCET analysis techniques introduce pessimism, so much of the work in WCET analysis is to reduce that pessimism so that the result is practical.

How instrumentation points generate a trace

A trace is constructed by executing instrumentation points. At each instrumentation point, we need to collect an identifier and a timestamp. Many different approaches to collecting identifiers and timestamps exist, depending on your target hardware. The most common approach is to write an identifier to an I/O port or externally visible address or data bus; then use a logic analyzer or Rapita's RTBx to read the value and collect a timestamp. Instrumentation can be implemented in one or two machine instructions.

3.4 Advancing the state of practice – using Rapi**Time**

Rapi**Time**, part of Rapita **Verification Suite** (R**VS**), builds on the current state of practice, automating the process we have previously described.

The areas of the process Rapi**Time** automates are:

1. Instrumentation of source code:
    - As part of the integration phase, the Rapi**Time** source code instrumenter is inserted into your build process.
    - Instrumentation is customizable: you can choose whether to instrument to every decision point in your code, at the level of top-level functions only, or at multiple levels between this.
    - During instrumentation, Rapi**Time** also derives a structural model of the code – it uses this when it identifies worst-case paths.
2. Mapping timing data back to source code:
    - As you run tests on the instrumented code, your target produces a trace (a time-stamped list of Ipoint identifiers that have been encountered). Rapi**Time** will automatically relate the Ipoint identifiers to decision points in the code, and consequently determine the execution time of segments of code.
    - In this process Rapi**Time** also provides a number of timing measurements of individual blocks of code and sub-programs, including minimum and maximum, as well as the high-water mark. This represents the longest observed execution of the code.
    - Rapi**Time** combines the structural model of the code derived during instrumentation with the timing data it derives from the execution trace. Using this combined data, Rapi**Time** predicts the worst-case path through the code and the worst-case execution time.

To do this, we recommend Rapi**Time** be used as follows:

- Using an initial set of tests, use Rapi**Time** to determine a worst-case execution path for your system (Instrument, run tests, run analysis).

3.5 The Rapi**Time** process

- You can use Rapi**Time**'s predicted worst case path to create a test case that takes as long to execute as the computed WCET.
- If the high water mark is less than the worst-case execution time, you can improve the testing and analysis by:
    - Eliminating infeasible paths by introducing annotations.
    - Finding tests that drive the code through the worst-case path.
    - Simplify the source code.
- Repeat this process (Figure 2) until the two values are within an acceptable tolerance of each other.

The benefits of using Rapi**Time** in a process similar to the above are as follows:

1. Improving results. Instrumenting code at a more detailed level than is possible by hand means that the timing measurements are more precise than previously achievable.
2. Reduced effort required to identify good test cases. Automating the detection of the worst-case path, based on evidence from testing, is significantly faster than attempting to manually determine worst-case paths.
3. Identify parts which affect WCET most significantly. If it is necessary to reduce the worst-case execution time, automatic detection of "hot-spots" provides evidence to support specific optimizations.

RapiTime's computed WCET

When performing worst-case execution time analysis, Rapi**Time** reports computed WCET. The computed WCET combines the longest observed execution times with the longest predicted paths.

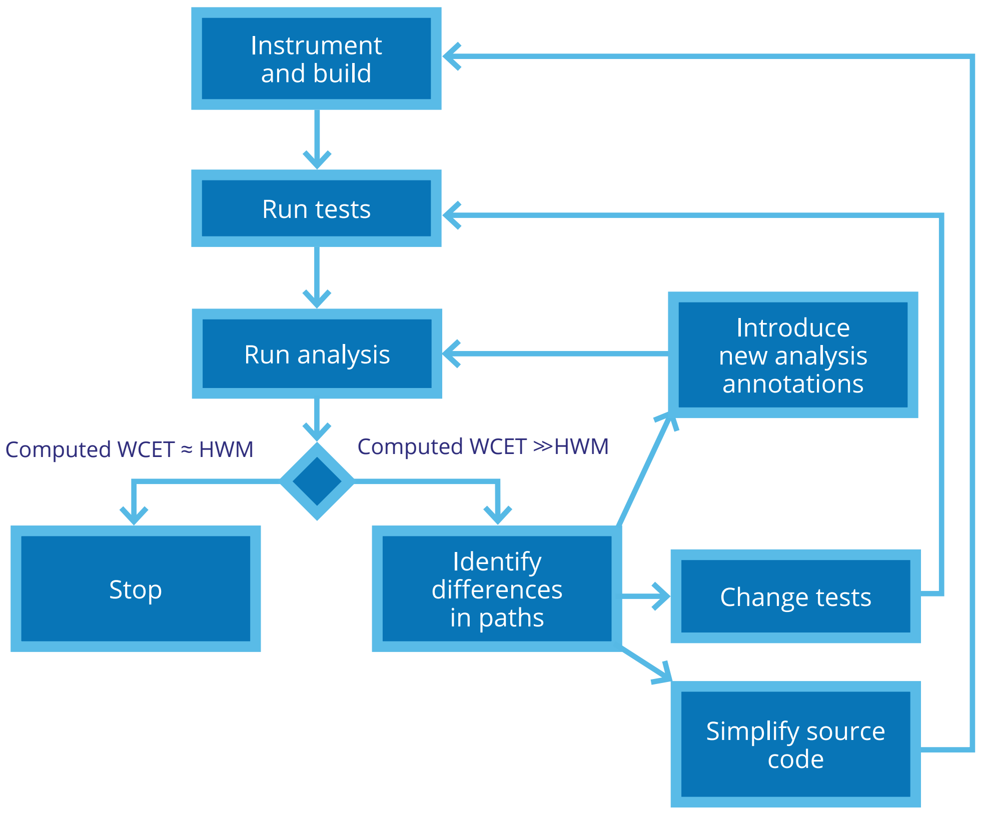


Figure 2 – The Rapi**Time** process

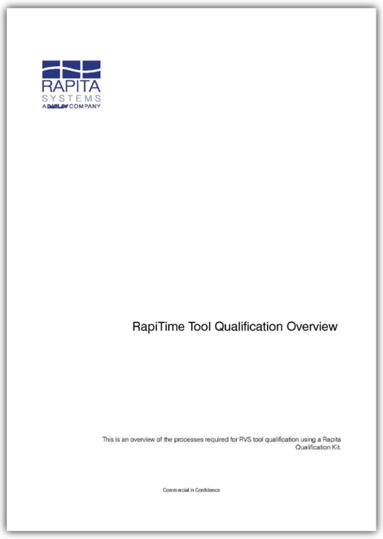
4. Using Rapi**Time** in a DO-178C environment

According to the guidance of DO-178C/DO-330, a software tool used on your project needs to be qualified if:

- The tool could fail to detect an existing error.
- The tool's output could not be verified by another activity.
- Processes are eliminated, reduced or automated by the tool.

Rapi**Time** meets these criteria, so it needs to be qualified.

To qualify Rapi**Time** within your project, you need to provide key information about the tool (Table 2). Some of this information represents general information about Rapi**Time** (for example the Tool Operational Requirements), while other information is specific to using Rapi**Time** in your system, for example the tool V&V records, which include test results from testing your installation of Rapi**Time** in your environment.



RAPITA SYSTEMS A DANLAW COMPANY

RapiTime Tool Qualification Overview

This is an overview of the processes required for RVS tool qualification using a Rapita Qualification Kit.

Commercial in Confidence

Figure 3 – DO-178B/C qualification kit

Table 2. Tool qualification information required for DO-178C

Item	DO-178C reference	DO-330 reference	Usage
PSAC (Plan for Software Aspects of Certification)	12.2, 12.2.3.a, 12.2.4	1.3c	Submit to Certification Authority (CA, e.g. FAA) early to discuss timescale and acceptable qualification methods and documentation approaches
TOR (Tool Operational Requirements)	12.2.3.c(2), 12.2.3.2	5.1	Must be available for review by CA
TAS (Tool Accomplishment Summary)	11.20, 12.2.4	10.1.15	Optional, but simplifies production of PSAC , Submit to CA
TVR (Tool Verification Results)	12.2.3	12.2.3	Must be available for review by CA
TQP (Tool Qualification Plan)	12.2.3a(1), 12.2.3.1 and 12.2.4	10.1.2	Optional, but simplifies production of PSAC

To support your use of RapiTime in a DO-178C project, we provide qualification support through a qualification kit and service.

We provide two main options for supporting tool qualification within avionics projects:

4.1 Qualification Kit

This includes generic evidence to demonstrate that RapiTime meets its requirements in a generic environment. Our qualification kits are based on DO-330: Software Tool Qualification Considerations.

4.2 Qualification Service

Our Qualification Service provides tests you can run to show that the integration of RapiTime with your specific platform is robust, along with expected results from these tests. When delivering the service, Rapita Engineers work with you to run the tests and review results.

Table 3. Rapi**Time** Tool Qualification Options and Components

		Qualification Kit	Qualification Service
Developer documents	TQP and TVR	✓	✓
	TQP and TAS	✓	✓
	Configuration Assessment Guide	✓	✓
Reference documents	User Guides	✓	✓
	Installation Guide	✓	✓
	Troubleshooting Guide	✓	✓
	Recommended Workflow	✓	✓
Tool user documentation templates	TQP and TVR	✓	✓
	TQP and TAS	✓	✓
Additional support	Ongoing assurance impact assessment	✓	✓
	Configuration testing		✓
	Consultation assessment		✓
	Consultation, liaison, assurance analysis		✓

5. Want to learn more?

If you want to learn more about worst-case execution time, visit our website where you gain access to a wide range of white papers and videos on the topic.

www.rapitasystems.com/worst-case-execution-time

Rapita Systems regularly releases new material and runs training courses on multicore timing analysis worldwide. To make sure you're notified, sign up to our mailing list.

www.rapitasystems.com/newsletter

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 249100, PROARTIS.

About Rapita

Rapita Systems provides on-target software verification tools and services globally to the embedded aerospace and automotive electronics industries.

Our solutions help to increase software quality, deliver evidence to meet safety and certification objectives and reduce costs.

Find out more

A range of free high-quality materials are available at:
rapitasystems.com/downloads

SUPPORTING CUSTOMERS WITH:

Tools	Engineering Services	Multicore verification
Rapita **Verification Suite**:	V&V Services	**MACH**^{178}
Rapi**Test**	Integration Services	Multicore Timing Solution
Rapi**Cover**	Qualification	
Rapi**Time**	SW/HW Engineering	
Rapi**Task**	Compiler Verification	

Contact

Rapita Systems Ltd.
Atlas House
York, YO10 3JB
UK

+44 (0)1904 413945

Rapita Systems, Inc.
41131 Vincenti Ct.
Novi, Mi, 48375
USA

+1 248-957-9801

Rapita Systems S.L.
Parc UPC, Edificio K2M
c/ Jordi Girona, 1-3
Barcelona 08034
Spain

+34 93 351 02 05

rapitasystems.com

linkedin.com/company/rapita-systems

info@rapitasystems.com