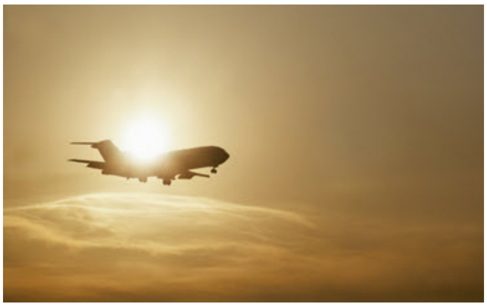
ESCHER TECHNOLOGIES

eCv++

ESCHER C++ VERIFIER



```
template<class X> class Queue final
{
  X ring[bufsize];
  typedef size_t invariant
     (value < bufsize) RingPointer;
  RingPointer hd, tl;

public:
  void add(X x) writes(*this)
  pre(!full())
  post(storedData() ==
       old(storedData()).append(x))
```

Escher C & C++ Verifier

Now you can write provably-correct software in C++!

Proving that software meets its functional specifications has traditionally required specialist computer languages and skills. The Escher C and C++ Verifier brings provable correctness within reach of many more software developers - by combining precise specification, advanced automated reasoning, and popular programming language choices for developing embedded software.

What is Escher C++ Verifier?

eCv++ is a tool that empowers you to develop critical embedded software in C and C++ together with proofs of its correctness, robustness and security. Building on our established product Escher C Verifier, *eCv++* gives you access to safety- and productivity-enhancing features of C++ such as encapsulation, inheritance and templates.

Use the C and C++ languages safely

eCv++ uses carefully-chosen verifiable subsets of the C and C++ languages that avoid classic vulnerabilities and strengthen the type system. If you're already coding to the MISRA-C, MISRA-C++ or JSF-C++ standard, you'll find that your programs are well on the way to being compatible with *eCv++*.

What does eCv++ prove?

eCv++ always tries to prove that your program is free from out-of-bounds array indexing, null pointer de-referencing, arithmetic overflow and other "undefined behaviour", and that the loops in your program terminate. If you use inheritance and virtual functions, *eCv++* also tries to prove type consistency between derived classes and their parents as required by DO-332 - the object-oriented supplement to DO-178C. If you write annotations to express functional specifications and safety properties, *eCv++* will attempt to prove them too.

Easy to learn and use

The automated reasoning technology of *eCv++* avoids the need for user involvement in constructing proofs. Additionally, if *eCv++* fails to find a proof, it will often suggest the missing precondition or invariant that makes proof possible. These features make *eCv++* easier to learn and use than traditional proof tools.

Easy to introduce into your process

If you want to apply *eCv++* to existing software, you don't need to verify it all at once. With *eCv++* you can annotate and verify individual C and C++ source files, if you provide minimal specifications for any external functions they call.

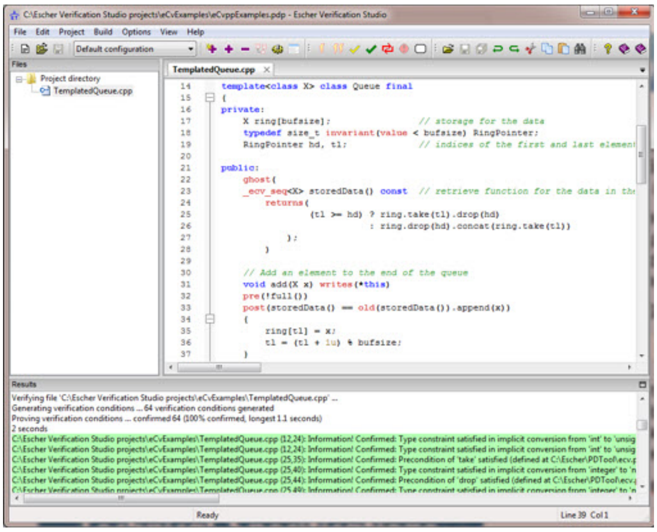
Built with mature technology

eCv++ uses the same Verified Design-by-Contract paradigm and powerful automated reasoning engine as *Perfect Developer* - the formal modelling tool used industrially to model and verify diverse applications, including SIL 4 defence software and business application logic.

Find out more today!

To discuss how *eCv++* can help you develop more reliable C and C++ software in less time, email **critical@eschertech.com** or telephone us on +44(0)20 8144 3265.

Escher C++ Verifier in use



```
template<class X> class Queue final
{
private:
    X ring[bufsize];                        // storage for the data
    typedef size_t invariant(value < bufsize) RingPointer;
    RingPointer hd, tl;                     // indices of the first and last element

public:
    ghost(
    _ecv_seq<X> storedData() const  // retrieve function for the data in the
        returns(
                (tl >= hd) ? ring.take(tl).drop(hd)
                           : ring.drop(hd).concat(ring.take(tl))
            );
        )

    // Add an element to the end of the queue
    void add(X x) writes(*this)
    pre(!full())
    post(storedData() == old(storedData()).append(x))
    {
        ring[tl] = x;
        tl = (tl + 1u) % bufsize;
    }
```

```
Verifying file 'C:\Escher Verification Studio projects\eCvExamples\TemplatedQueue.cpp' ...
Generating verification conditions ... 64 verification conditions generated
Proving verification conditions ... confirmed 64 (100% confirmed, longest 1.1 seconds)
2 seconds
C:\Escher Verification Studio projects\eCvExamples\TemplatedQueue.cpp (12,24): Information! Confirmed: Type constraint satisfied in implicit conversion from 'int' to 'unsig
C:\Escher Verification Studio projects\eCvExamples\TemplatedQueue.cpp (12,24): Information! Confirmed: Type constraint satisfied in implicit conversion from 'int' to 'unsig
C:\Escher Verification Studio projects\eCvExamples\TemplatedQueue.cpp (25,35): Information! Confirmed: Precondition of 'take' satisfied (defined at C:\Escher\PDTool\ecv.p
C:\Escher Verification Studio projects\eCvExamples\TemplatedQueue.cpp (25,40): Information! Confirmed: Type constraint satisfied in implicit conversion from 'integer' to 'n
C:\Escher Verification Studio projects\eCvExamples\TemplatedQueue.cpp (25,44): Information! Confirmed: Precondition of 'drop' satisfied (defined at C:\Escher\PDTool\ecv.
C:\Escher Verification Studio projects\eCvExamples\TemplatedQueue.cpp (25,49): Information! Confirmed: Type constraint satisfied in implicit conversion from 'integer' to 'n
```

Technical Specifications

Development platform requirements

PC with fast x64 processor and 4Gb or more main memory.

Windows 7 or 10 operating system, 64-bit (contact us if you require a Linux edition).

Supported source code languages

Formally-verifiable subsets of C'90, C'99, C++'03 and C++'11, including the majority of constructs permitted by MISRA-C, MISRA-C++ and JSF-C++.

Output

Verification summary displayed in user interface. Verification report can be saved to file. Analysis of unproven verification condition and associated suggestions saved to file in a choice of formats. Successful proofs of verification conditions can be saved to file in a choice of formats (HTML, LaTeX, or plain text).

What others say about us

"Our need is to meet the requirements of defence standard 00-55 to Safety Integrity Level 4. Escher Technologies software met our requirements best."

"We were especially impressed by the automation of verification proofs, which will substantially reduce our costs, and by the level of support provided by Escher Technologies."

Guy Mason, General Dynamics UK Ltd.

"Escher Technologies software follows a very pragmatic methodology to provide proven software. We were impressed by the automation and ease of adoption of the verification proofs."

Douglas Eadie, Bitwise Ltd.

"We have used Perfect Developer for about four years and we have received excellent support from Escher Technologies throughout."

John Warren, Precision Design Technology Ltd.

About Escher Technologies

Escher Technologies was founded in 1995 to research and develop leading-edge software development technology.

Our mission is to reduce the cost of developing dependable software, so that reliability can be the norm rather than the exception, even for non-critical software.

Although our team has a strong commercial background, we maintain close links with the automated reasoning and formal methods research communities in leading universities worldwide.

For more information visit **http://www.eschertech.com** or email **critical@eschertech.com**

eCV++
ESCHER C++ VERIFIER

ESCHER
TECHNOLOGIES

October 2016. This datasheet contains pre-release information about a forthcoming product. Specifications are subject to change without notice.
MISRA is a trademark of MIRA Limited.

Escher Technologies Limited, Woodpeckers, Wearne, Langport TA10 0QQ, United Kingdom.
Telephone: +44 (0)20 8144 3265   Registered in England no. 3128474