The AUTOSAR coding standard was created by an alliance of hundreds of automotive manufacturers, developers, and suppliers.

Formerly called "Guidelines for the use of the C++14 language in critical and safety-related systems", the standards consists of nearly 400 requirements that industry experts agree will mitigate bugs, remove inconsistencies and clarify C++ code.

	All Rules	Advisory Rules	Required Rules
Understand % Coverage	91%	94%	90%
Understand Coverage	364	33	330
Total Rules	401	35	365

Checks

Check ID	Check Name	Supported	Automation	Category
A0-1-1	A project shall not contain instances of non-volatile variables being given values that are not subsequently used	Yes	Automated	Required
A0-1-2	The value returned by a function shall be used	Yes	Automated	Required
A0-1-3	Every function defined in an anonymous namespace, or static function with internal linkage, or private member function shall be used	Yes	Automated	Required
A0-1-4	There shall be no unused named parameters in non-virtual functions	Yes	Automated	Required
A0-1-5	There shall be no unused named parameters in the set of parameters for a virtual function and all the functions that override it	Yes	Automated	Required
A0-1-6	There should be no unused type declarations	Yes	Automated	Advisory
A0-4-1	Floating-point	No	Non-automated	Required

	implementation shall comply with IEEE 754 standard			
A0-4-2	Type long double shall not be used	Yes	Automated	Required
A0-4-3	The implementations in the chosen compiler shall strictly comply with the C++14 Language Standard	No	Automated	Required
A0-4-4	Range, domain and pole errors shall be checked when using math functions	No	Non-automated	Required
A1-1-1	All code shall conform to ISO/IEC 14882:2014 - Programming Language C++ and shall not use deprecated features	No	Automated	Required
A1-1-2	A warning level of the compilation process shall be set in compliance with project policies	No	Non-automated	Required
A1-1-3	An optimization option that disregards strict standard compliance shall not be turned on in the chosen compiler	No	Non-automated	Required
A1-2-1	When using a compiler toolchain, in safety-related software, the tool confidence level (TCL) shall be determined	No	Non-automated	Required
A1-4-1	Code metrics and their valid boundaries shall be defined and code shall comply with defined boundaries of code metrics	No	Non-automated	Required
A1-4-3	All code should compile free of compiler warnings	Yes	Automated	Advisory
A2-3-1	Only those characters	Yes	Automated	Required

	specified in the C++ Language Standard basic source character set shall be used in the source code			
A2-5-1	2-3-1 Trigraphs shall not be used	Yes	Automated	Required
A2-5-2	Digraphs shall not be used	Yes	Automated	Required
A2-7-1	The character \ shall not occur as a last character of a C++ comment	Yes	Automated	Required
A2-7-2	Sections of code shall not be "commented out"	Yes	Non-automated	Required
A2-7-3	All declarations of "user-defined" types, static and non-static data members, functions and methods shall be preceded by documentation	Yes	Automated	Required
A2-7-5	Comments shall not document any actions or sources (e.g. tables, figures, paragraphs, etc.) that are outside of the file	No	Non-automated	Required
A2-8-1	A header file name should reflect the logical entity for which it provides declarations	Yes	Non-automated	Required
A2-8-2	An implementation file name should reflect the logical entity for which it provides definitions	Yes	Non-automated	Advisory
A2-10-1	Shadowed Identifiers	Yes	Automated	Required
A2-10-4	The identifier name of a non-member object with static storage duration or static function shall not be reused within a namespace	Yes	Automated	Required

A2-10-5	An identifier name of a function with static storage duration or a non-member object with external or internal linkage should not be reused	Yes	Automated	Advisory
A2-10-6	A class or enumeration name shall not be hidden by a variable, function or enumerator declaration in the same scope	Yes	Automated	Required
A2-11-1	Volatile keyword shall not be used	Yes	Automated	Required
A2-13-1	Only those escape sequences that are defined in ISO/IEC 14882:2014 shall be used	Yes	Automated	Required
A2-13-2	Concatenating String Literals of Different Encodings	Yes		
A2-13-3	Type wchar_t shall not be used	Yes	Automated	Required
A2-13-4	String literals shall not be assigned to non-constant pointers	Yes	Automated	Required
A2-13-5	Hexadecimal constants should be upper case	Yes	Automated	Advisory
A2-13-6	Universal character names shall be used only inside character or string literals	Yes	Automated	Required
A3-1-1	It shall be possible to include any header file in multiple translation units without violating the One Definition Rule	Yes	Automated	Required
A3-1-2	Header files, that are defined locally in the project, shall have a file name extension of one	Yes	Automated	Required

	of: ".h", ".hpp" or ".hxx"			
A3-1-3	Implementation files, that are defined locally in the project, should have a file name extension of ".cpp"	Yes	Automated	Advisory
A3-1-4	When an array with external linkage is declared, its size shall be stated explicitly	Yes	Automated	Required
A3-1-5	A function definition shall only be placed in a class definition if (1) the function is intended to be inlined (2) it is a member function template (3) it is a member function of a class template	Yes	Partially Automated	Required
A3-1-6	Trivial accessor and mutator functions should be inlined.	Yes	Automated	Advisory
A3-3-1	Objects or functions with external linkage (including members of named namespaces) shall be declared in a header file	Yes	Automated	Required
A3-3-2	Static and thread-local objects shall be constant-initialized	Yes	Automated	Required
A3-8-1	An object shall not be accessed outside of its lifetime	No	Automated	Required
A3-9-1	Fixed Width Integers	Yes	Automated	Required
A4-5-1	Expressions with type enum or enum class shall not be used as operands to built-in and overloaded operators other than the subscript operator [ ], the	Yes	Automated	Required

	assignment operator =, the equality operators == and ! =, the unary & operator, and the relational operators <, <=, >, >=			
A4-7-1	An integer expression shall not lead to data loss.	Yes	Automated	Required
A4-10-1	Only nullptr literal shall be used as the null-pointer-constant	Yes	Automated	Required
A5-0-1	The value of an expression shall be the same under any order of evaluation that the standard permits	Yes	Automated	Required
A5-0-2	Condition of if statement shall be bool	Yes	Automated	Required
A5-0-3	No more than 2 levels of pointer indirection	Yes	Automated	Required
A5-0-4	Pointer arithmetic shall not be used with pointers to non-final classes	Yes	Automated	Required
A5-1-1	Literal values shall not be used apart from type initialization, otherwise symbolic names shall be used instead	Yes	Automated	Required
A5-1-2	Variables shall not be implicitly captured in a lambda expression	Yes	Automated	Required
A5-1-3	Parameter list (possibly empty) shall be included in every lambda expression	Yes	Automated	Required
A5-1-4	A lambda expression object shall not outlive any of its reference-captured objects	Yes	Automated	Required
A5-1-6	Specify Lambda Return Type	Yes	Automated	Advisory

A5-1-7	A lambda shall not be an operand to decltype or typeid	Yes	Automated	Required
A5-1-8	Lambda expressions should not be defined inside another lambda expression	Yes	Automated	Advisory
A5-1-9	Identical unnamed lambda expressions shall be replaced with a named function or a named lambda expression	Yes	Automated	Advisory
A5-2-1	dynamic_cast should not be used	Yes	Automated	Advisory
A5-2-2	Traditional C-style casts shall not be used	Yes	Automated	Required
A5-2-3	A cast shall not remove any const or volatile qualification from the type of a pointer or reference	Yes	Automated	Required
A5-2-4	reinterpret_cast shall not be used	Yes	Automated	Required
A5-2-5A	An array or container shall not be accessed beyond its range (Part A)	Yes	Automated	Required
A5-2-5B	An array or container shall not be accessed beyond its range Part B	Yes	Automated	Required
A5-2-6	Operands of Logical Boolean Operators	Yes	Automated	Required
A5-3-1	Evaluation of the operand to the typeid operator shall not contain side effects.	Yes	Non-automated	Required
A5-3-2	Before dereferencing a pointer, compare it with NULL	Yes	Partially Automated	Required
A5-3-3	Deleting Pointers to Incomplete Class Types	Yes	Automated	Required

A5-5-1	A pointer to member shall not access non-existent class members	Yes	Automated	Required
A5-6-1A	The right hand operand of the integer division or remainder operators shall not be equal to zero	Yes	Automated	Required
A5-6-1B	The right hand operand of the integer division or remainder operators shall not be equal to zero	Yes	Automated	Required
A5-10-1	A pointer to member virtual function shall only be tested for equality with null-pointer-constant	Yes	Automated	Required
A5-16-1	The ternary conditional operator shall not be used as a sub-expression	Yes	Automated	Required
A6-2-1	Move and copy assignment operators shall either move or respectively copy base classes and data members of a class, without any side effects	Yes	Automated	Required
A6-2-2	Explicit Calls to Constructors of Temporary Objects	Yes	Automated	Required
A6-4-1	A switch statement shall have at least two case-clauses, distinct from the default label	Yes	Automated	Required
A6-5-1	A for-loop that loops through all elements of the container and does not use its loop-counter shall not be used	Yes	Automated	Required
A6-5-2	A for loop shall contain a single loop-counter which shall not have	Yes	Automated	Required

	floating-point type			
A6-5-3	Do statements should not be used	Yes	Automated	Advisory
A6-5-4	For-init-statement and expression should not perform actions other than loop-counter initialization and modification	Yes	Automated	Advisory
A6-6-1	The goto statement shall not be used.	Yes	Automated	Required
A7-1-1	Constexpr or const specifiers shall be used for immutable data declaration	Yes	Automated	Required
A7-1-2	The constexpr specifier shall be used for values that can be determined at compile time	Yes	Automated	Required
A7-1-3	CV-qualifiers shall be placed on the right hand side of the type that is a typedef or a using name	Yes	Automated	Required
A7-1-4	The register keyword shall not be used	Yes	Automated	Required
A7-1-5	The auto specifier shall not be used apart from following cases: (1) to declare that a variable has the same type as return type of a function call, (2) to declare that a variable has the same type as initializer of non-fundamental type, (3) to declare parameters of a generic lambda expression, (4) to declare a function template using trailing return type syntax	Yes	Automated	Required
A7-1-6	The typedef specifier	Yes	Automated	Required

	shall not be used			
A7-1-7	Each expression statement and identifier declaration shall be placed on a separate line	Yes	Automated	Required
A7-1-8	A non-type specifier shall be placed before a type specifier in a declaration.	Yes	Automated	Required
A7-1-9	A class, structure, or enumeration shall not be declared in the definition of its type	Yes	Automated	Required
A7-2-1	An expression with enum underlying type shall only have values corresponding to the enumerators of the enumeration	Yes	Automated	Required
A7-2-2	Enumeration underlying base type shall be explicitly defined	Yes	Automated	Required
A7-2-3	Enumerations shall be declared as scoped enum classes	Yes	Automated	Required
A7-2-4	In an enumeration, either (1) none, (2) the first or (3) all enumerators shall be initialized	Yes	Automated	Required
A7-2-5	Enumerations should be used to represent sets of related named constants	No	Non-automated	Advisory
A7-3-1	Overloaded Function Not Visible From Where it is Called	Yes	Automated	Required
A7-4-1	The asm declaration shall not be used.	Yes	Automated	Required
A7-5-1	A function shall not return a reference or a pointer to a parameter that is passed by reference to const.	Yes	Automated	Required

A7-5-2	Functions shall not call themselves, either directly or indirectly.	Yes	Automated	Required
A7-6-1	Functions declared with the [[noreturn]] attribute shall not return	Yes	Automated	Required
A8-2-1	When declaring function templates, the trailing return type syntax shall be used if the return type depends on the type of parameters.	Yes	Automated	Required
A8-4-1	Functions shall not be defined using the ellipsis notation.	Yes	Automated	Required
A8-4-2	Always return a value in non-void functions	Yes	Automated	Required
A8-4-3	Common ways of passing parameters should be used.	Yes	Automated	Required
A8-4-4	Multiple output values from a function should be returned as a struct or tuple.	Yes	Automated	Advisory
A8-4-5	"consume" parameters declared as X && shall always be moved from.	Yes	Automated	Required
A8-4-6	"forward" parameters declared as T && shall always be forwarded.	Yes	Automated	Required
A8-4-7	"in" parameters for "cheap to copy" types shall be passed by value.	Yes	Automated	Required
A8-4-8	Output parameters shall not be used.	Yes	Automated	Required
A8-4-9	"in-out" parameters declared as T & shall be modified.	Yes	Automated	Required
A8-4-10	A parameter shall be passed by reference if it can't be NULL	Yes	Automated	Required

A8-4-11	A smart pointer shall only be used as a parameter type if it expresses lifetime semantics	Yes	Automated	Required
A8-4-12	Invalid Use of std::unique_ptr	Yes	Automated	Required
A8-4-13	Invalid Use of std::shared_ptr	Yes	Automated	Required
A8-4-14	Interfaces shall be precisely and strongly typed	No	Non-automated	Required
A8-5-0	Uninitialized Memory Read	Yes	Automated	Required
A8-5-1	Incorrect Order of Initialization	Yes	Automated	Required
A8-5-2	Initializing Variables Without Using Braced-Initialization	Yes	Automated	Required
A8-5-3	Auto Variable	Yes	Automated	Required
A8-5-4	Class Constructor with Parameter Type std::initializer_list	Yes	Automated	Advisory
A9-3-1	Member functions shall not return non-const raw pointers or references to private or protected data owned by the class	Yes	Automated	Required
A9-5-1	Unions Shall not be Used	Yes	Automated	Required
A9-6-1	Data types used for interfacing	Yes	Partially Automated	Required
A9-6-2	Bit-fields shall be used only when interfacing to hardware or conforming to communication protocols	No	Non-automated	Required
A10-0-1	Public Inheritance not Used in a "is-a" Relationship	Yes	Non-automated	Required
A10-0-2	Membership or non-public inheritance shall be used to implement	No	Non-automated	Required

	"has-a" relationship			
A10-1-1	Multiple Base Classes	Yes	Automated	Required
A10-2-1	Non-virtual public or protected member functions shall not be redefined in derived classes	Yes	Automated	Required
A10-3-1	Virtual function declaration shall contain exactly one of the three specifiers: (1) virtual, (2) override, (3) final	Yes	Automated	Required
A10-3-2	Use Override	Yes	Automated	Required
A10-3-3	Virtual functions shall not be introduced in a final class	Yes	Automated	Required
A10-3-5	User-defined assignment operator shall not be virtual	Yes	Automated	Required
A10-4-1	Hierarchies should be based on interface classes	Yes	Non-automated	Advisory
A11-0-1	A non-POD type should be defined as class	Yes	Automated	Advisory
A11-0-2	A type defined as struct shall: (1) provide only public data members, (2) not provide any special member functions or methods, (3) not be a base of another struct or class, (4) not inherit from another struct or class	Yes	Automated	Required
A11-3-1	Friend declarations shall not be used.	Yes	Automated	Required
A12-0-1	If a class declares a copy or move operation, or a destructor, either via "=default", "=delete", or via a user-provided declaration, then all others of these five	Yes	Automated	Required

	special member functions shall be declared as well.			
A12-0-2	Bitwise operations and operations that assume data representation in memory shall not be performed on objects.	Yes	Automated	Required
A12-1-1	Constructors shall explicitly initialize all virtual base classes, all direct non-virtual base classes and all non-static data members.	Yes	Automated	Required
A12-1-2	Both NSDMI and a non-static member initializer in a constructor shall not be used in the same type.	Yes	Automated	Required
A12-1-3	If all user-defined constructors of a class initialize data members with constant values that are the same across all constructors, then data members shall be initialized using NSDMI instead.	Yes	Automated	Required
A12-1-4	All constructors that are callable with a single argument of fundamental type shall be declared explicit.	Yes	Automated	Required
A12-1-5	Common class initialization for non-constant members shall be done by a delegating constructor.	Yes	Partially Automated	Required
A12-1-6	Derived classes that do not need further explicit initialization and require all the constructors from	No	Automated	Required

	the base class shall use inheriting constructors			
A12-4-1	Destructor of a base class shall be public virtual, public override or protected non-virtual	Yes	Automated	Required
A12-4-2	If a public destructor of a class is non-virtual, then the class should be declared final.	Yes	Automated	Advisory
A12-6-1	All class data members that are initialized by the constructor shall be initialized using member initializers.	Yes	Automated	Required
A12-7-1	If the behavior of a user-defined special member function is identical to implicitly defined special member function, then it shall be defined =default or be left undefined.	Yes	Automated	Required
A12-8-1	Move and copy constructors shall move and respectively copy base classes and data members of a class, without any side effects	Yes	Automated	Required
A12-8-2	User-defined copy and move assignment operators should use user-defined no-throw swap function.	Yes	Automated	Advisory
A12-8-3	Moved-from object shall not be read-accessed.	Yes	Partially Automated	Advisory
A12-8-4	Move constructor shall not initialize its class members and base classes using copy semantics.	Yes	Automated	Required
A12-8-5	A copy assignment and a move assignment	Yes	Automated	Required

	operators shall handle self-assignment.			
A12-8-6	Copy and move constructors and copy assignment and move assignment operators shall be declared protected or defined "=delete" in base class.	Yes	Automated	Required
A12-8-7	Assignment operators should be declared with the ref-qualifier &.	Yes	Automated	Advisory
A13-1-2	User defined suffixes of the user defined literal operators shall start with underscore followed by one or more letters	Yes	Automated	Required
A13-1-3	User defined literals operators shall only perform conversion of passed parameters	Yes	Automated	Required
A13-2-1	An assignment operator shall return a reference to "this"	Yes	Automated	Required
A13-2-2	A binary arithmetic operator and a bitwise operator shall return a "prvalue"	Yes	Automated	Required
A13-2-3	A relational operator shall return a boolean value	Yes	Automated	Required
A13-3-1	A function that contains "forwarding reference" as its argument shall not be overloaded	Yes	Automated	Required
A13-5-1	If "operator[]" is to be overloaded with a non-const version, const version shall also be implemented	Yes	Automated	Required
A13-5-2	All user-defined conversion operators shall be defined explicit	Yes	Automated	Required

A13-5-3	User-defined conversion operators should not be used	Yes	Automated	Advisory
A13-5-4	If two opposite operators are defined, one shall be defined in terms of the other	Yes	Automated	Required
A13-5-5	Comparison operators shall be non-member functions with identical parameter types and noexcept	Yes	Automated	Required
A13-6-1	Digit sequences separators ' shall only be used as follows: (1) for decimal, every 3 digits, (2) for hexadecimal, every 2 digits, (3) for binary, every 4 digits	Yes	Automated	Required
A14-1-1	A template should check if a specific template argument is suitable for this template	Yes	Non-automated	Advisory
A14-5-1	A template constructor shall not participate in overload resolution for a single argument of the enclosing class type	Yes	Automated	Required
A14-5-2	Class members that are not dependent on template class parameters should be defined in a separate base class	Yes	Partially Automated	Advisory
A14-5-3	A non-member generic operator shall only be declared in a namespace that does not contain class (struct) type, enum type or union type declarations	Yes	Automated	Advisory
A14-7-1	A type used as a	Yes	Automated	Required

	template argument shall provide all members that are used by the template			
A14-7-2	Template specialization shall be declared in the same file as the primary template	Yes	Automated	Required
A14-8-2	Explicit specializations of function templates shall not be used	Yes	Automated	Required
A15-0-1	A function shall not exit with an exception if it is able to complete its task	No	Non-automated	Required
A15-0-2	At least the basic guarantee for exception safety shall be provided for all operations. In addition, each function may offer either the strong guarantee or the nothrow guarantee	No	Partially Automated	Required
A15-0-3	Exception safety guarantee of a called function shall be considered	No	Non-automated	Required
A15-0-4	Unchecked exceptions shall be used to represent errors from which the caller cannot reasonably be expected to recover.	No	Non-automated	Required
A15-0-5	Checked exceptions shall be used to represent errors from which the caller can reasonably be expected to recover	No	Non-automated	Required
A15-0-6	An analysis shall be performed to analyze the failure modes of exception handling	No	Non-automated	Required
A15-0-7	Exception handling	No	Partially Automated	Required

	mechanism shall guarantee a deterministic worst-case time execution time			
A15-0-8	A worst-case execution time (WCET) analysis shall be performed to determine maximum execution time constraints of the software, covering in particular the exceptions processing	No	Non-automated	Required
A15-1-1	Only instances of types derived from std::exception should be thrown	Yes	Automated	Advisory
A15-1-2	An exception object shall not be a pointer	Yes	Automated	Required
A15-1-3	All thrown exceptions should be unique	Yes	Automated	Advisory
A15-1-4	If a function exits with an exception, then before a throw, the function shall place all objects/ resources that the function constructed in valid states or it shall delete them	Yes	Partially Automated	Required
A15-1-5	Exceptions thrown across execution boundaries	Yes	Non-automated	Required
A15-2-1	Constructors that are not noexcept shall not be invoked before program startup	Yes	Automated	Required
A15-2-2	If a constructor is not noexcept and the constructor cannot finish object initialization, then it shall deallocate the object's resources and it	Yes	Partially Automated	Required

	shall throw an exception			
A15-3-2	If a function throws an exception, it shall be handled when meaningful actions can be taken, otherwise it shall be propagated	No	Non-automated	Required
A15-3-3	Unhandled Exceptions on Main Function	Yes	Partially Automated	Required
A15-3-4	Catch-all (ellipsis and std::exception) handlers shall be used only in (a) main, (b) task main functions, (c) in functions that are supposed to isolate independent components and (d) when calling third-party code that uses exceptions not according to AUTOSAR C++14 guidelines	Yes	Non-automated	Required
A15-3-5	A class type exception shall be caught by reference or const reference	Yes	Automated	Required
A15-4-1	Dynamic exception-specification shall not be used	Yes	Automated	Required
A15-4-2	If a function is declared to be noexcept, noexcept(true) or noexcept(<truecondition>), then it shall not exit with an exception	Yes	Automated	Required
A15-4-3	The noexcept specification of a function shall either be identical across all translation units, or identical or more	Yes	Automated	Required

	restrictive between a virtual member function and an overrider			
A15-4-4	A declaration of non-throwing function shall contain noexcept specification	Yes	Automated	Required
A15-4-5	Checked exceptions that could be thrown from a function shall be specified together with the function declaration and they shall be identical in all function declarations and for all its overriders.	Yes	Automated	Required
A15-5-1	All user-provided class destructors, deallocation functions, move constructors, move assignment operators and swap functions shall not exit with an exception. A noexcept exception specification shall be added to these functions as appropriate	Yes	Automated	Required
A15-5-2	Program shall not be abruptly terminated	Yes	Automated	Required
A15-5-3	The std::terminate() function shall not be called implicitly	Yes	Automated	Required
A16-0-1	Incorrect Use of Pre-processor	Yes	Automated	Required
A16-2-1	Header File Name	Yes	Automated	Required
A16-2-2	There shall be no unused include directives (slow)	Yes	Automated	Required
A16-2-3	An include directive shall be added explicitly for every symbol used in a file	Yes	Non-automated	Required
A16-6-1	#error directive shall not	Yes	Automated	Required

	be used			
A16-7-1	The #pragma directive shall not be used	Yes	Automated	Required
A17-0-1	Reserved Builtin Macros	Yes	Automated	Required
A17-0-2	All project's code including used libraries and any third-party user code shall conform to the AUTOSAR C++14 Coding Guidelines	No	Non-automated	Required
A17-1-1	Use of the C Standard Library shall be encapsulated and isolated	No	Non-automated	Required
A17-6-1	Non-standard entities shall not be added to standard namespaces	Yes	Automated	Required
A18-0-1	The C library facilities shall only be accessed through C++ library headers	Yes	Automated	Required
A18-0-2	The error state of a conversion from string to a numeric value shall be checked	Yes	Automated	Required
A18-0-3	Library <clocale> (locale.h)	Yes	Automated	Required
A18-1-1	C-style Array	Yes	Automated	Required
A18-1-2	The std::vector<bool> specialization shall not be used	Yes	Automated	Required
A18-1-3	The std::auto_ptr type shall not be used	Yes	Automated	Required
A18-1-4	A pointer pointing to an element of an array of objects shall not be passed to a smart pointer of single object type	Yes	Automated	Required
A18-1-6	All std::hash specializations for user-	Yes	Automated	Required

	defined types shall have a noexcept function call operator			
A18-5-1	Functions malloc, calloc, realloc and free shall not be used	Yes	Automated	Required
A18-5-2	Non-placement new or delete expressions shall not be used	Yes	Partially Automated	Required
A18-5-3	The form of the delete expression shall match the form of the new expression used to allocate the memory	Yes	Automated	Required
A18-5-4	If a project has a sized or unsized version of operator "delete" globally defined, then both sized and unsized versions shall be defined	Yes	Automated	Required
A18-5-5	Memory management functions shall ensure the following	No	Partially Automated	Required
A18-5-6	An analysis shall be performed to analyze the failure modes of dynamic memory management	No	Non-automated	Required
A18-5-7	Dynamic Memory Usage on Realtime Phase	Yes	Non-automated	Required
A18-5-8	Objects that do not outlive a function shall have automatic storage duration	Yes	Partially Automated	Required
A18-5-9	New Method Throwing an Exception	Yes	Automated	Required
A18-5-10	Placement new shall be used only with properly aligned pointers to sufficient storage capacity	No	Automated	Required
A18-5-11	operator "new" and operator "delete" shall	Yes	Automated	Required

	be defined together			
A18-9-1	The std::bind shall not be used	Yes	Automated	Required
A18-9-2	Forwarding values to other functions shall be done via: (1) std::move if the value is an rvalue reference, (2) std::forward if the value is forwarding reference	Yes	Automated	Required
A18-9-3	The std::move shall not be used on objects declared const or const&	Yes	Automated	Required
A18-9-4	An argument to std::forward shall not be subsequently used	Yes	Automated	Required
A20-8-1	An already-owned pointer value shall not be stored in an unrelated smart pointer	Yes	Automated	Required
A20-8-2	A std::unique_ptr shall be used to represent exclusive ownership	Yes	Automated	Required
A20-8-3	A std::shared_ptr shall be used to represent shared ownership	Yes	Automated	Required
A20-8-4	A std::unique_ptr shall be used over std::shared_ptr if ownership sharing is not required	Yes	Automated	Required
A20-8-5	std::make_unique shall be used to construct objects owned by std::unique_ptr	Yes	Automated	Required
A20-8-6	std::make_shared shall be used to construct objects owned by std::shared_ptr	Yes	Automated	Required
A20-8-7	Cyclic Structure of std::shared_ptr	Yes	Non-automated	Required

A21-8-1	Arguments to character-handling functions shall be representable as an unsigned char	Yes	Automated	Required
A23-0-1	An iterator shall not be implicitly converted to const_iterator	Yes	Automated	Required
A23-0-2	Elements of a container shall only be accessed via valid references, iterators, and pointers	No	Automated	Required
A25-1-1	Predicate Function Objects Copied Incorrectly	Yes	Automated	Required
A25-4-1	Ordering predicates used with associative containers and STL sorting and related algorithms shall adhere to a strict weak ordering relation	No	Non-automated	Required
A26-5-1	Pseudorandom numbers shall not be generated using std::rand()	Yes	Automated	Required
A26-5-2	Random number engines shall not be default-initialized	Yes	Automated	Required
A27-0-1	Inputs from independent components shall be validated	Yes	Non-automated	Required
A27-0-2	A C-style string shall guarantee sufficient space for data and the null terminator	No	Automated	Advisory
A27-0-3	Alternate input and output operations on a file stream shall not be used without an intervening flush or positioning call	Yes	Automated	Required
A27-0-4	C-style strings shall not be used	Yes	Automated	Required

CPP_C048	Transferring Control to a Try or Catch Block Using Goto or Switch Statement	Yes	Non-automated	Required
CPP_E030	Concatenating String Literals of Different Encodings	Yes	Automated	Required
M0-1-1	A project shall not contain unreachable code	Yes	Automated	Required
M0-1-2	A project shall not contain infeasible paths	Yes	Automated	Required
M0-1-3	A project shall not contain unused variables	Yes	Automated	Required
M0-1-4	A project shall not contain non-volatile POD variables having only one use.	Yes	Automated	Required
M0-1-8	All functions with void return type shall have external side effect(s)	Yes	Automated	Required
M0-1-9	There shall be no dead code	No	Automated	Required
M0-1-10	Every defined function shall be called at least once.	Yes	Automated	Advisory
M0-2-1	Assigning Object to an Overlapping Object	Yes	Automated	Required
M0-3-1	Minimization of run-time failures shall be ensured by the use of static analysis tools	Yes	Non-automated	Required
M0-3-2	If a function generates error information, then that error information shall be tested	No	Non-automated	Required
M0-4-1	Undocumented Use of Scaled-integer or Fixed-point Arithmetic	Yes	Non-automated	Required
M0-4-2	Undocumented Use of Floating-point Arithmetic	Yes	Non-automated	Required

M1-0-2	Multiple compilers shall only be used if they have a common, defined interface	No	Non-automated	Required
M2-7-1	The character sequence /* shall not be used within a C-style comment.	Yes	Automated	Required
M2-10-1	Different identifiers shall be typographically unambiguous	Yes	Automated	Required
M2-13-2	Octal constants (other than zero) and octal escape sequences (other than "\0") shall not be used.	Yes	Automated	Required
M2-13-3	A "U" suffix shall be applied to all octal or hexadecimal integer literals of unsigned type.	Yes	Automated	Required
M2-13-4	Literal suffixes shall be upper case	Yes	Automated	Required
M3-1-2	Functions shall not be declared at block scope	Yes	Automated	Required
M3-2-1	All declarations of an object or function shall have compatible types	Yes	Automated	Required
M3-2-2	The One Definition Rule	Yes	Automated	Required
M3-2-3	A type, object or function that is used in multiple translation units shall be declared in one and only one file	Yes	Automated	Required
M3-2-4	An identifier with external linkage shall have exactly one definition	Yes	Automated	Required
M3-3-2	If a function has internal linkage then all redeclarations shall include the static storage class specifier	Yes	Automated	Required

M3-4-1	Declarations at Lowest Scope	Yes	Automated	Required
M3-9-1	The types used for an object, a function return type, or a function parameter shall be token-for-token identical in all declarations and re-declarations	Yes	Automated	Required
M3-9-3	The underlying bit representations of floating-point values shall not be used	Yes	Automated	Required
M4-5-1	Expressions with type bool shall not be used as operands to built-in operators other than the assignment operator =, the logical operators &&, ||, !, the equality operators == and !=, the unary & operator, and the conditional operator	Yes	Automated	Required
M4-5-3	Character Operators	Yes	Automated	Required
M4-10-1	NULL shall not be used as an integer value	Yes	Automated	Required
M4-10-2	Literal zero (0) shall not be used as the null-pointer-constant.	Yes	Automated	Required
M5-0-2	Limited dependence should be placed on C++ operator precedence rules in expressions	Yes	Automated	Advisory
M5-0-3	A cvalue expression shall not be implicitly converted to a different underlying type	Yes	Automated	Required
M5-0-4	An implicit integral conversion shall not change the signedness of the underlying type	Yes	Automated	Required
M5-0-5	There shall be no implicit	Yes	Automated	Required

	floating-integral conversions			
M5-0-6	An implicit integral or floating-point conversion shall not reduce the size of the underlying type	Yes	Automated	Required
M5-0-7	There shall be no explicit floating-integral conversions of a cvalue expression	Yes	Automated	Required
M5-0-8	An explicit integral or floating-point conversion shall not increase the size of the underlying type of a cvalue expression	Yes	Automated	Required
M5-0-9	An explicit integral conversion shall not change the signedness of the underlying type of a cvalue expression	Yes	Automated	Required
M5-0-10	If the bitwise operators ~ and << are applied to an operand with an underlying type of unsigned char or unsigned short, the result shall be immediately cast to the underlying type of the operand	Yes	Automated	Required
M5-0-11	The plain char type shall only be used for the storage and use of character values	Yes	Automated	Required
M5-0-12	Signed char and unsigned char type shall only be used for the storage and use of numeric values	Yes	Automated	Required
M5-0-14	The first operand of a conditional-operator	Yes	Automated	Required

	shall have type bool			
M5-0-15	Array indexing over pointer arithmetic	Yes	Automated	Required
M5-0-16	A pointer operand and any pointer resulting from pointer arithmetic using that operand shall both address elements of the same array	Yes	Automated	Required
M5-0-17	Subtraction between pointers shall only be applied to pointers that address elements of the same array	Yes	Automated	Required
M5-0-18	>, >=, <, <= shall not be applied to objects of pointer type, except where they point to the same array	Yes	Automated	Required
M5-0-20	Non-constant operands to a binary bitwise operator shall have the same underlying type	Yes	Automated	Required
M5-0-21	Bitwise operators shall only be applied to operands of unsigned underlying type	Yes	Automated	Required
M5-2-2	A pointer to a virtual base class shall only be cast to a pointer to a derived class by means of dynamic_cast	Yes	Automated	Required
M5-2-3	Casts from a base class to a derived class should not be performed on polymorphic types	Yes	Automated	Advisory
M5-2-6	A cast shall not convert a pointer to a function to any other pointer type, including a pointer to function type	Yes	Automated	Required
M5-2-8	An object with integer	Yes	Automated	Required

	type or pointer to void type shall not be converted to an object with pointer type.			
M5-2-9	Pointer to Integer Cast	Yes	Automated	Required
M5-2-10	The increment (++) and decrement (--) operators shall not be mixed with other operators in an expression	Yes	Automated	Required
M5-2-11	The comma operator, && operator and the || operator shall not be overloaded	Yes	Automated	Required
M5-2-12	Array to Pointer Decay	Yes	Automated	Required
M5-3-1	Each operand of the ! operator, the logical && or the logical || operators shall have type bool	Yes	Automated	Required
M5-3-2	Unary Minus Operator Applied to an Expression with an Unsigned Type	Yes	Automated	Required
M5-3-3	The unary & operator shall not be overloaded	Yes	Automated	Required
M5-3-4	Evaluation of the operand to the sizeof operator shall not contain side effects	Yes	Automated	Required
M5-8-1	The right hand operand of a shift operator shall lie between zero and one less than the width in bits of the underlying type of the left hand operand.	Yes	Partially Automated	Required
M5-14-1	The right hand operand of a logical &&, || operators shall not contain side effects	Yes	Automated	Required
M5-17-1	The semantic equivalence between a binary operator and its	Yes	Non-automated	Required

	assignment operator form shall be preserved			
M5-18-1	The comma operator shall not be used.	Yes	Automated	Required
M5-19-1	Evaluation of constant unsigned integer expressions shall not lead to wrap-around	No	Automated	Required
M6-2-1	Assignment operators shall not be used in sub-expressions	Yes	Automated	Required
M6-2-2	Floating-point expressions shall not be directly or indirectly tested for equality or inequality	Yes	Partially Automated	Required
M6-2-3	Before preprocessing, a null statement shall only occur on a line by itself; it may be followed by a comment, provided that the first character following the null statement is a white-space character	Yes	Automated	Required
M6-3-1	The statement forming the body of a switch, while, do ... while or for statement shall be a compound statement	Yes	Automated	Required
M6-4-1	An if (condition) construct shall be followed by a compound statement. The else keyword shall be followed by either a compound statement, or another if statement	Yes	Automated	Required
M6-4-2	All if and else if constructs shall be terminated with an else clause	Yes	Automated	Required

M6-4-3	A switch statement shall be a well-formed switch statement	Yes	Automated	Required
M6-4-4	A switch-label shall only be used when the most closely-enclosing compound statement is the body of a switch statement	Yes	Automated	Required
M6-4-5	An unconditional throw or break statement shall terminate every non-empty switch-clause	Yes	Automated	Required
M6-4-6	The final clause of a switch statement shall be the default-clause	Yes	Automated	Required
M6-4-7	The condition of a switch statement shall not have bool type	Yes	Automated	Required
M6-5-2	If loop-counter is not modified by -- or ++, then, within condition, the loop-counter shall only be used as an operand to <=, <, > or >=	Yes	Automated	Required
M6-5-3	The loop-counter shall not be modified within condition or statement	Yes	Automated	Required
M6-5-4	The loop-counter shall be modified by one of: --, ++, -= n, or += n; where n remains constant for the duration of the loop	Yes	Automated	Required
M6-5-5	A loop-control-variable other than the loop-counter shall not be modified within condition or expression	Yes	Automated	Required
M6-5-6	A loop-control-variable other than the loop-counter which is	Yes	Automated	Required

	modified in statement shall have type bool			
M6-6-1	Any label referenced by a goto statement shall be declared in the same block, or in a block enclosing the goto statement	Yes	Automated	Required
M6-6-2	The goto statement shall jump to a label declared later in the same function body	Yes	Automated	Required
M6-6-3	Continue Statement Used in a not Well-formed For Loop	Yes	Automated	Required
M7-1-2	A pointer or reference parameter in a function shall be declared as pointer to const or reference to const if the corresponding object is not modified	Yes	Automated	Required
M7-3-1	Global Namespace Declarations	Yes	Automated	Required
M7-3-2	The identifier main shall not be used for a function other than the global function main	Yes	Automated	Required
M7-3-3	There shall be no unnamed namespaces in header files.	Yes	Automated	Required
M7-3-4	Using-directives shall not be used.	Yes	Automated	Required
M7-3-6	using-directives and using-declarations (excluding class scope or function scope using-declarations) shall not be used in header files.	Yes	Automated	Required
M7-4-1	Assembly Language Code Usage not Documented	Yes	Non-automated	Required

M7-4-2	Assembler instructions shall only be introduced using the asm declaration.	Yes	Automated	Required
M7-4-3	Assembly language shall be encapsulated and isolated.	Yes	Automated	Required
M7-5-1	A function shall not return a reference or a pointer to an automatic variable (including parameters), defined within the function.	Yes	Non-automated	Required
M7-5-2	The address of an object with automatic storage shall not be assigned to another object that may persist after the first object has ceased to exist.	Yes	Non-automated	Required
M8-0-1	Single Declarations	Yes	Automated	Required
M8-3-1	Parameters in an overriding virtual function shall either use the same default arguments as the function they override, or else shall not specify any default arguments.	Yes	Automated	Required
M8-4-2	The identifiers used for the parameters in a re-declaration of a function shall be identical to those in the declaration.	Yes	Automated	Required
M8-4-4	A function identifier shall either be used to call the function or it shall be preceded by &.	Yes	Automated	Required
M8-5-2	Incorrect Initializer Lists	Yes	Automated	Required
M9-3-1	Const Member Function Returning Non-Const Pointer or Reference	Yes	Automated	Required

M9-3-3	If a member function can be made static then it shall be made static, otherwise if it can be made const then it shall be made const.	Yes	Automated	Required
M9-6-1	When the absolute positioning of bits representing a bit-field is required, then the behavior and packing of bit-fields shall be documented	No	Non-automated	Required
M9-6-4	Bit-field Length	Yes	Automated	Required
M10-1-1	Class Derived From Virtual Bases	Yes	Automated	Advisory
M10-1-2	A base class shall only be declared virtual if it is used in a diamond hierarchy	Yes	Automated	Required
M10-1-3	An accessible base class shall not be both virtual and non-virtual in the same hierarchy	Yes	Automated	Required
M10-2-1	Similiar Entity Names within Multiple Inheritance	Yes	Automated	Advisory
M10-3-3	A virtual function shall only be overridden by a pure virtual function if it is itself declared as pure virtual	Yes	Automated	Required
M11-0-1	Member Data in Non-POD Class not Private	Yes	Automated	Required
M12-1-1	An object's dynamic type shall not be used from the body of its constructor or destructor	Yes	Automated	Required
M14-5-3	A copy assignment operator shall be declared when there is a template assignment	Yes	Automated	Required

	operator with a parameter that is a generic parameter			
M14-6-1	In a class template with a dependent base, any name that may be found in that dependent base shall be referred to using a qualified-id or this->	Yes	Automated	Required
M15-0-3	Control shall not be transferred into a try or catch block using a goto or a switch statement	Yes	Automated	Required
M15-1-1	Exception Object	Yes	Automated	Required
M15-1-2	NULL Throw	Yes	Automated	Required
M15-1-3	Empty Throw	Yes	Automated	Required
M15-3-1	Exceptions shall be raised only after start-up and before termination of the program	Yes	Automated	Required
M15-3-3	Handlers of a function-try-block implementation of a class constructor or destructor shall not reference non-static members from this class or its bases	Yes	Automated	Required
M15-3-4	Each exception explicitly thrown in the code shall have a handler of a compatible type in all call paths that could lead to that point	Yes	Automated	Required
M15-3-6	Order of Catch Blocks with Derived Classes	Yes	Automated	Required
M15-3-7	Where multiple handlers are provided in a single try-catch statement or function-try-block, any ellipsis (catch-all) handler shall occur last	Yes	Automated	Required
M16-0-1	#include Directives Not	Yes	Automated	Required

	Grouped Together			
M16-0-2	Macros shall only be #define'd or #undef'd in the global namespace.	Yes	Automated	Required
M16-0-5	Function-like Macro Containing Preprocessing Directives	Yes	Automated	Required
M16-0-6	In the definition of a function-like macro, each instance of a parameter shall be enclosed in parentheses, unless it is used as the operand of # or ##	Yes	Automated	Required
M16-0-7	Undefined macro identifiers shall not be used in #if or #elif preprocessor directives, except as operands to the defined operator	Yes	Automated	Required
M16-0-8	Invalid Preprocessor Directives	Yes	Automated	Required
M16-1-1	The defined preprocessor operator shall only be used in one of the two standard forms	Yes	Automated	Required
M16-1-2	All #else, #elif and #endif preprocessor directives shall reside in the same file as the #if, #ifdef or #ifndef directive to which they are related	Yes	Automated	Required
M16-2-3	Include guards shall be provided	Yes	Automated	Required
M16-3-1	There shall be at most one occurrence of the # or ## operators in a single macro definition	Yes	Automated	Required
M16-3-2	The # and ## operators should not be used	Yes	Automated	Advisory
M17-0-2	The names of standard	Yes	Automated	Required

	library macros and objects shall not be reused			
M17-0-3	Standard Library Function Names	Yes	Automated	Required
M17-0-5	The setjmp macro and the longjmp function shall not be used	Yes	Automated	Required
M18-0-3	<cstdlib> Library Functions	Yes	Automated	Required
M18-0-4	Time Handling Functions of <ctime>	Yes	Automated	Required
M18-0-5	Unbounded Functions of <cstring>	Yes	Automated	Required
M18-2-1	The macro offsetof shall not be used	Yes	Automated	Required
M18-7-1	The signal handling facilities of <csignal> shall not be used	Yes	Automated	Required
M19-3-1	The error indicator errno shall not be used	Yes	Automated	Required
M27-0-1	The stream input/output library <cstdio> shall not be used	Yes	Automated	Required