SAFE RIVER

Safety & Security Forge

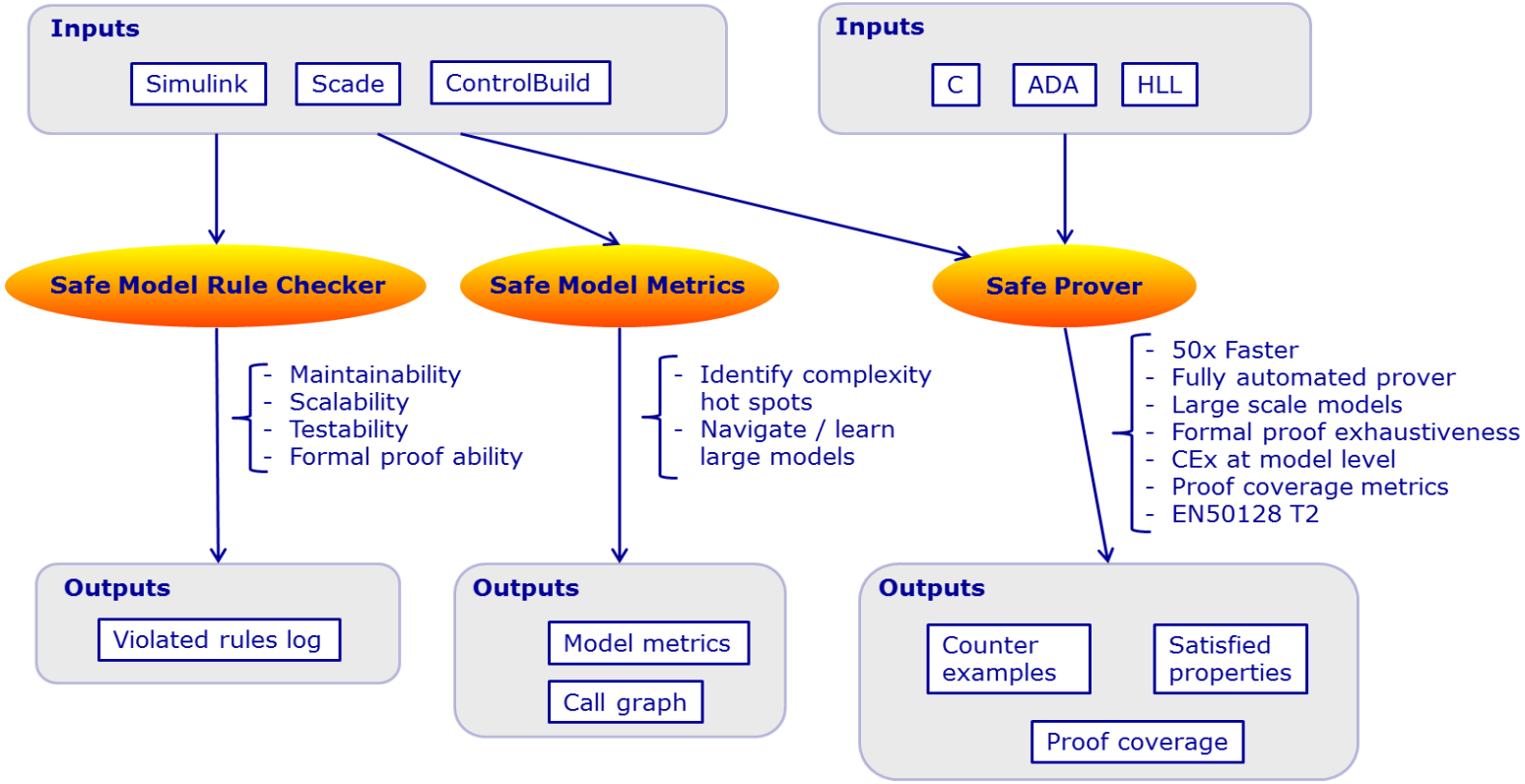
SAFE RIVER
Safety & Security Forge

SafeRiver

- **SME**
  - Independent- founded december 2005
  - 18 consultants highly skilled in Software and Formal methods
  - Turnover 2015: 1,5M€ (excluding R&D public fundings)
- **Added Value Solutions for Embedded Systems**
  - Functional Safety (FuSa)
  - Software Security
- **Tools for FuSa and Software Security**
- **Packaged Services**
- **CIR agreed**

SAFE RIVER
Safety & Security Forge

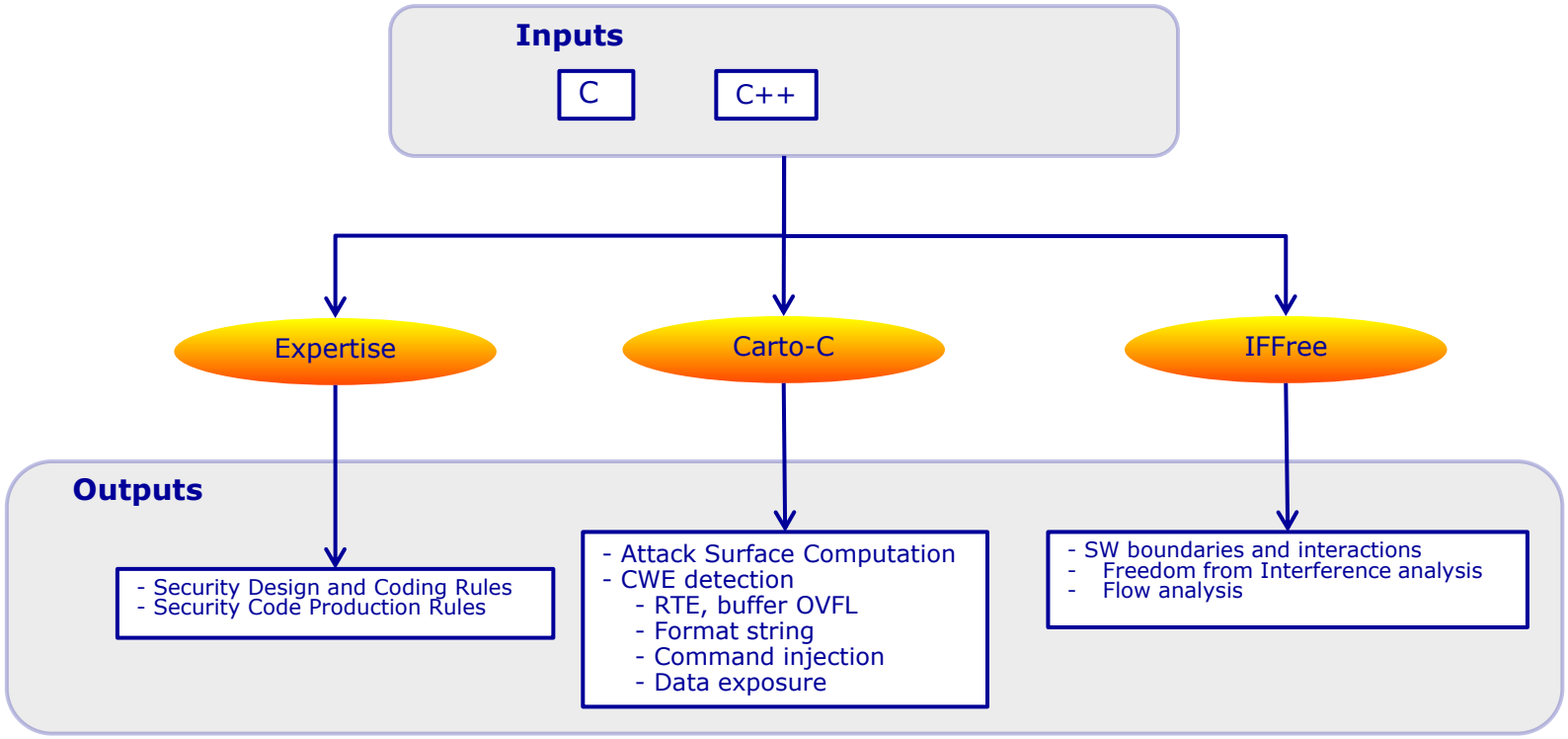
Functional Safety



- **Modeling rules**
- **Models metrology (technical debt, change management)**
- **Formal Proof of Functional safety requirements and System level safety properties –**
- **Model- Code Equivalence**

SAFE RIVER
Safety & Security Forge

Software Security



- **Carto-C supports Vulnerability Analysis (SVA) – Benchmarked on Juliet Database**
- **IFFree addresses Software architecture analysis with respect to trust/integrity domains both for safety and security. Supports FFI analysis and helps in interfaces mastering- ISO 26262-6**

SAFE RIVER
Safety & Security Forge

Static Analysis for SW Security: key issues

- **150 to 200 tools**
- **What does mean « Perform a static analysis » ?**
  - Tools classification /underlying techniques
    - Sound
    - Unsound
  - Verification objectives
    - Rules Verification/Detection of Coding Rules violations
      - DSIG Cert-C CERT-Java
    - Detection of well known vulnerabilities
      - NIST, CVE, CWE
    - Run Time Errors detection
  - Level of assurance and errors coverage
    - Public reference
    - Evaluation

SAFE RIVER
Safety & Security Forge

Static tools for security: Observations

- **Results from sound tools**
  - Fit a small subset of security flaws
  - Are subject to false positive
  - But are not subject to false negatives
  - Do not take the security environment into account
- **Results from unsound tools**
  - Fit a large subset of security flaws
  - Are subject to false positive
  - Are subject to false negative
  - Do take the security environment into account

Sound	**RTE (subset of CWE)**	**False positive**
Unsound	**CWE, CVE, CAPEC or** **CERT C, CERT java, JavaSec, DISA STIG**	**False positive and False negative**

- ⇒ **Adequate tool is difficult to choose and use**

SAFE RIVER
Safety & Security Forge

Static Analysis for Security : Configuration kits

- **Detection objectives**
  - Eliminate most current vulnerabilities as defined by
    - SANS Top25
    - OWASP Top 10
- **Configuration kit content**
  - Sets of checkers to be activated
  - Detection parameters
  - Definition of criticality levels
  - Result filters and synthesis
- **Available kits**
  - Klocwork for Java or C: 69 checkers for 22 CWE
  - Coverity for java or C: 30 checkers for 20 CWE

SAFE RIVER
Safety & Security Forge

Static Analysis for security: Evaluation kit

- **Juliet**
  - Is developed in SAMATE SATE project to challenge static tools
  - Is composed of ~45000 C codes
  - Analyzable in « flaw » and « fix » mode
    - Flaw: the source code contains a flaw
    - Fix: the source code contains a fix of the flaw
  - Covering more than 121 main classes of CWE flaws
- **Juliet User kit by SafeRiver**
  - Libraries Support
    - libC, POSIX
  - Automatic launch
  - Automatic synthesis

SAFE RIVER
Safety & Security Forge

Static Analysis for Security: Carto-C

- **Why Carto-C ?**
  - Use cases
    - Support of Secure Development
    - Support of Security audits
  - Only sound static tool to detect
    - Missing input filtering
      - Impact on known flaws
    - Missing asset protections
      - Impact on asset exposure
- **Evaluation with Juliet Test base**
  - Internationally recognized tests base
  - Independent test base

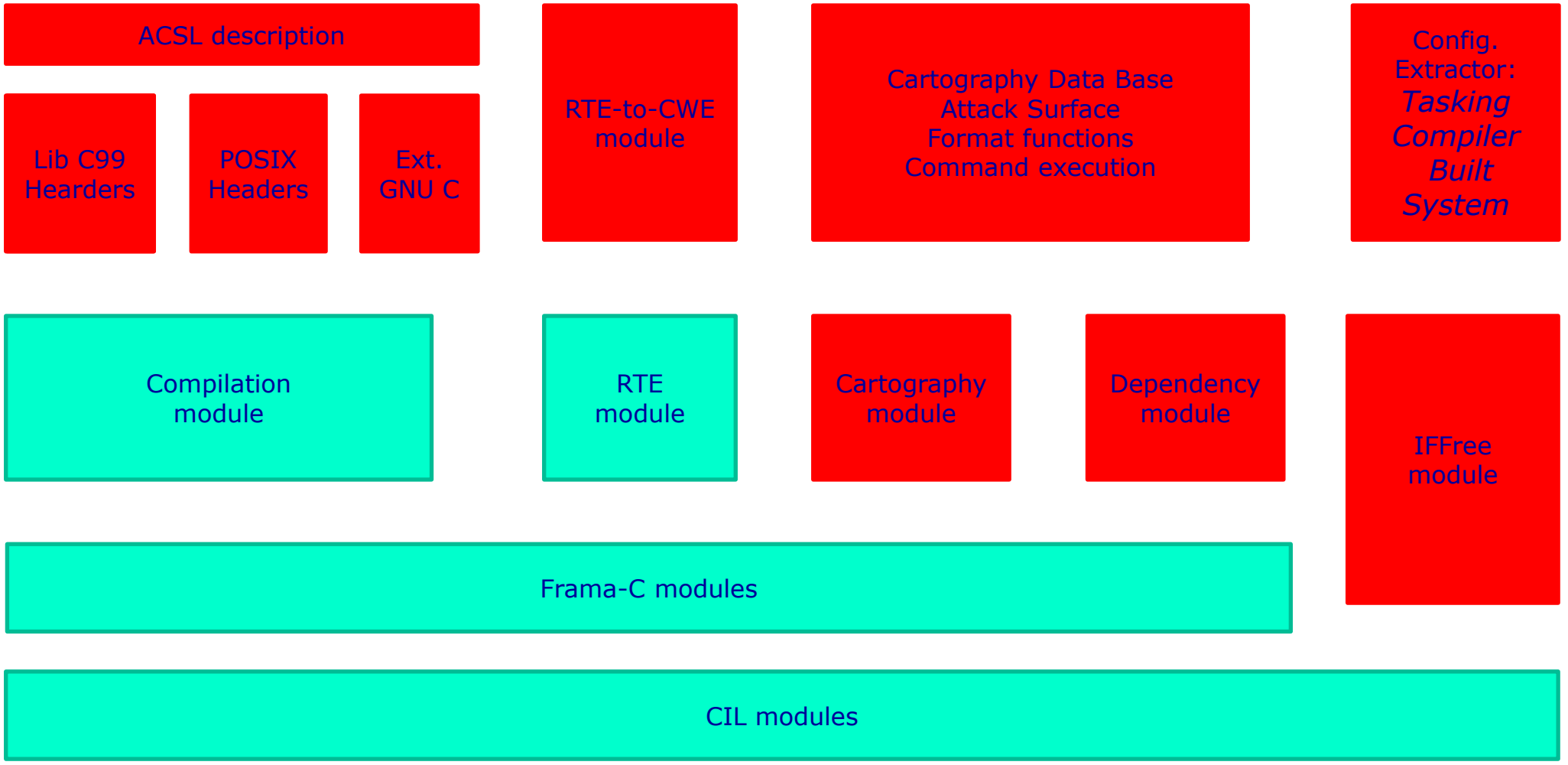
SAFE RIVER
Safety & Security Forge

Carto-C

- **Carto-C is a static Analyzer based on the open source platform Frama-C, that we have specialized for Security**
  - Attack Surface Computation
  - Format String and Injection Related Weaknesses Detection
  - Risk analysis support
    - Identification of assets that can be reached/controlled by malevolent actions through attack surface
    - Verification of protections
  - Freedom from Interference Analysis
    - Characterization of cascading failures that can be caused by uncontrolled or malevolent interactions
    - Use cases : document interactions between software that have different integrity or assurance levels

SAFE RIVER
Safety & Security Forge

Carto-C architecture



SAFE RIVER
Safety & Security Forge

Added Value analysis

- **Frama-C modules**
  - Static analysis algorithms
  - RTE detection
  - asserts
- **Carto-C Proprietary modules for end user generic needs**
  - Usability for complete applications
    - Stubs (ACSL description)
  - False positive reduction
- **Carto-C Proprietary modules for customer needs**
  - Attack Surface
  - Detection of weaknesses according to CWE model
  - Freedom From Interference analysis

SAFE RIVER
Safety & Security Forge

Carto-C Feature 1
Identify Attack surface

- **Attack Ways**
  - All the entry points and exit points methods
  - The set of input / output channels
  - The set of input / output data
  - All the calls to external code (third party tool, open source)
- **Protection functions**
  - Resource connection and authentication
  - Authorization
  - Data validation and encoding
  - Events logging
- **User defined declarations**
  - I/O functions
  - Protection functions
  - Trusted channels

SAFE RIVER
Safety & Security Forge

Carto-C Feature 1
Identify Attack surface

- **Attack objectives**
  - Assets of the application
    - confidential, sensitive, regulated data
    - secrets and keys, intellectual property, critical business data, personal data and PI
    - (user defined)
- **Protection functions**
  - Encryption, digest
  - access and authorization
  - data integrity and operational security controls
- **User defined declaration**
  - Valuable data
  - Protection functions

SAFE RIVER
Safety & Security Forge

Feature 2
Detect exhaustively certain classes of flaws

- **Extracted from Frama-C RTE**
  - CWE119: Improper Restriction of Operations within the Bounds of a Memory Buffer
    - CWE787
      - CWE121_Stack_Based_Buffer_Overflow
      - CWE122_Heap_Based_Buffer_Overflow
      - CWE124_Buffer_Underwrite
    - CWE125
      - CWE126_Buffer_Overread
      - CWE127_Buffer_Underread
  - CWE664 : Improper Control of resources through lifetime
    - CWE401_Improper release of memory before removing last reference
    - CWE457_use of uninitialized variable
    - CWE665_Improper Initialization
  - CWE682: Incorrect Calculation
    - CWE190: Integer Overflow or Wraparound
    - CWE191: Integer Underflow or Wraparound
    - CWE369: Divide_by_Zero
    - CWE681: Incorrect Conversion between Numeric Types

SAFE RIVER
Safety & Security Forge

Feature 2
Detect exhaustively certain classes of flaws

- **Carto-C specific Plug ins/modules**
  - Cartography
    - **CWE-749: Exposed Dangerous Method or Function (format and command execution function)**
  - Syntactic Verification
    - **CWE-628: Function Call with Incorrectly Specified Arguments**
      - CWE685_Function_Call_With_Incorrect_Number_of_Arguments
      - CWE686:Function with Incorrect Argument Type
      - CWE688_Function_Call_With_Incorrect_Variable_or_Reference_as_Argument
  - Dependency analysis
    - **CWE-134: Uncontrolled Format String)**
    - **CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')**

SAFE RIVER
Safety & Security Forge

Carto C results on Juliet benchmark

CWE Entry ID	CWE Entry Name	Flaw tes	Flaw detect Rate	Fix test cas	Fix detect Rate
78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	40	100%	40	100%
134	Uncontrolled Format String	30	100%	60	100%
191	Integer Underflow (Wrap or Wraparound)	29	79%	66	74%
190	Integer Overflow or Wraparound	48	75%	108	70%
681	Incorrect Conversion between Numeric Types	3	67%	3	33%
369	Divide By Zero	16	63%	36	78%
126	Buffer Over-read	23	39%	30	83%
124	Buffer Underwrite (‘Buffer Underflow’)	19	32%	32	97%
127	Buffer Under-read	21	29%	32	97%
122	Heap-based Buffer Overflow	66	21%	75	75%
121	Stack-based Buffer Overflow	48	19%	68	93%

- **Carto-C specific Plug ins/modules -> detection rate 100%**
- **Extracted Results (underflw and overflow, buffer errors) surprising -> open point under investigation**

SAFE RIVER
Safety & Security Forge

Feature 3
Exploitation of flaws

- **Controllable from the attack surface entry points**

  Example: command read from the keyboard is highly dangerous

  Controllability : high / low / unknown

- **Observable from the attack surface exit points**
  - Example: password printed in a log
  - Observability: high / low / unknown

SAFE RIVER
Safety & Security Forge

RTE 2 CWE

- **Problematic**
  - Formal Backend analyzers detect errors that have an unambiguous specification
  - Some analyzers detect errors wrt
    - Patterns
    - Rules
  - CWE model is an enumeration, not a clear classification tool
- **RTE2CWE module**
  - Maps RTE detected by Frama-C in terms of CWE flaws
  - Helps for benchmarking and comparison of tools

SAFE RIVER
Safety & Security Forge

Open Source Model Applicability

- **Strengths**
  - Recognized static analyzers
  - Public Static Analyzers may be evaluated
  - Hard problems to be addressed by the community
- **Weaknesses**
  - Usage restrictions of formal static analysis methods
    - Language restrictions
    - Requires semantic specification at language level
  - Lack of interest or lack of cooperation for evaluation and benchmarks
  - Security analysis do not match directly to static analysis results
    - Many customer data to be taken into account

SAFE RIVER
Safety & Security Forge

Open Source Model Applicability

- **Opportunities for cooperation**
  - Development of static analysis market
  - Languages to be covered
    - C++
    - Java
    - Script
  - Relaxed analysis methods
  - Confidence level requirements
    - Benchmarks/labels
- **Threats**
  - Stubs and Libraries are necessary but user does not want to pay for
  - Same for « false positive » reduction
  - Competition among many SMEs

SAFE RIVER
Safety & Security Forge

Cyber-security References

- **ANSSI (on going study)**
  - Development of a Referential for Static Analysis Tools labellisation with respect to detection of cyber security flaws
- **THALES Communication & Security & DGA**
  - Study and business case on formal methods for cryptographic modules development, creation of a prototype
  - Development of a certified XML parser, Assurance level **EAL4+**
- **ANSSI**
  - Study (state of the art, security analysis) of the functional **La**nguages **Fo**r **Sec**ure applications (LaFoSec), Development Guide for Ocaml language and tools usage
  - Development of an XML parser, with proof of robustness (vulnerabilities detection and analysis)

SAFE RIVER
Safety & Security Forge

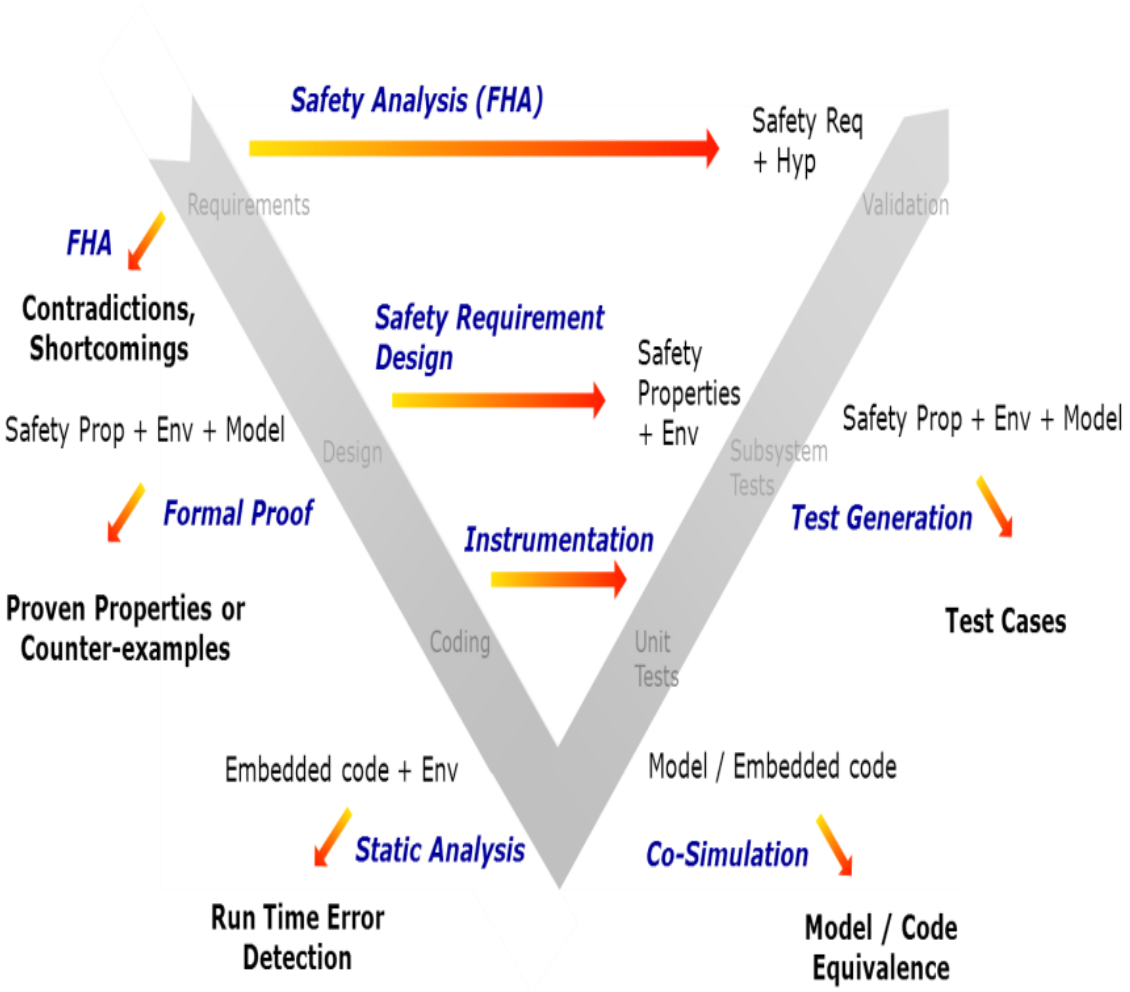
Cyber-security References

- **DGA (CORAC) 2013-2014-2015**
  - Cyber-security methods and tools for the future avionic platform
- **SAFRAN (2013-2014)**
  - Code Analysis Tools Configuration (Klocwork) for CWE detection
  - Development of a document of the functional security requirements for a flight recorder, using the EBIOS method
- **Airbus (2008-2010)**
  - Risks Analysis on the safety and the security of the information system embedded on the Aircraft
  - Security Guidelines for the information system components suppliers (coding rules, COTS evaluation guidelines, vulnerabilities analysis

SAFE RIVER
Safety & Security Forge

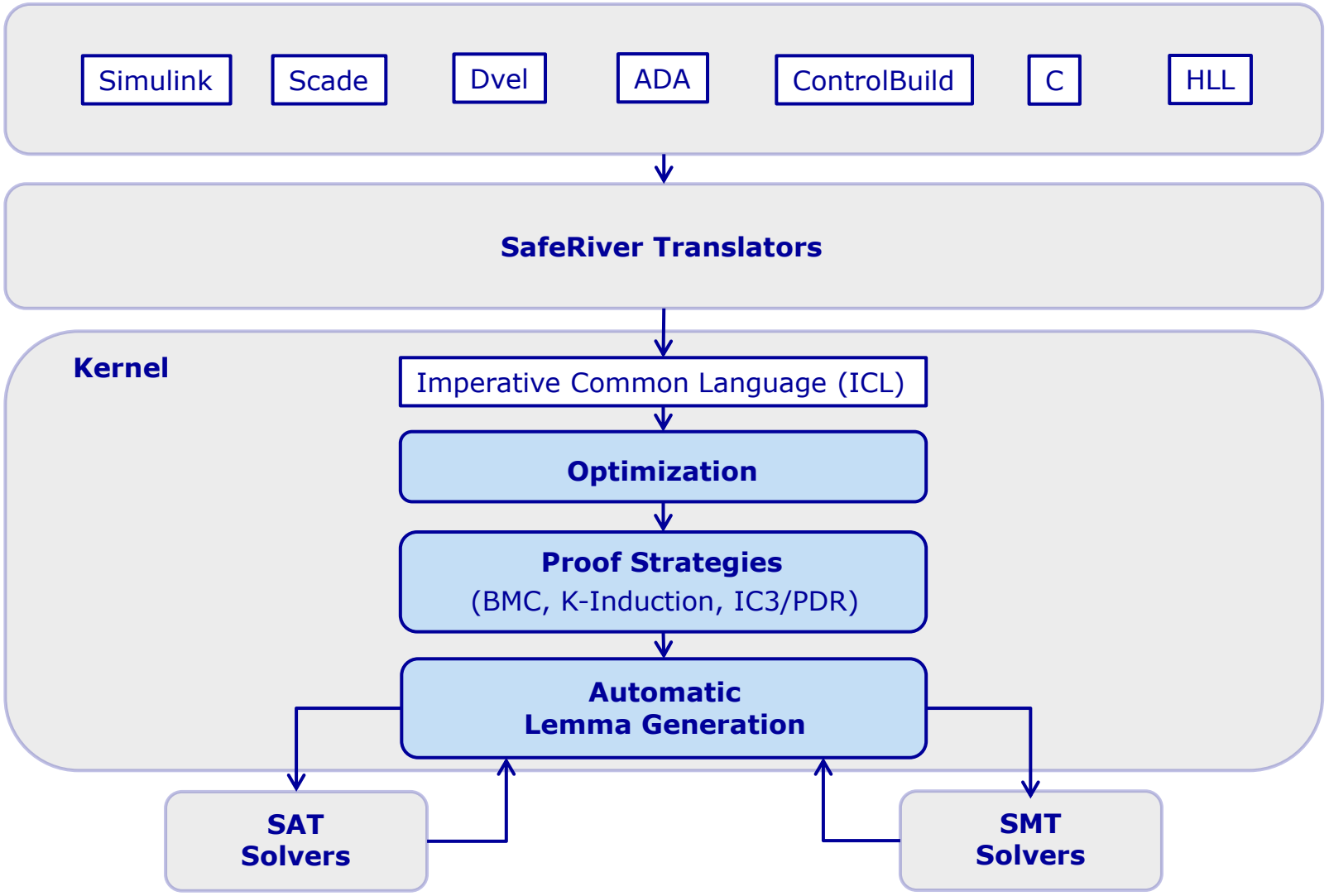
Formal Verification

- Use case : Verification of Design with respect to functional safety requirements in an MBD process



SAFE RIVER
Safety & Security Forge

SafeProver tool chain



SAFE RIVER
Safety & Security Forge

User Accessibility

- **SMT solvers are very powerful**
  - Deduction part
  - Can solve problems with many many variables
- **But users do not interact with low level formulas**
  - Representation of models and formulas in high level formalisms (e.g. Simulink, SCADE, Control Build etc..)
  - Translation from High level formalisms to intermediate format
  - High level optimizations -> help in proof convergence
  - Proof strategy -> compliance with expected results, soundness
  - Automatic Lemma Generation « the hard problem »

Kernel is the key for user accessibility

- **Kernel components importance**
  - Guarantee soundness of deduction parts
  - Ensure convergence and mastering of scalability issues
  - Are responsible of taking benefits of backend solvers
  - Not only results of the verification are provided but many other information:
    - Counterexamples, traced back to the high level model
    - Proof coverage
    - Proof log or evidence that the whole process is auditable
- **Kernel development is a very expensive effort**
  - Needs for verification and certification of the kernel imply a very rigourous documentation and development process, quite centralized rather than cooperative
- **Kernel is a very important asset for expertise and service**

SAFE RIVER
Safety & Security Forge

Confidence in formal Verification

- **Low level engines and solvers**
  - May be diversified
  - Some benchmarks exist
- **Verification kernel is safety critical**
  - Failure or even false positive may be caused by translation and optimization errors
  - Verified checkers are required (strategy implementation)
- **Needs for « proven » or « verified » kernel**
  - Impacts the applicability of Open source model

SAFE RIVER
Safety & Security Forge

Service level Metrology

- **Time to make the model able to be analyzed/proved**
  - Compatibility (syntactic)
  - Convergence
  - Iterations with the model owner
- **Time to implement and execute one proof**
  - Memory/execution time consumption
  - Scalability
- **Time to analyze counter-examples**
- **Proof Coverage**

SAFE RIVER
Safety & Security Forge

Return from Experience

- **Intensive use on large CBTC models (railway domain)**
  - Complex Functions modelled in Matlab/Simulink
    - Localization,
    - Train Tracking
    - Evacuation
    - Passenger exchange
    - Etc.
  - Time for Compatibility and Convergence :
    - 40 days about for the whole model
  - More than 700 properties
    - Functional safety requirements traceable with the design requirements level
  - Average Cycle time for one property
    - 2.5 days the first issue
    - 1.5 days for rework.

SAFE RIVER
Safety & Security Forge

Open Source Model Applicability

- **Opportunities**
  - Cooperation on backend solvers
    - Parametrization
    - Distributed models
    - Etc.
  - Strategies development
  - Many contributors at academic level (SAT, SMT, Proof assistants etc.)
- **Threats**
  - COTS editors are more aggressive than they were on the topic
    - E.g. The Mathworks
  - Certification -> Kernel shall be evaluated
    - Development cost
    - Changes and evolutions are more difficult to manage

SAFE RIVER
Safety & Security Forge

Conclusion

- **Formal Methods**
  - Many backend solvers are being developed
  - User accessibility bottlenecks still the same
    - Scalability
    - Abstraction level
    - Controllability of results
      - False positive
      - Proof coverage
- **Cooperation between actors**
  - Academic and Expertise companies
  - TRL assignment depending on actors