FROM RESEARCH TO INDUSTRY

cea tech

frama C

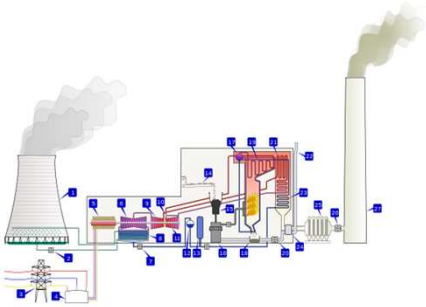
Software Analyzers

LEVERAGING STATIC ANALYSIS IN INDUSTRIAL CONTEXTS

list

THE SEMANTICS OF SOFTWARE VERIFICATION

- DO-178B/C for aerospace safety
- EN 50128 for rail safety
- ISO 26262 for road vehicles
- CENELEC 61508
- Common Criteria EAL7 for security

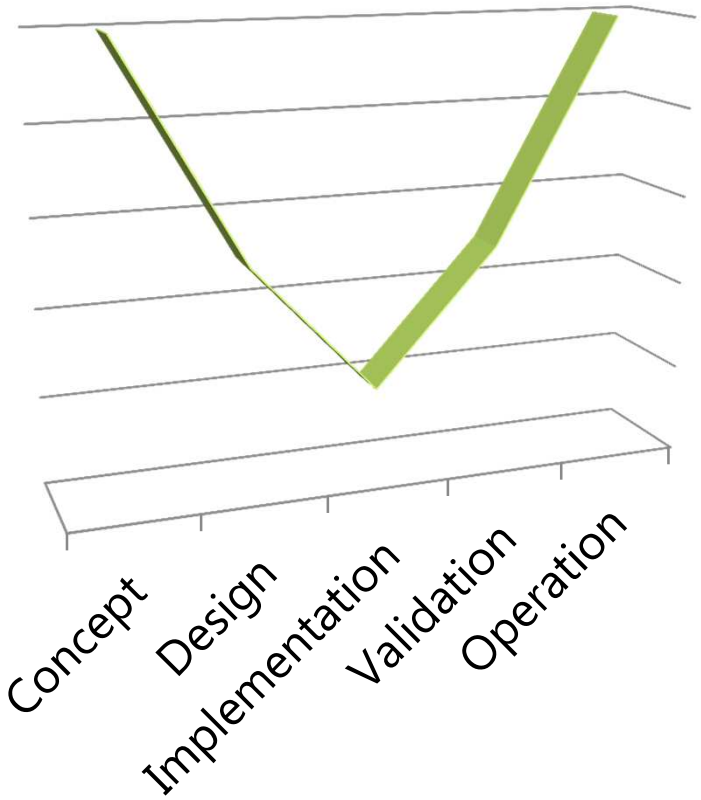


26262 ISO

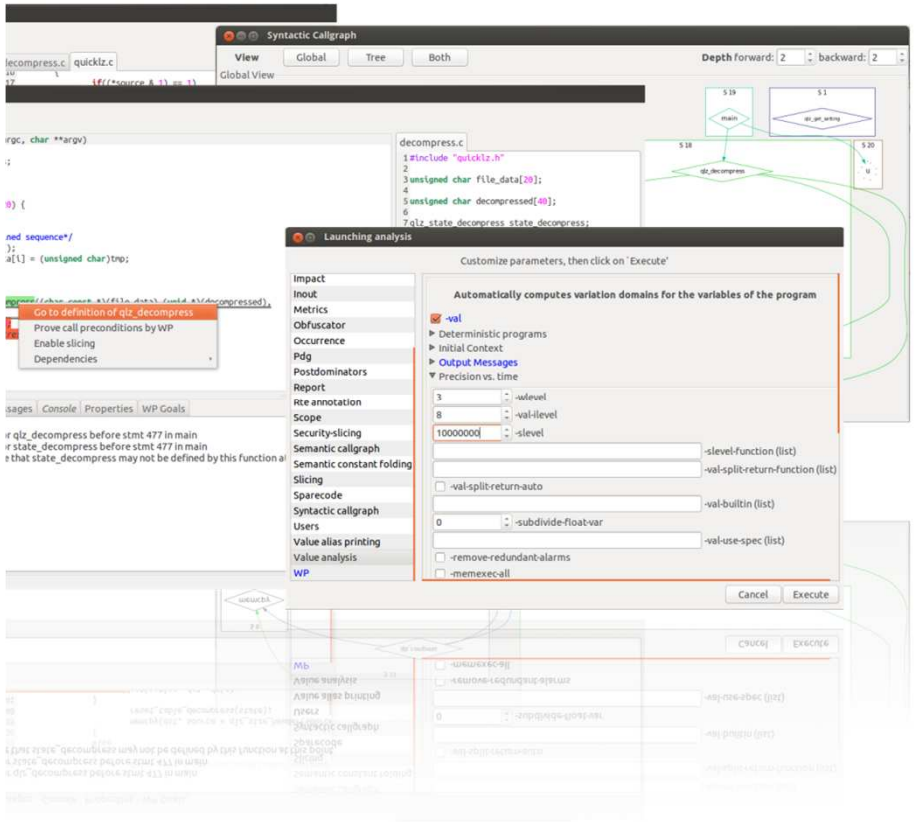
Common Criteria

APPLYING SOFTWARE VERIFICATION

- **Simple “press button” rule enforcement**
- **Partial runtime error or security flaw detection**
- **Complete validation against a given specification**

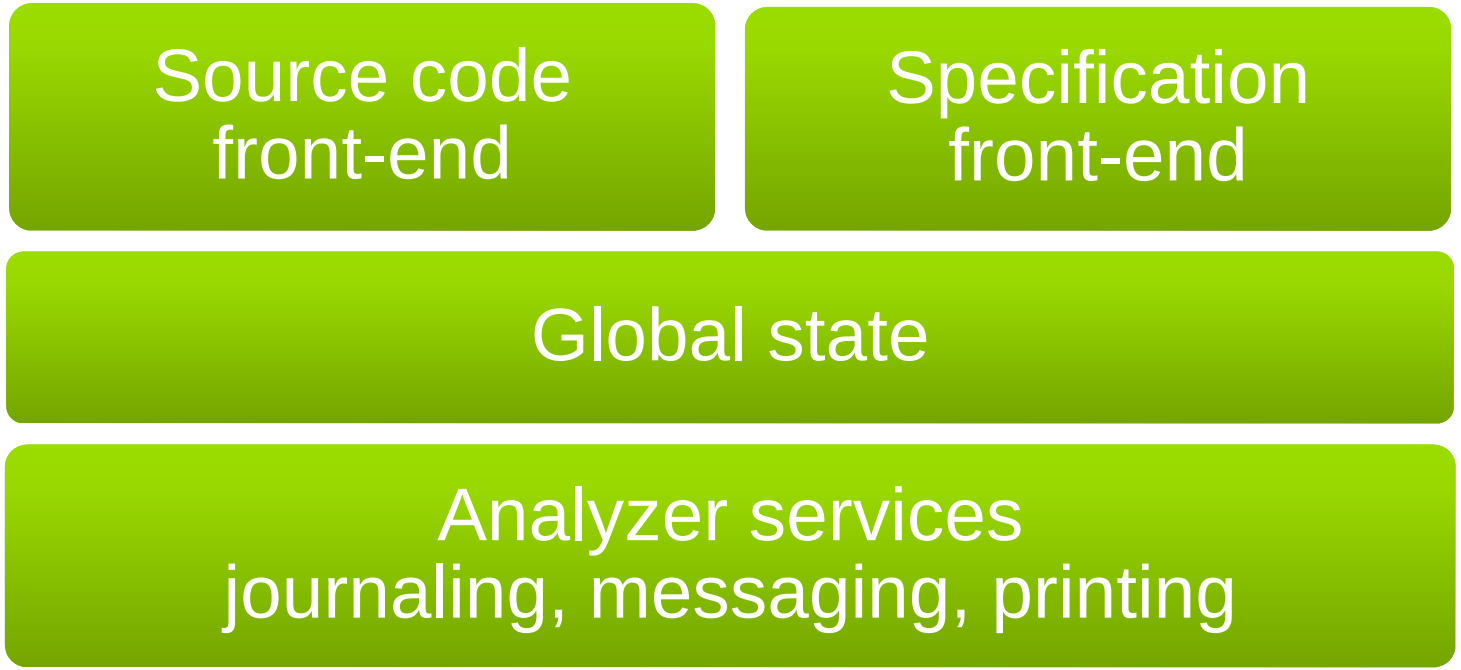


A SOURCE CODE ANALYSIS PLATFORM



- Extensible via plug-ins
- Collaborative
- Designed at CEA & Inria
- Semantic analyses
- ANSI / ISO C 99

THE FRAMA-C KERNEL



LINGUA FRANCA: THE ACSL SPECIFICATION LANGUAGE

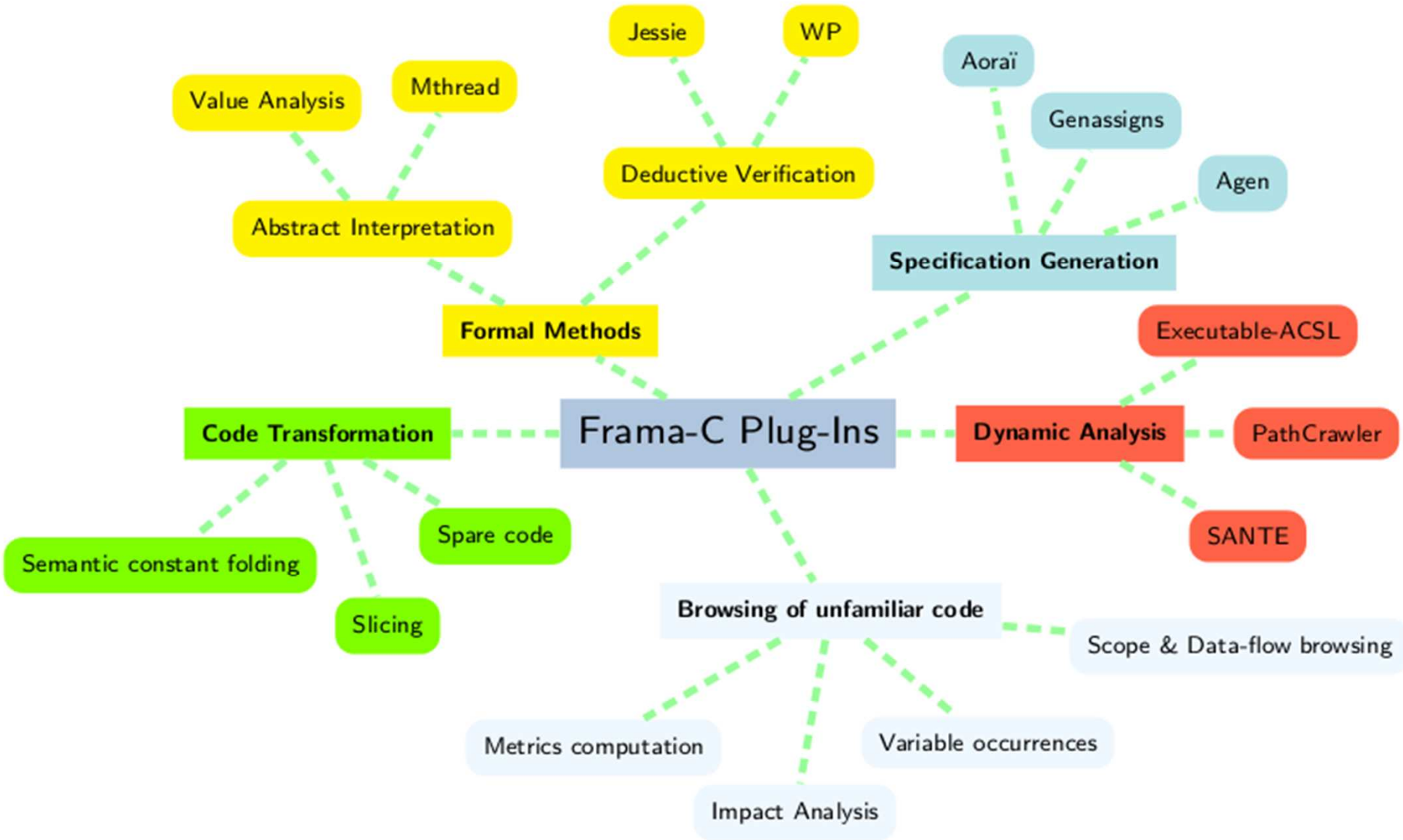
Specification front-end

l state

- Contract-based
- Functional properties
- User-friendly
- Analysis-independent

```
unsigned int M ;
/*@
requires \valid (p) && \valid (q);
ensures M == (*p + *q) / 2;
*/
void mean (  unsigned int* p,
             unsigned int* q ) {
if (* p >= * q )
  M = (* p - * q ) / 2 + * q ;
else
  M = (* q - * p ) / 2 + * p ;
}
```

- Caller-callee contract
- Callee **requires** some pre-conditions from the caller
- Callee **ensures** some post-conditions hold when it returns


Jessie
WP
Aoraï
Value Analysis
Mthread
Genassigns
Deductive Verification
Agen
Abstract Interpretation
Specification Generation
Formal Methods
Executable-ACSL
Code Transformation
Frama-C Plug-Ins
Dynamic Analysis
PathCrawler
Spare code
SANTE
Semantic constant folding
Slicing
Browsing of unfamiliar code
Scope & Data-flow browsing
Metrics computation
Variable occurrences
Impact Analysis

ENFORCING CODING STANDARDS WITH FRAMA-C

- **Result** **Airbus and Atos have designed the Taster plugin on top of Frama-C to enforce coding standards.**

- **Conclusion** **Frama-C yields productivity gains and ensures code quality.**

Benefits :

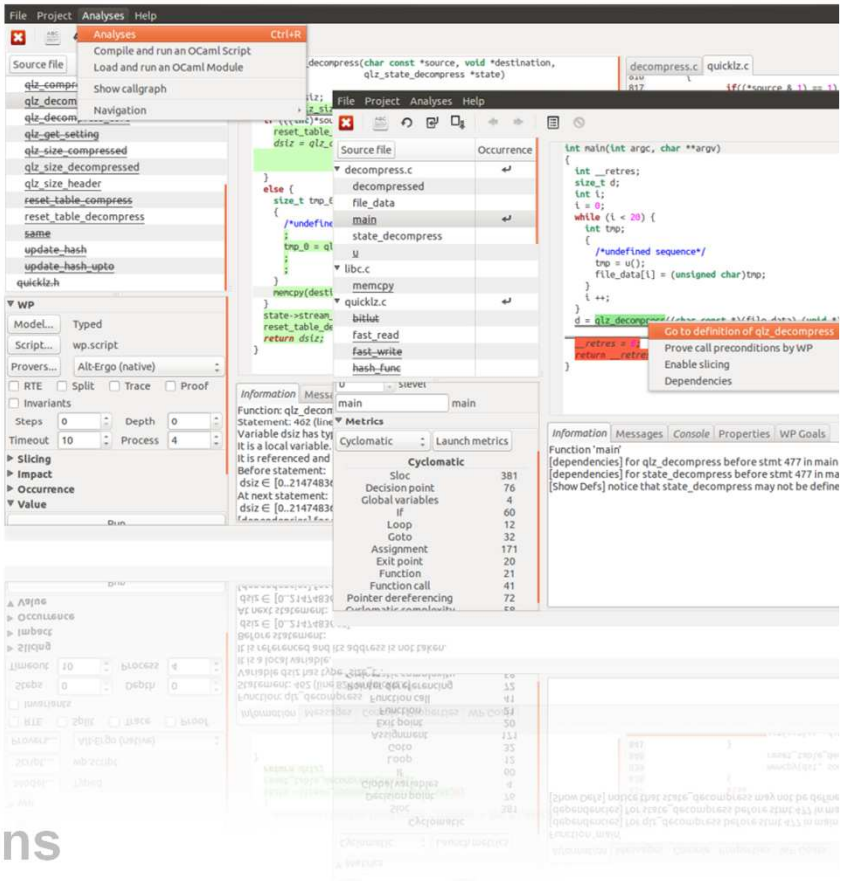
✓ Eases code review on syntactic or typing rules.

✓ Validation of semantic rules:

- dataflow related rules on variables,
- runtime errors requiring a value analysis.

SEMANTICAL ANALYSIS

- Automated process
- Integral & pointer ranges
- ACSL verifications
- Runtime-errors threats
- Side-effects & dependency analysis
- Program structure & transformations
- Confidentiality or integrity leaks



CHECKING INTRINSIC FAULTS IN SCADA SYSTEMS

- **Result** Researchers have demonstrated the absence of multiple fault families in safety-critical software.
- **In addition** derived analyses cover structural properties on memory separation and cyclic behaviors.
- **Conclusion** Frama-C enables highly-automated verification runs.

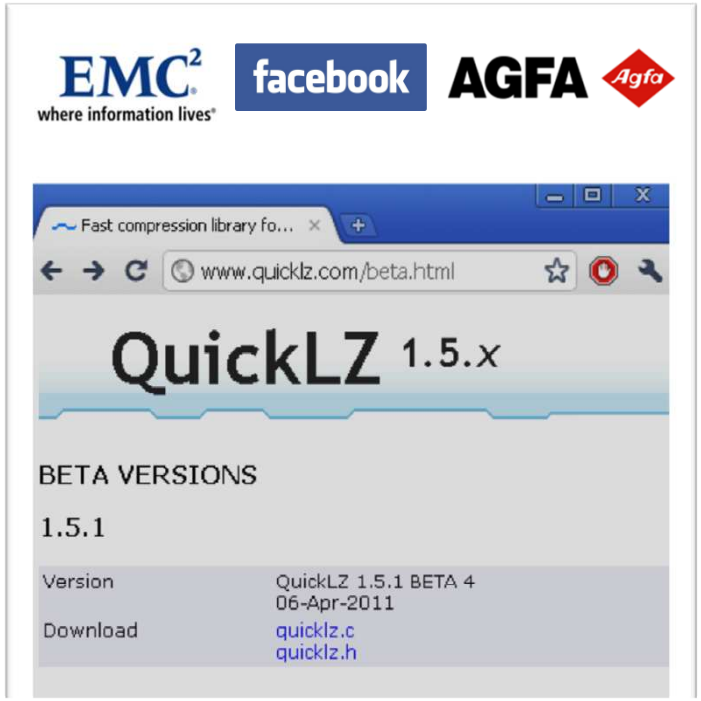
```
> 100+ kloc
> C source code
> Highest
      certification
      requirements

> 80% code coverage
> 200 alarms
```

DETECTION OF A SECURITY FLAW IN A COTS COMPRESSION LIBRARY

- **Result** CEA researchers identified a bug in the QuickLZ library. This bug was acknowledged by the designer and corrected in version beta 1.5.1.

- **Conclusion** Software analysis can be applied to general-purpose COTS, enabling their use in security-critical systems.

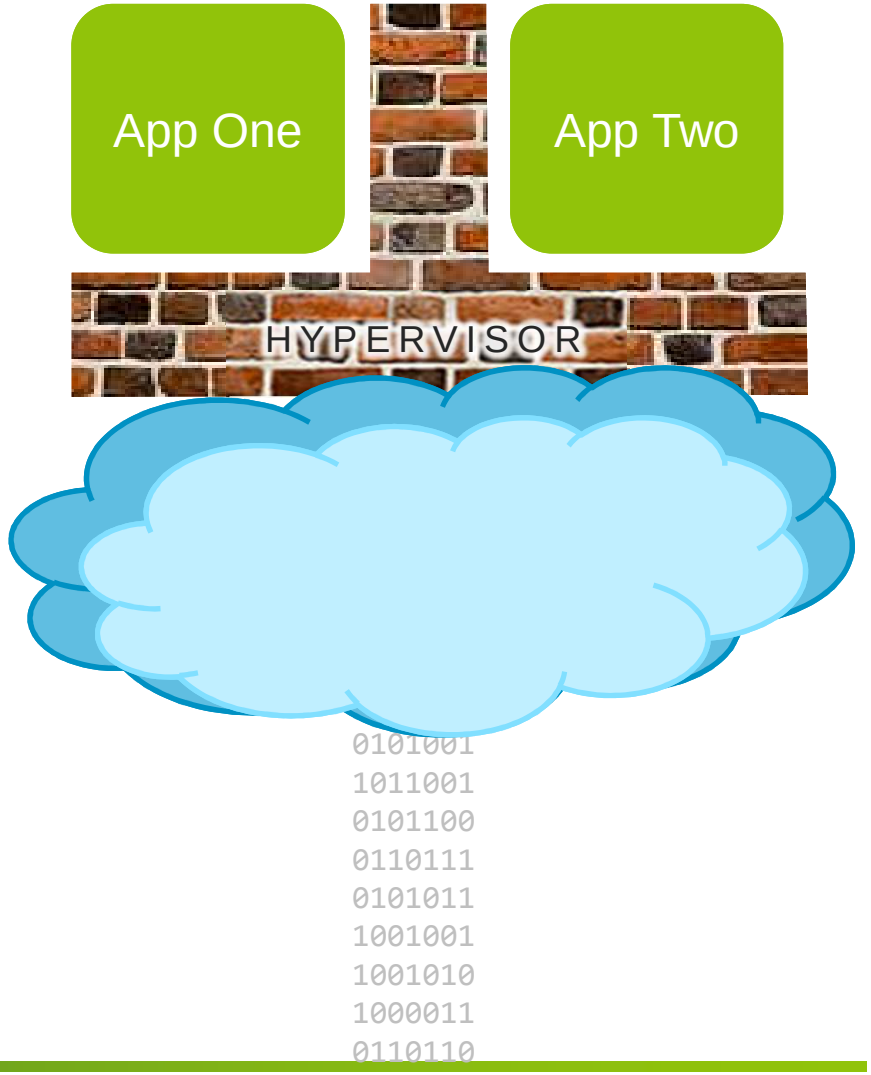



```
Fixed a condition where
QLZ_MEMORY_SAFE could fail
detecting corrupted data.
Thanks to Pascal Cuoq and
Kerstin Hartig who used Frama-
C's value analysis!
```

ADVANCED VALIDATION OF A SET OF HYPERVISOR PROPERTIES

- **Result** **Design-time security goals allow to specify and verify custom software properties.**

- **Conclusion** **A rigorous methodology can include the use of Frama-C to address strong software requirements.**



RUNTIME MONITORING AND VERIFICATION

- Result **Use program analysis and transformations to synthesize:**
  - **security monitors**
  - **fault injectors**

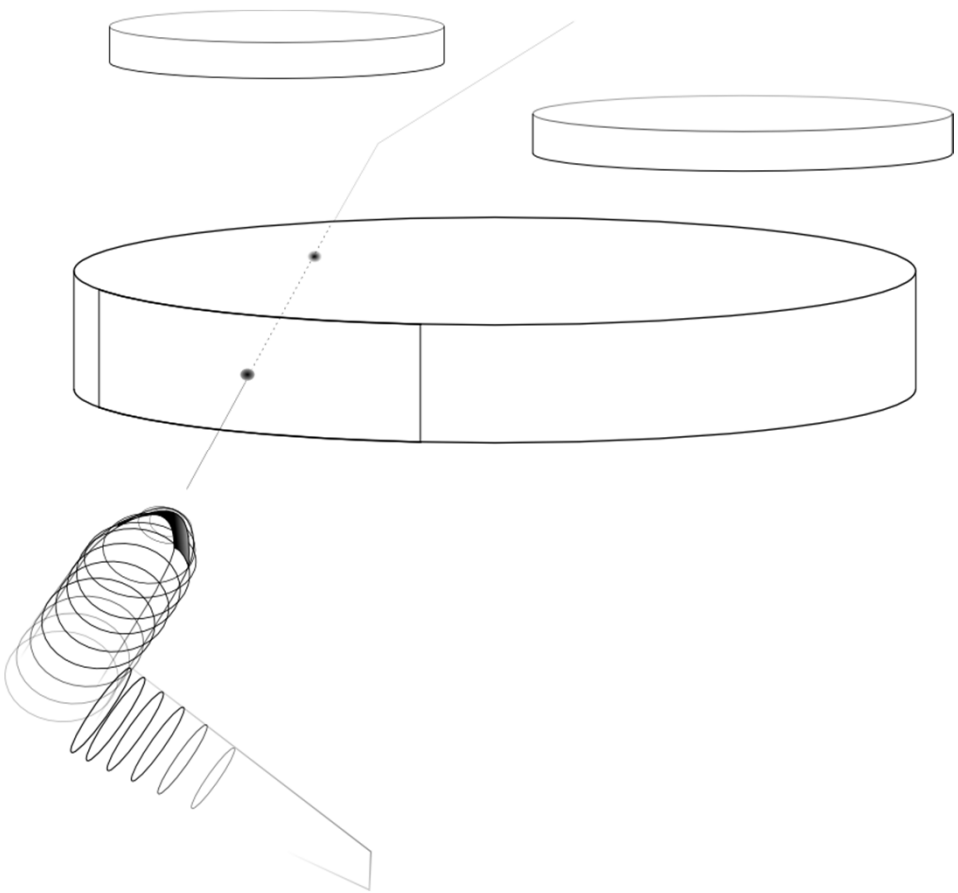
- Conclusion **Runtime code can be added to harden legacy software through hardware-enabled runtime verification.**

```
extern int a, b;
void f(int);

void g(){
  if (b == 0) a = 1;
  else if(b == 1) a = 2;
  else return;

  assert((a == 1 && b == 0) || (a == 2 && b == 1))
  f(a);
}
```

FORMAL ALGORITHMIC CONFORMANCE PROOF



```
% Conflict during interval [B,T]
conflict_2D?(s,v) : bool =
  EXISTS (t: Lookahead):
sqv(s+t*v) < sq(D)

% 2-D Conflict Detection (cd2d)
cd2d?(s,v) : bool =
  horizontal_los?(s+B*v) OR
omega_vv(s)(v) < 0

% THEOREM: cd2d is correct and
complete
cd2d : THEOREM
  conflict_2D?(s,v)
  IFF
  cd2d?(s,v)
```

Frama-C

A Software Analysis Perspective*

Pascal Cuoq, Florent Kirchner, Nikolaï Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski

with Patrick Baudin, Richard Bonichon, Bernard Botella, Loïc Correnson, Zaynah Dargaye, Philippe Herrmann, Benjamin Monate, Yannick Moy, Anne Pacalet, Armand Puccetti, Muriel Roger, and Nicky Williams

CEA, LIST, Software Safety Laboratory, PC 174, 91191 Gif-sur-Yvette France
firstname.lastname@cea.fr

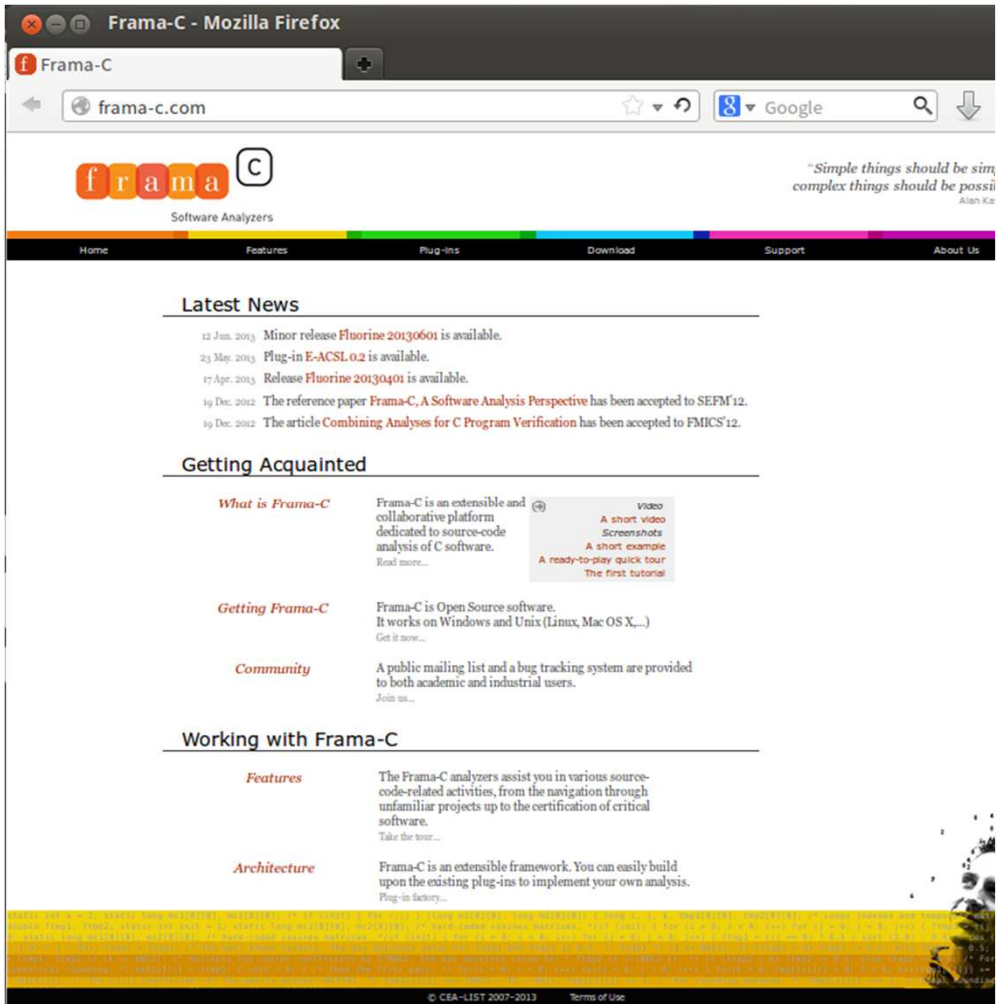
Abstract. Frama-C is a source code analysis platform that aims at conducting verification of industrial-size C programs. It provides its users with a collection of plug-ins that perform static analysis, deductive verification, and testing, for safety- and security-critical software. Collaborative verification across cooperating plug-ins is enabled by their integration on top of a shared kernel and datastructures, and their compliance to a common specification language. This foundational article presents a synthetic view of the platform, its main and composite analyses, and some of its industrial achievements.

Introduction

he past forty years have seen much of the groundwork of formal software analsis being laid. Several angles and theoretical avenues have been explored, from eductive reasoning to abstract interpretation to program transformation to conolic testing. While much remains to be done from an academic standpoint, ome of the major advances in these fields are already being successfully impleented [18,41,21,47,52] – and met with growing industrial interest. The ensuing ush for mainstream diffusion of software analysis techniques has raised several hallenges. Chief among them are: *a.* their scalability, *b.* their interoperability, nd *c.* the soundness of their results.

Point (a) is predictably important from the point of vue of adoptability. Scaling o large problems is a prerequisite for the industrial diffusion of software analysis nd verification techniques. It also represents a mean to better understand how nguage idioms (e.g. pointers, unions, or dynamic memory allocation) influence he underlying architecture of large software developments. Overall, achieving calability in the design of software analyzers for a wide range of software patrns remains a difficult question.

*This work was partly supported by ANR U3CAT and Veridyc, and FUI9 Hi-Lite projects



METHODS AND TOOLS FOR HCS&S

- Scientific roots and community
  - Formal proof
  - Model checking
  - Constraint solving
  - Simulation
  - Abstract interpretation
  - Test case generation
  - Architecture Exploration
  - Synchronous languages
- Prototyping and development of industrial-strength tools
  for academia
  for the industry

Guided by industrial requirements

Scaling & Performance

- Objectives
  Raise the level of confidence in software
  Lower the costs of verification
  Enable the evaluation of academic prototypes on industrial problems

Technological strategy	creation of collaborative platforms
Scientific strategy:	combination of approaches
Applicative strategy:	cross-domain fertilization – aero, space, rail, energy, banking, defense

TRUST IN SOFT

Frama-C for software safety and security

www.trust-in-soft.com

SOFTWARE DEVELOPERS

- **Industrial support**
- **Commercial licenses**
- **Preinstalled workstations**

SOFTWARE INTEGRATORS

Off-the-shelf validation kits for common open-source packages

SERVICE PROVIDERS

Dedicated affiliate programs

Laboratoire pour la Sûreté des Logiciels
Département d'Ingénierie des Logiciels et Systèmes
CEA LIST

Florent Kirchner
florent.kirchne@cea.fr