DE LA RECHERCHE À L'INDUSTRIE

www.cea.fr

From 80% to 99%

An Industrial Collaboration for Automating Frama-C/WP

L. Correnson

FRAMA-C & SPARK DAY 2017

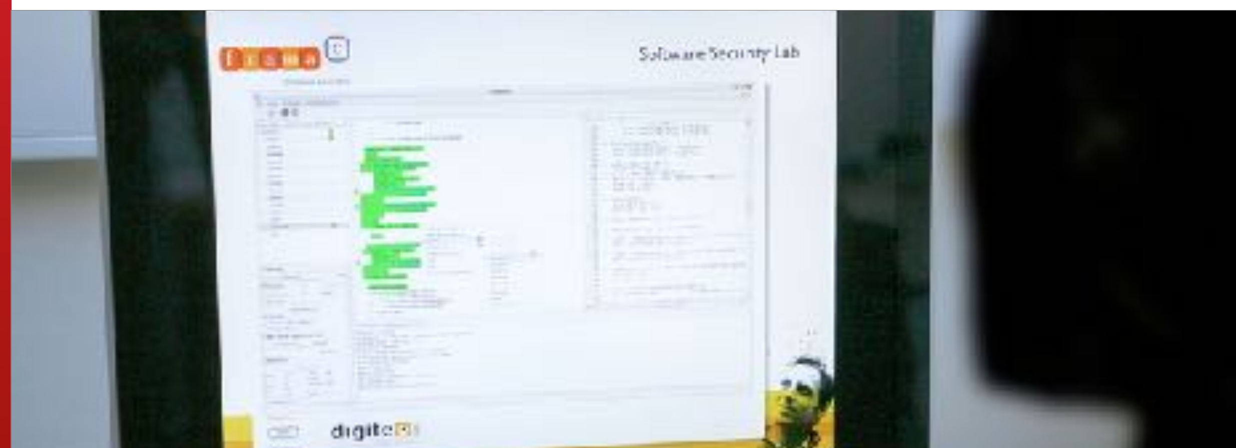
DE LA RECHERCHE À L'INDUSTRIE

cea

From ~~80%~~ 50% to ~~99%~~ 100%

An Industrial Collaboration for Automating Frama-C/WP (nearly)

L. Correnson



list

www.cea.fr

FRAMA-C & SPARK DAY 2017

DE LA RECHERCHE À L'INDUSTRIE

cea

From ~~80%~~ 50% to ~~99%~~ 100%

An Industrial Collaboration for Automating Frama-C/WP (nearly)

L. Correnson

AIRBUS

J. Souyris

list

www.cea.fr

FRAMA-C & SPARK DAY 2017

Caveat

… FROM UNIT TESTS TO UNIT PROOFS …

Caveat Tool (late 2000s)

- ☑ Formal Specifications
- ☑ Automated Proof (by Rewriting)
- ☑ Interactive Proof Transformation

Caveat @ Airbus (2005)

- ☑ Replacement for Unit Tests
- ☑ Complete Behaviours as Test Plan
- ☑ Dedicated Memory Model
- ☑ Limited Aliasing & Coding Rules
- ☑ Runtime-Error-Free Hypothesis
- ☑ Qualified for DO 178 B

…WHEN FINALLY CAME

WP

A PARTNERSHIP STORY

From Caveat to NUPW

2010 — 2017

THE SITUATION (2010)

- Caveat Obsolescence
- Caveat Limitations
- Promising Frama-C

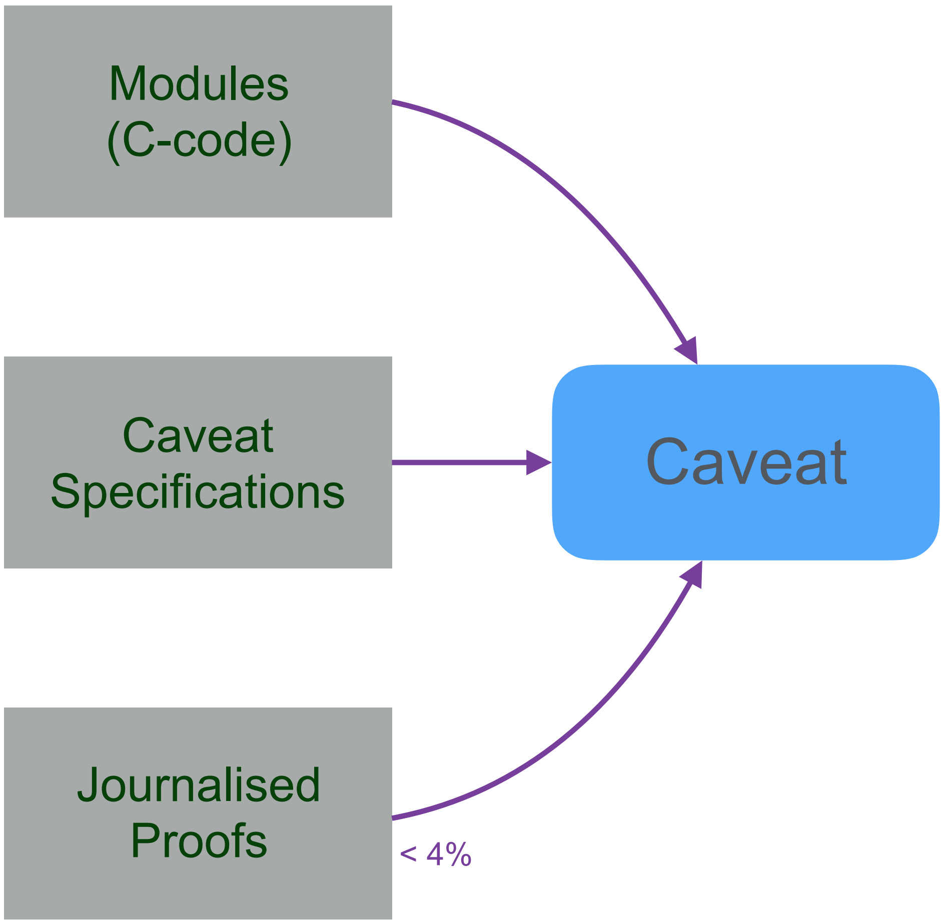
THE SITUATION (2010)

- ☑ Caveat Expertise
- ☑ Deployed Unit Proofs
- ☑ Extended with Alt-Ergo
- ☑ **96%** Automation Rate
- ☑ Aliasing Limitations (Review)

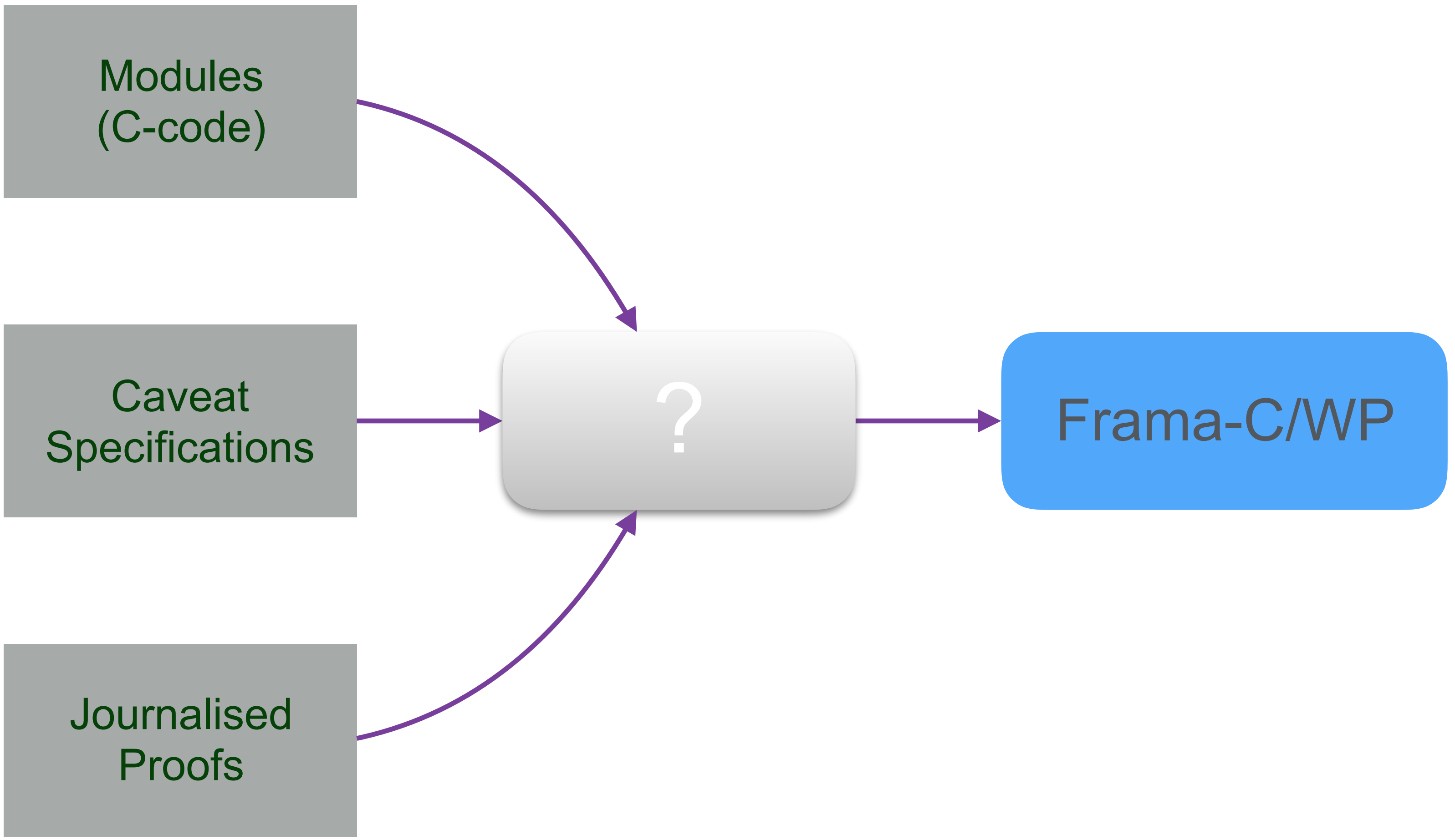
OBJECTIVES

- ☑ **16 modules bench**
- ☑ Automation Rate > **96%**
- ☑ Caveat in-place Replacement

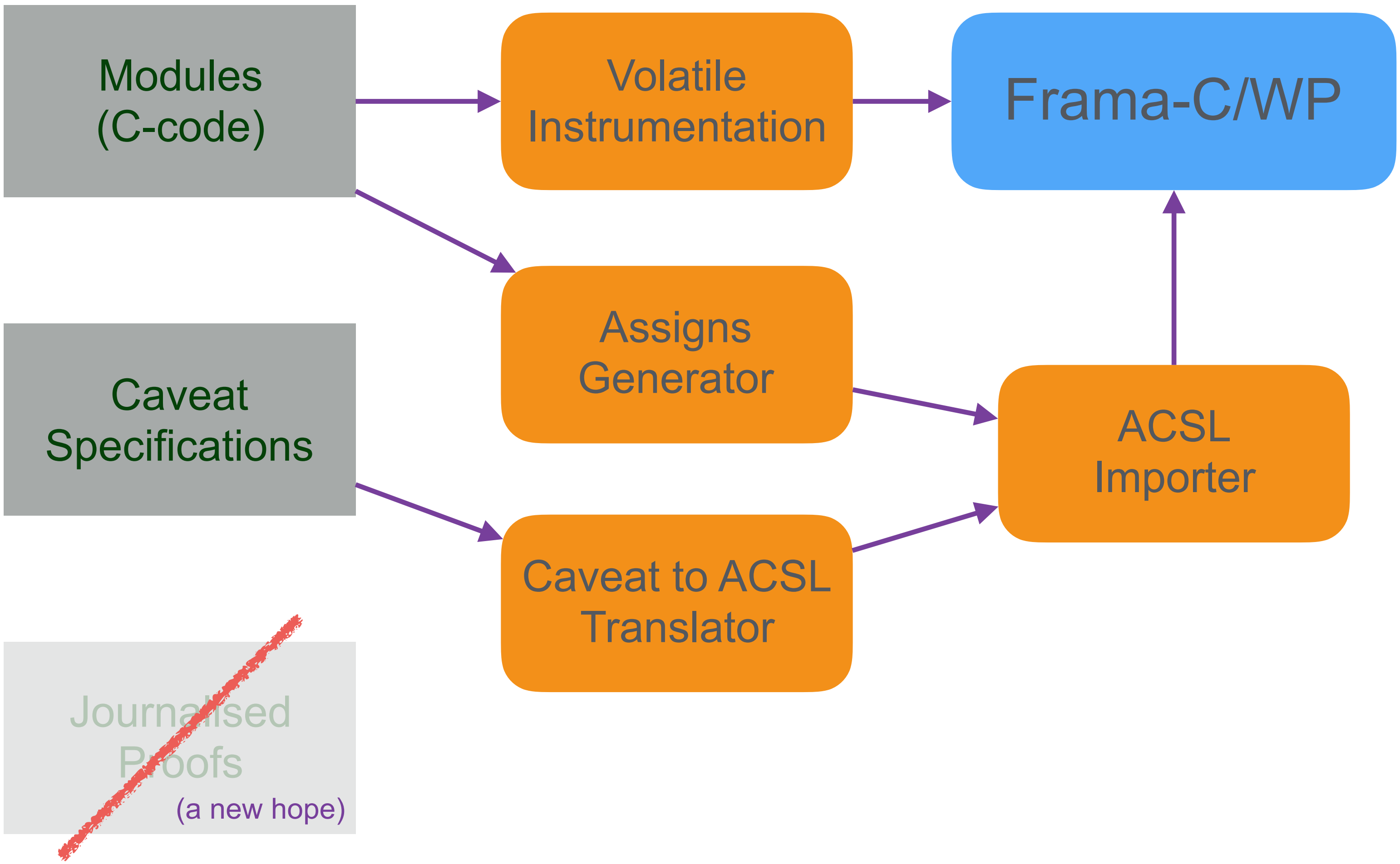
AVAILABLE ARTEFACTS



MIGRATION PROJECT



MIGRATION PROJECT



AIRBUS NUPW 2010 — 2012

THE GAP !

Module	#VC	WP	Alt-Ergo	Coq	Failed	Success
Bench #1	13	4	9	-	-	100.0 %
Bench #2	54	11	13	30	-	100.0 %
Bench #3	35	6	8	17	4	88.6 %
Bench #4	Timeout					

Automation < 50%

— 2011 —

AIRBUS (*a bit anxious*)

« Hey, CEA, you shall do something, aren’t you? »

CEA (*embarrassed*)

« … ah, really, you don’t like Coq? »

AIRBUS (***furious***)

« … »

CEA (*confused*)

« … ok, ok, let’s try something! »

INTRODUCING QED IN FRAMA-C/WP

Module	#VC	WP	Alt-Ergo	Coq	Failed	Success
`Bench #1`	11	11				**100 %**
`Bench #2`	25	25				**100 %**
`Bench #3`	22	18	3		**1**	**95 %**
`Bench #4`	172	116	56			**100 %**

— 2012 —

INTRODUCING QED IN FRAMA-C/WP

(Published 2014)

Qed. Computing What Remains to Be Proved

Loïc Correnson

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

Abstract. We propose a framework for manipulating in a efficient way terms and formulæ in classical logic modulo theories. Qed was initially designed for the generation of proof obligations of a weakest-precondition engine for C programs inside the Frama-C framework, but it has been implemented as an independent library. Key features of Qed include on-the-fly strong normalization with various theories and maximal sharing of terms in memory. Qed is also equipped with an extensible simplification engine. We illustrate the power of our framework by the implementation of non-trivial simplifications inside the Wp plug-in of Frama-C. These optimizations have been used to prove industrial, critical embedded softwares.

1 Introduction

In the context of formal verification of critical softwares, the recent fantastic improvement of automated theorem provers and SMT solvers[1] opens new routes. Inside the Frama-C [2] platform, we have developed the Wp plug-in to implement an efficient *weakest precondition calculus* to formally prove a C program against its specification. The specification is written in terms of the "ANSI-C Specification Language" (ACSL) [3], which is a first-order logic system with dedicated constructs to express C properties such as pointer validity and floating point operations.

The Wp plug-in actually compile C and ACSL constructs into an internal logic representation that is finally exported to SMT solvers and other theorem provers. Thus, we need an internal system to represent and manipulate first-order logical formulæ. This is exactly what Qed has been designed for.

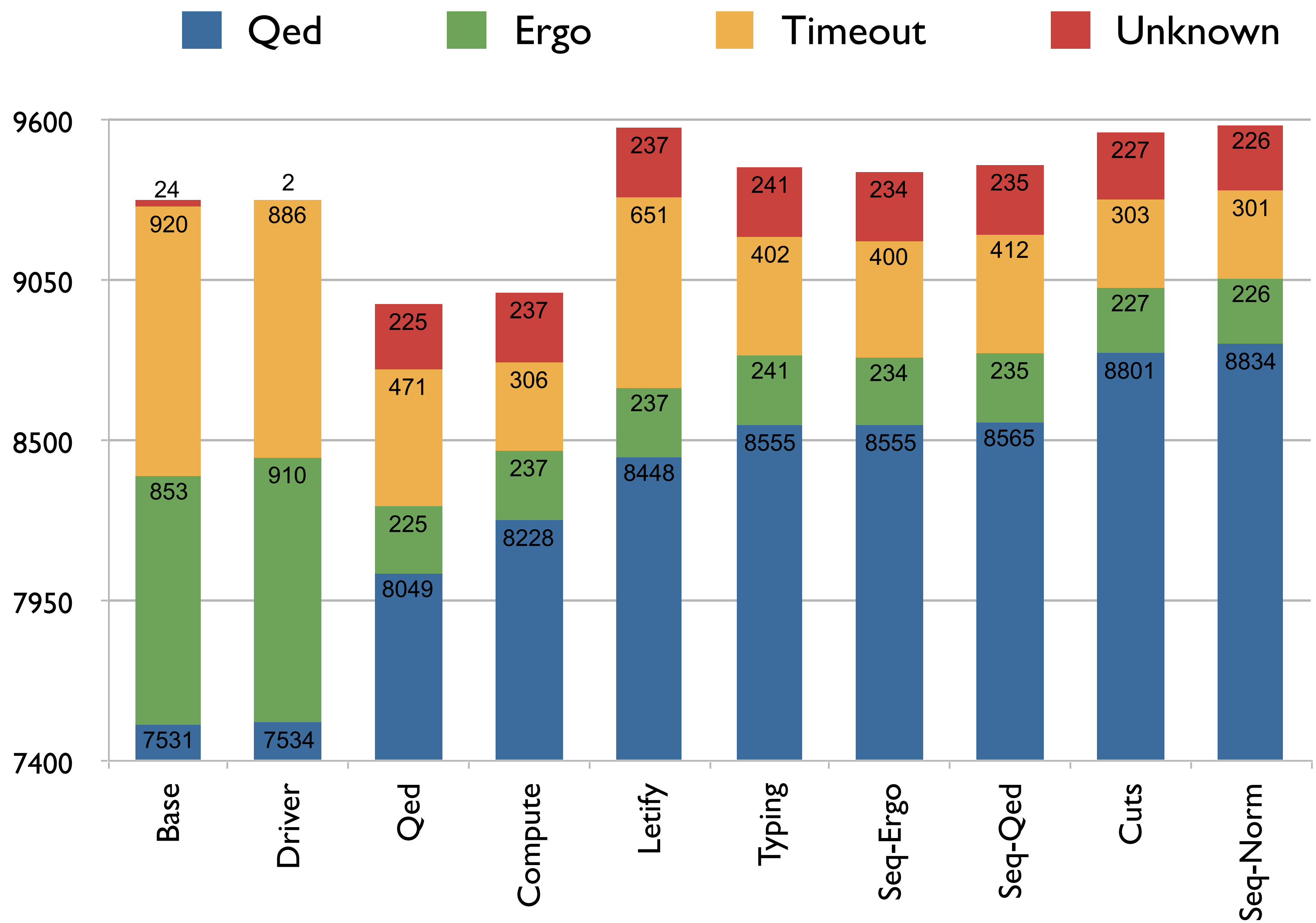
Designing such a library is not difficult in itself. Some datatype is needed for expressing terms and properties, combined with pretty-printing facilities to export them into several languages. This is what we implemented in our early prototypes.

However, experimental results shown that a formula can not be naively build then translated and finally sent to an external back-end prover. We actually observed limitations of such a naive approach on real life examples from critical embedded software:

– SMT solvers are quite efficient, but they are sensitive to the amount of hypotheses they receive. Having a proof for $A \rightarrow B$ does not mean you will have a proof for $A \wedge A' \rightarrow B$.

J.M. Badger and K.Y. Rozier (Eds.): NFM 2014, LNCS 8430, pp. 215–229, 2014.

MATURING QED & WP



2011 — 2012

NUPW AUTOMATION

AIRBUS 16-Modules Bench																
2012	93%	88%	88%	73%	89%	96%	100%	92%	98%	68%	35%	79%	96%	92%	64%

— 2012 —

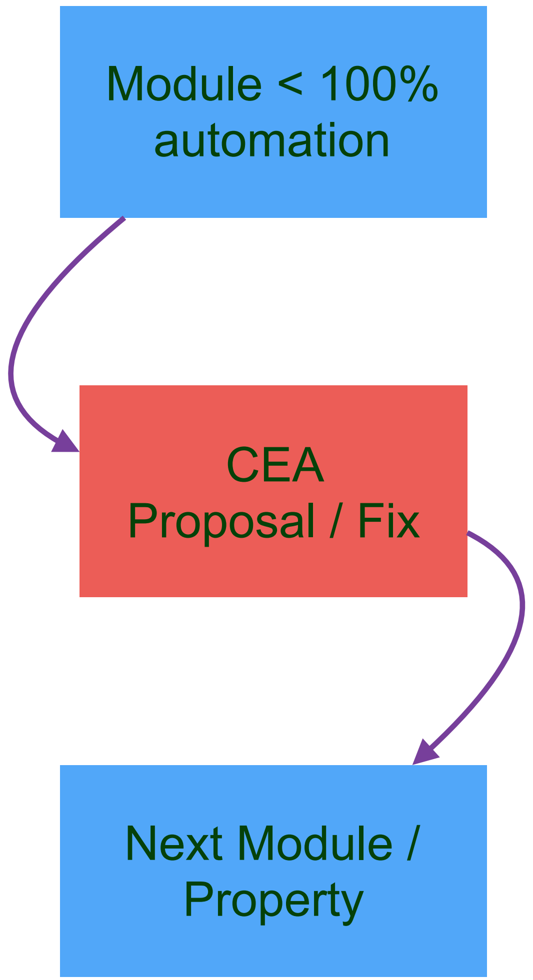
AIRBUS (*ambitious*)

« Seems that 100% is reachable… »

CEA (*volunteer*)

« Go! we setup a task force! »

SYSTEMATIC APPROACH

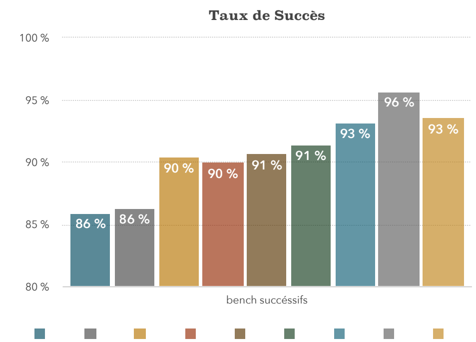


20 GitLab Issues
3 CEA Researcher-Engineers
Continuous Progression Reports

L. Correnson
P. Baudin
F. Bobot

- #1. (Qed) typing lets with alt-ergo
- #2. (Qed) traduction des booléens
- #3. (Qed) remontée des conditionnelles
- #4. (WP) modèle Caveat
- #5. (WP) assigns des tableaux et structures
- #6. (Qed) control-flow au travers des ifs
- #7. (WP) comparaison des structures
- #8. (WP) global-const
- #9. (NUPW) driver des séquences
- #10. (NUPW) répétition dans les séquences
- #11. (Qed) preuve des initialiseurs de tableaux
- #12. (WP) \let dans les prédicats ACSL
- #13. (WP) affaiblissement des quantificateurs sur les entiers
- #14. (WP) warning sur les tableaux infinis
- #15. (WP) exit behavior des simulés
- #16. (Qed) conversion des décalages d'entiers
- #17. (WP) références initialement valides
- #18. (FRAMA-C) consolidation des benchs
- #19. (WP) séparation des formelles entre les modèles Hoare et Typed
- #20. (Qed) perte de partage

NUPW – TRAVAUX 2014



#	Tâche	Closed
#1	Typage pour Alt-Ergo	☑
#2	Egalité booléenne	☑
#3	Egalité conditionnelle	☑
#4	Modèle Caveat	☑
#5	Assigns de structures	☐
#6	Flots d'égalités dans les branches	☑
#7	Egalité de structures	☐
#8	Variable globales constantes	☑
#9	Driver des séquences	☑
#10	Répétition de séquences	☑
#11	Initialisation des grands tableaux	☑
#12	ACSL let-in de propriétés	☐
#13	Contraintes des entiers C	☑
#14	Bug sur tableaux infinis	☑
#15	Exit behavior (related to global const)	☑
#16	Conversions bits-à-bits	☑
#17	Références initialement valides	☑
#18	Export des résultats de WP	☐
#19	Séparation entre Hoare et Typed	☑
#20	Perte de partage en sortie de Qed	☑

#	Tâche	Closed
#21	Initialiseurs Compacts	☑
#26	Typage des tableaux pour Alt-Ergo	☑
#27	Introduction des existentielles	☑
#28	Filtrage agressif des hypothèses	☑
#31	Pré-requis des simulés inappropriés	☑
#32	Simplification des expr. constantes	☐
		☐
		☐
		☐
		☐
		☐
		☐
		☐
		☐
		☐
		☐
		☐
		☐
		☐
		☐

NUPW AUTOMATION

AIRBUS 16-Modules Bench																
2012	93%	88%	88%	73%	89%	96%	100%	92%	98%	68%	35%	79%	96%	92%	64%
2014	100%	99%	99%	100%	95%	100%	100%	97%	96%	81%	95%	100%	100%	95%	85%

2012 — 2014

AIRBUS

« We need to deploy! »

« For Caveat migration *and* for new projects! »

CEA (*a bit nervous*)

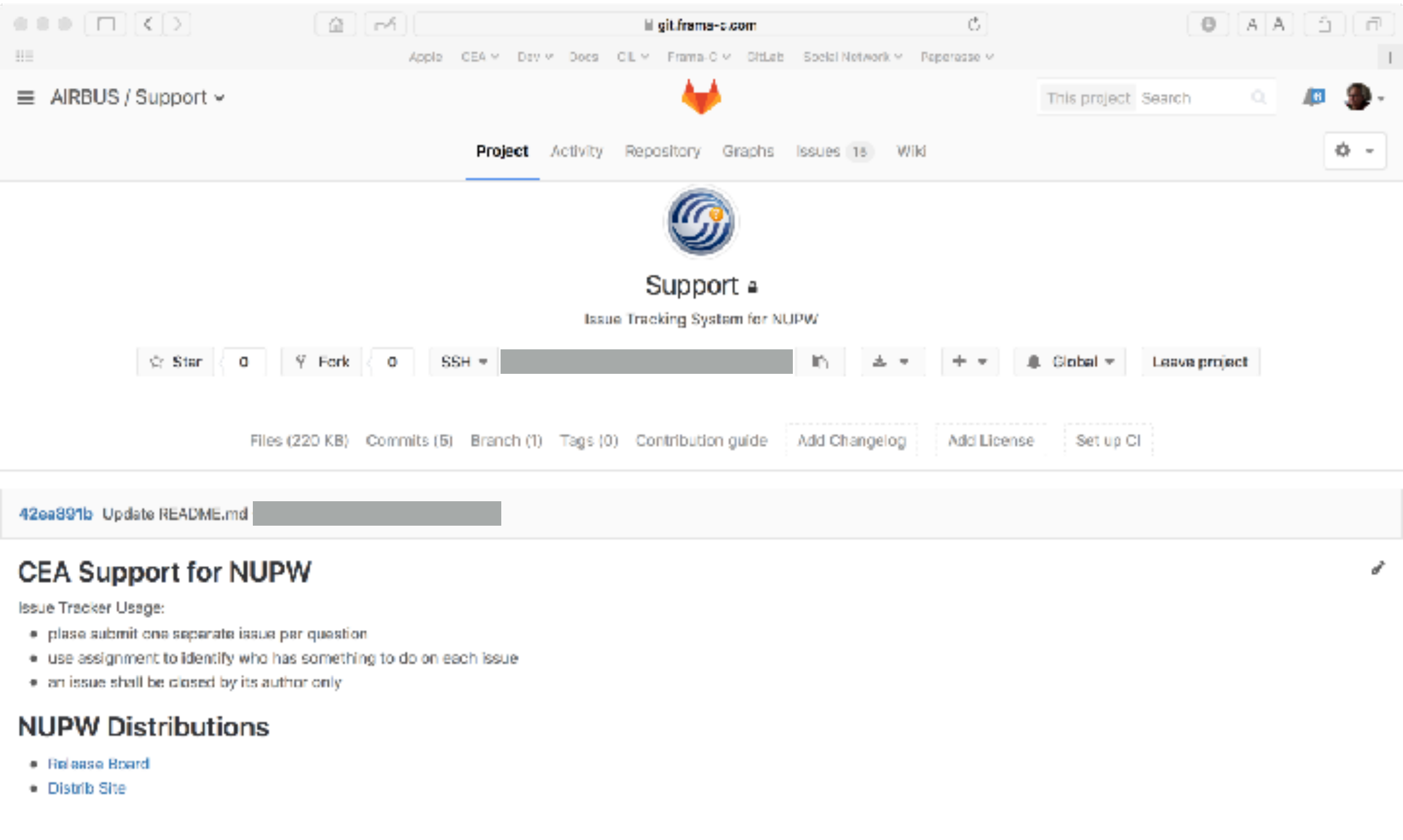
« Now? »

CEA (*apart*)

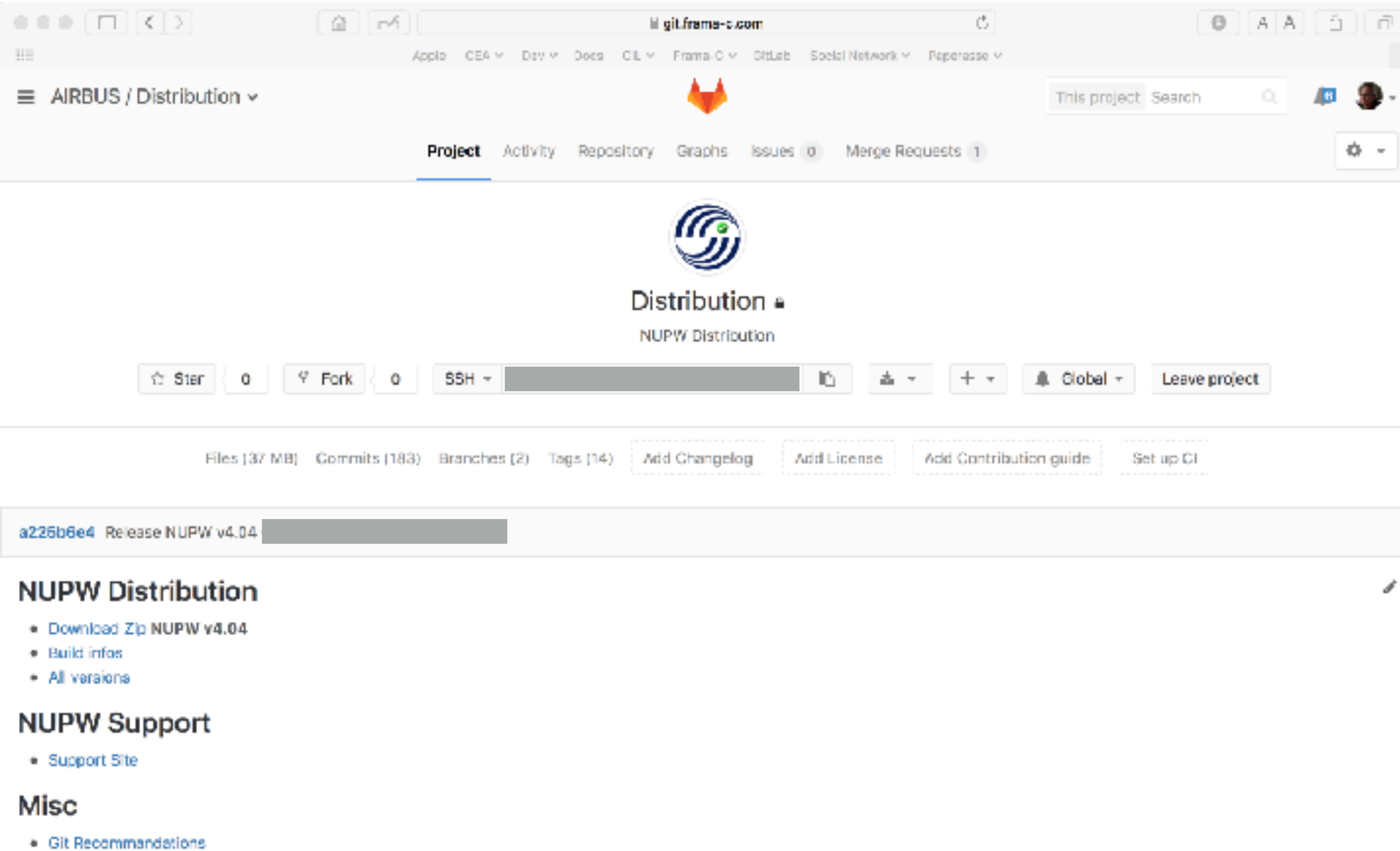
« we need an organisation »

— 2017 —

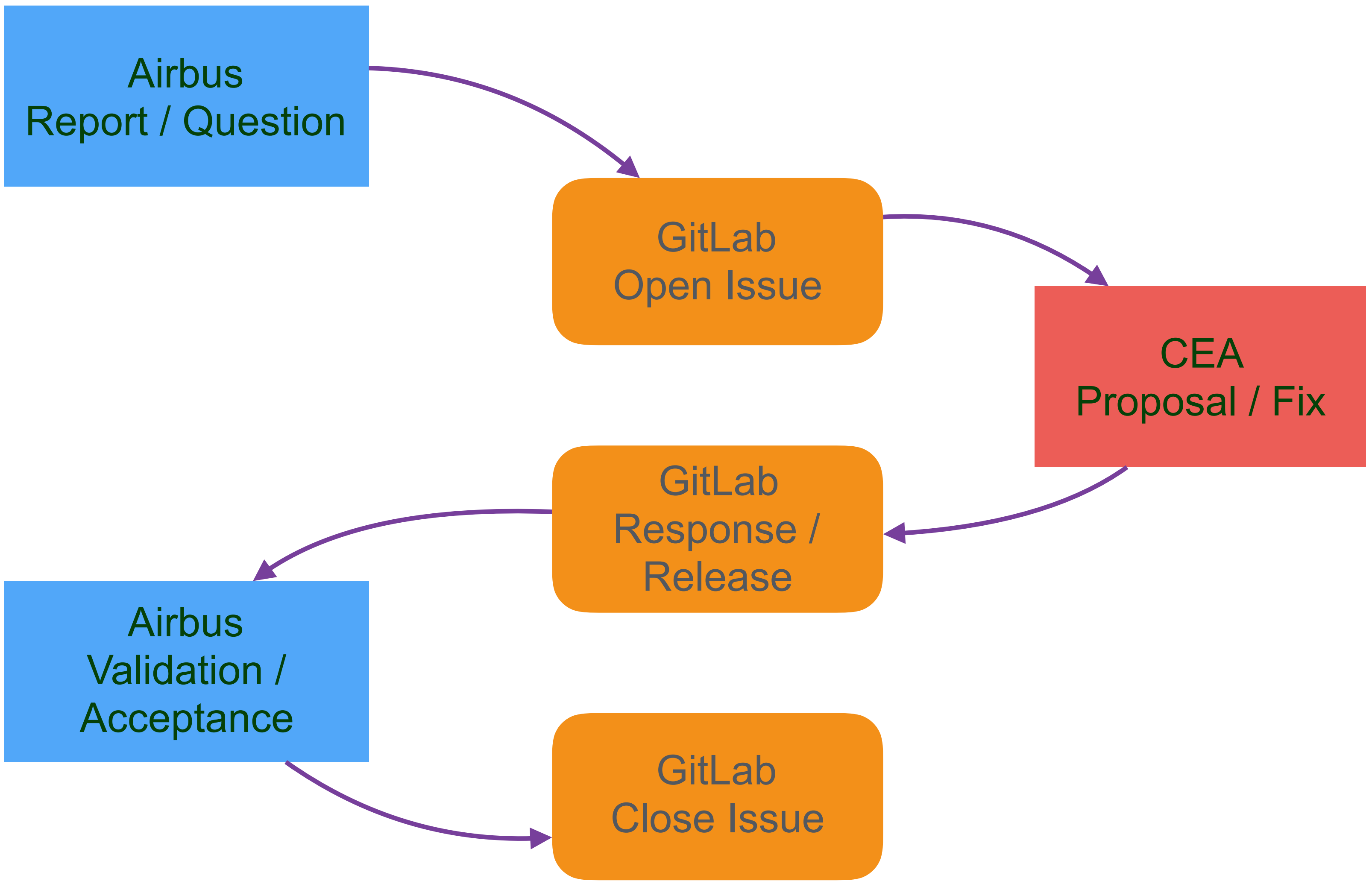
GITLAB TO RESCUE



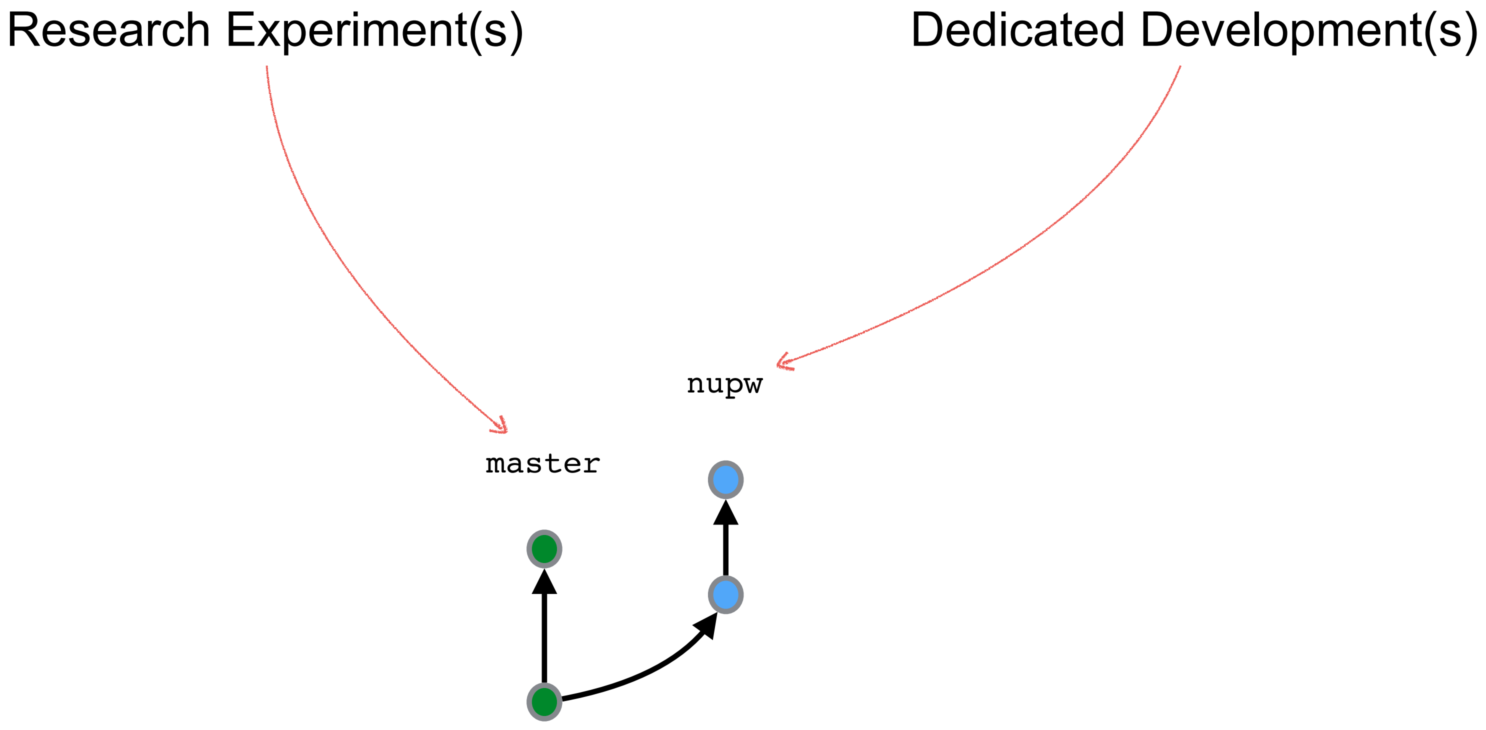
GITLAB TO RESCUE



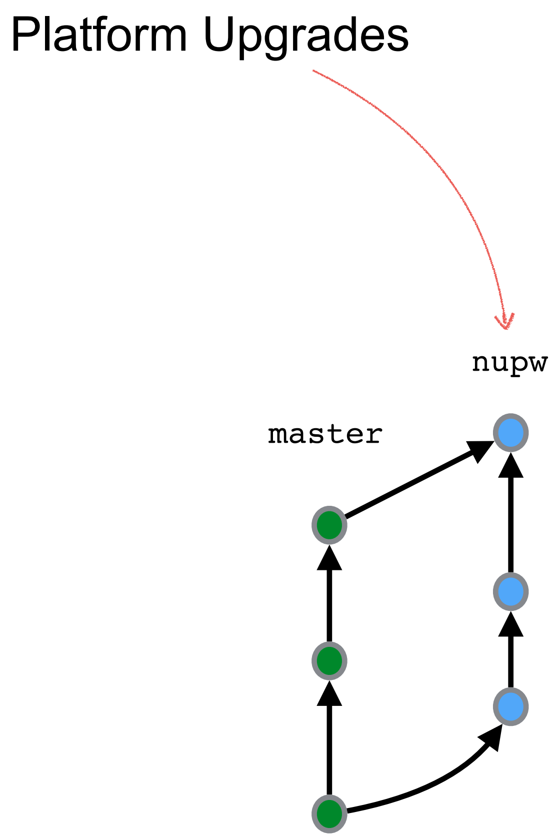
« SUPPORTING » MODE



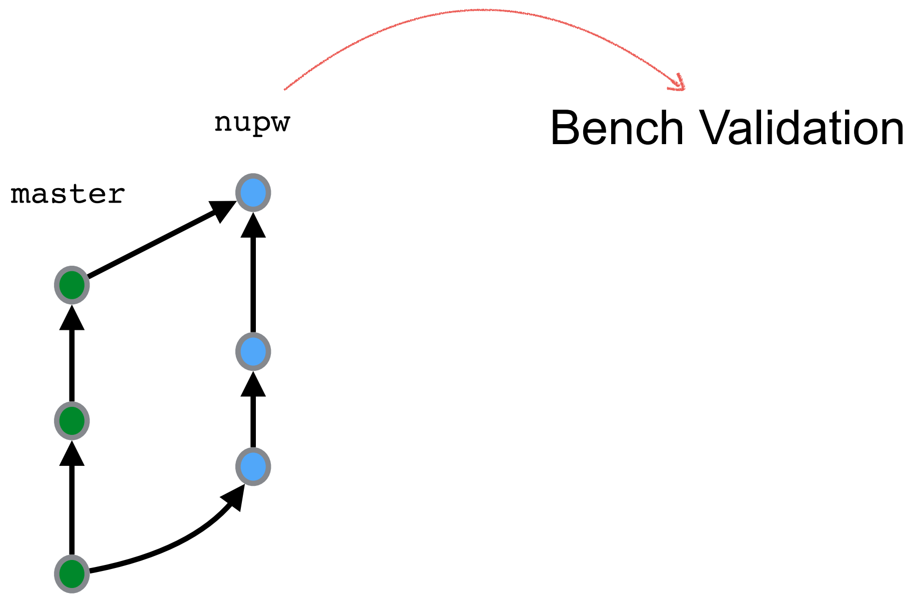
MIXING « R » & « D » VIA GIT



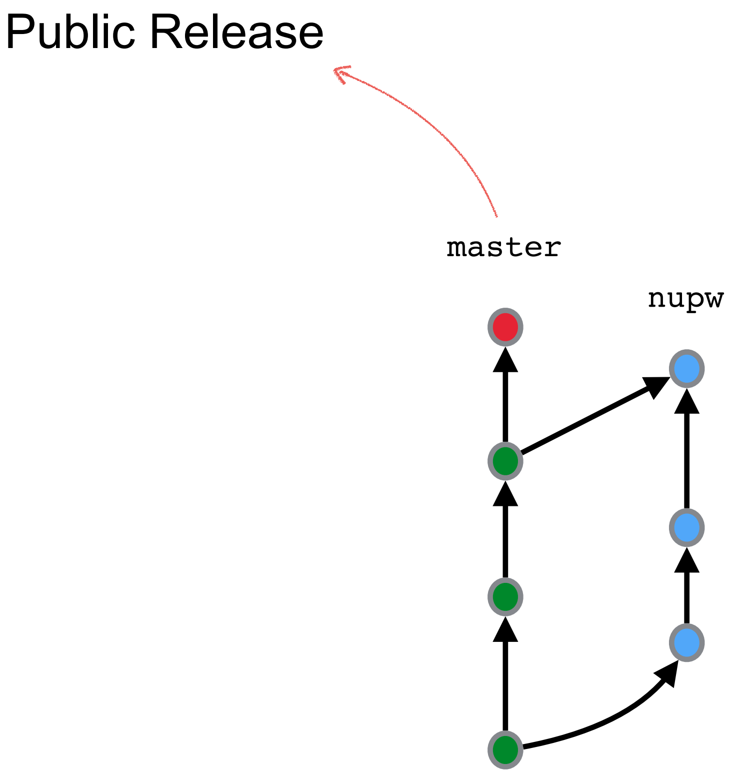
MIXING « R » & « D » VIA GIT



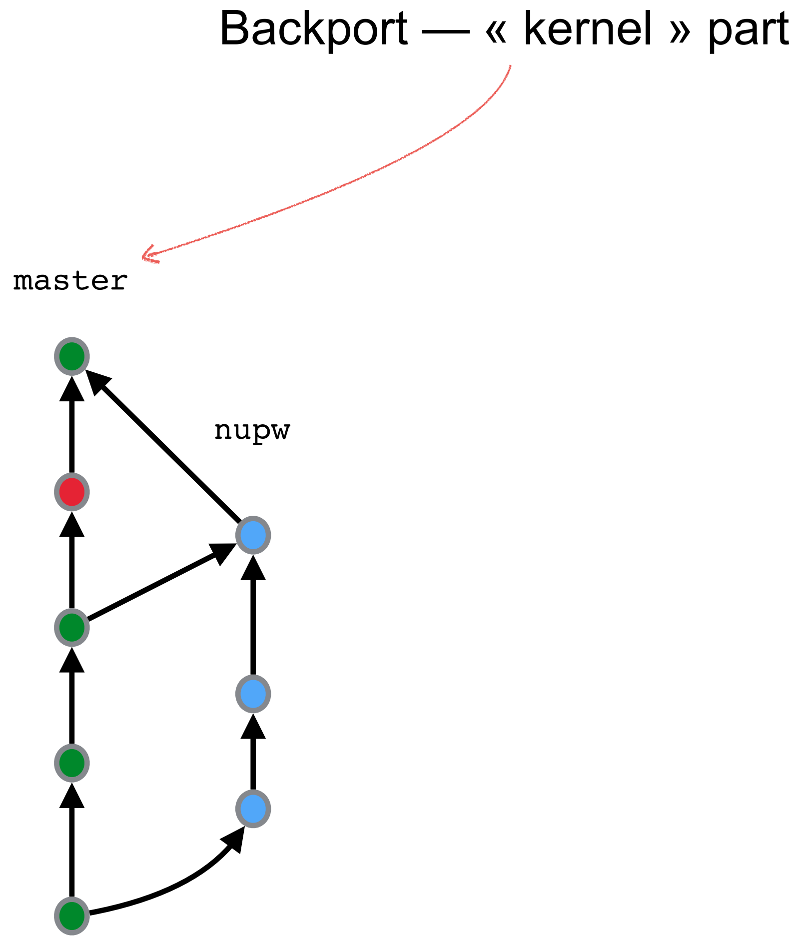
MIXING « R » & « D » VIA GIT



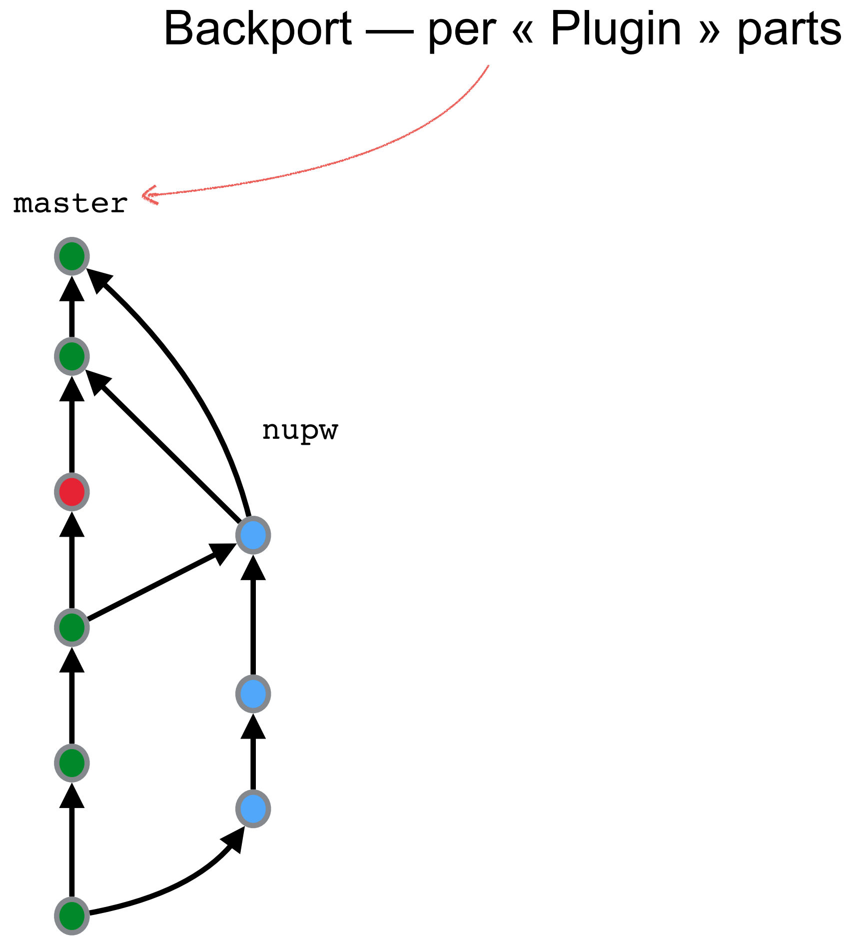
MIXING « R » & « D » VIA GIT



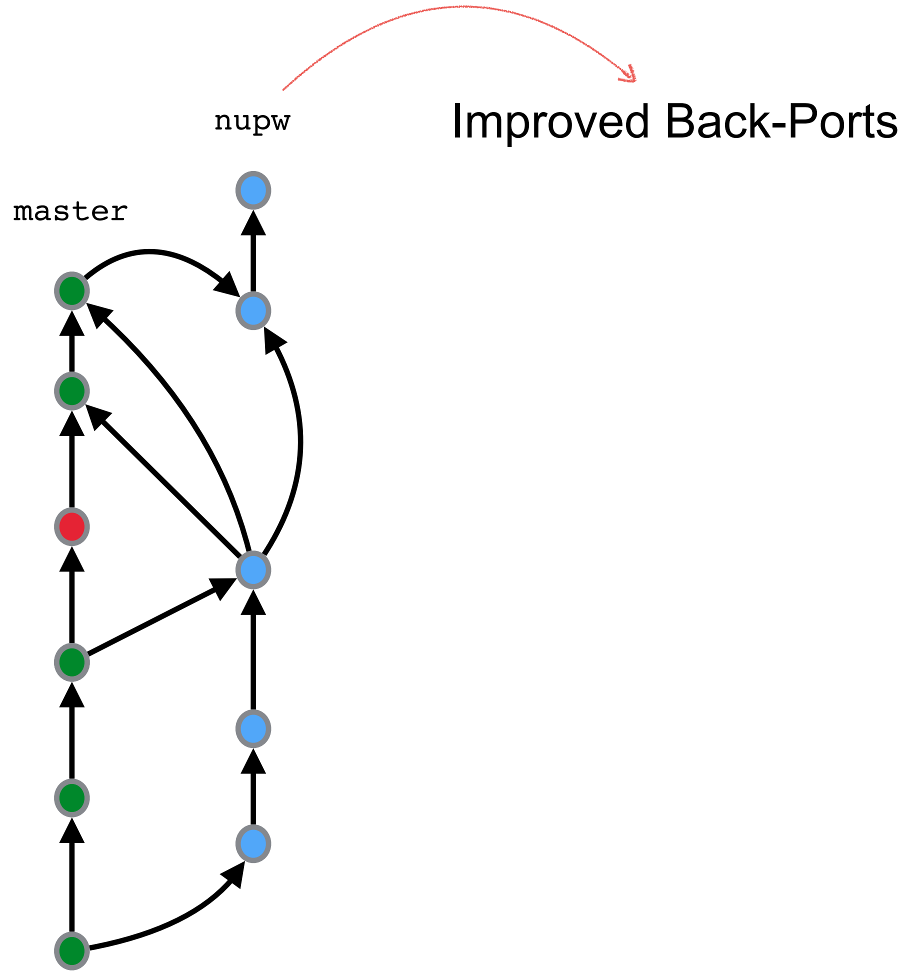
MIXING « R » & « D » VIA GIT



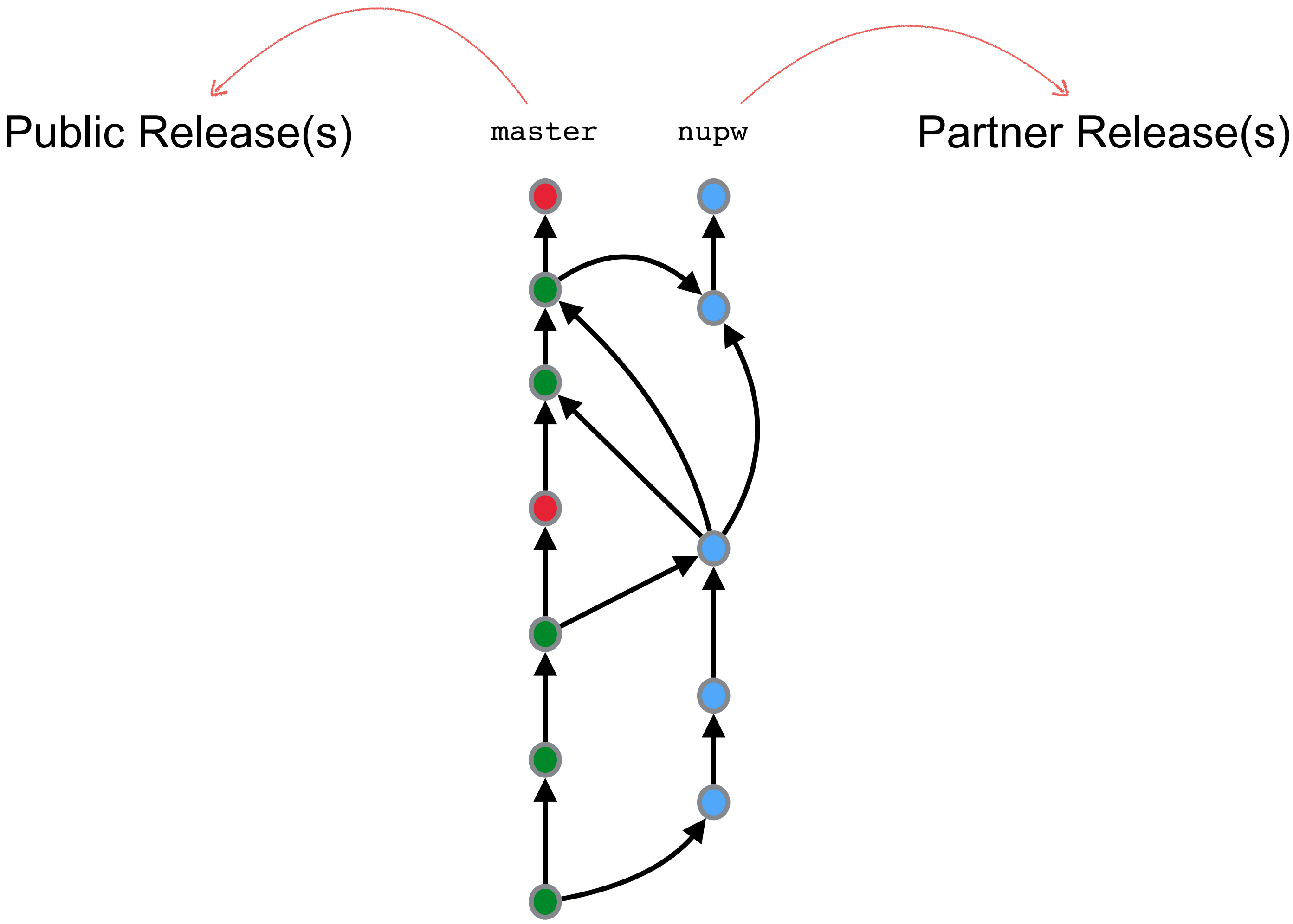
MIXING « R » & « D » VIA GIT



MIXING « R » & « D » VIA GIT



MIXING « R » & « D » VIA GIT



THE TECHNICAL PARTS

- ☑ Qed as an intermediate representation
- ☑ Inlined normalisation to avoid combinatorial explosion
- ☑ Maximal in-memory sharing preservation
- ☑ Agressive constant & equality propagation
- ☑ Integer modulo computations
- ☑ Integer bitwise simplifications
- ☑ Loop unrolling heuristics
- ☑ Hypothesis compaction & erasing
- ☑ Hypothesis generalisation (eg. init parts)
- ☑ Systematic branch pruning
- ☑ Type synthesis for Alt-Ergo
- ☑ Small bugs in Alt-Ergo (few, indeed)
- ☑ External drivers (closer to SMT)
- ☑ Dedicated simplifications (beyond SMT)
- ☑ Proof obligation understanding & retro-engineering
- ☑ User-defined simplifications
- ☑ Elicit Caveat Aliasing (no more manual review)
- ☑ Kernel-aware runtime errors (overflows & downcasts)
- ☑ …

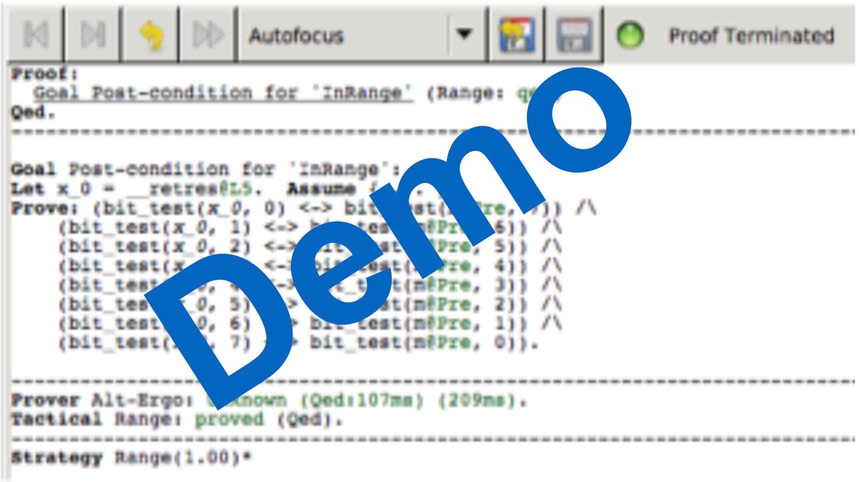
NUPW AUTOMATION

AIRBUS 16-Modules Bench																
2012	93%	88%	88%	73%	89%	96%	100%	92%	98%	68%	35%	79%	96%	92%	64%	
2014	100%	99%	99%	100%	95%	100%	100%	97%	96%	81%	95%	100%	100%	95%	85%	
2016 (auto / tip)	100%	100%	90 / 99%	100%	100%	100%	100%	99 / 100%	97 / 100%	95 / 100%	100%	100%	100%	96 / 100%	94 / 95%	

2012 — 2017

THE LAST 1% TO BE PROVED

Frama-C/WP
Interactive Proof Transformer



— 2017 —

THE LAST 1% TO BE PROVED

Frama-C/WP
Interactive Proof Transformer

- ☑ Goal exploration / view
- ☑ Memory model interpretation
- ☑ User-defined Tactics
- ☑ User-defined Heuristics
- ☑ Replay Scripts
- ☑ Code & Spec reconciliation

In Frama-C Phosphorus

TIMELINE (CONCLUSION)

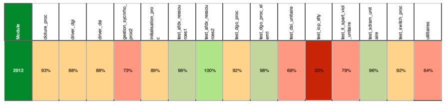
2011 | 2014 | 2017

R&D | CAVEAT — **R&D | NEW UNIT PROOF WORKSHOP** — **R&D | LLR**

SUPPORT

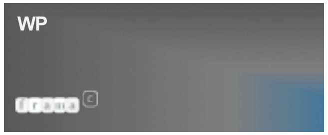
2008

A 10 year investment in static code analysis, the Caveat tool from CEA is used in production to validate safety-critical code in the A380 program, and a few years later on the A350 and A400M



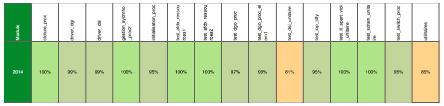
2011

Obsolescence management triggers the investigation of tooling renewal, and the identification of compatibility and performance challenges in proposed solutions



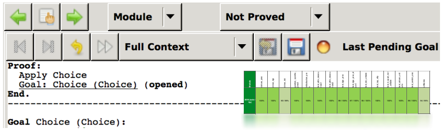
2012

Teams at CEA List complete the development of the Frama-C/WP plugin, complete with migration helpers: together they form the new unit proof workshop NUPW



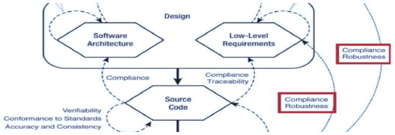
2014

Efficient reasoning techniques dramatically boost the level of proof automation, and bring NUPW to performance-parity with legacy tooling



2016

The Frama-C/WP plugin is extended to provide advanced interactive features that support the proof engineering phases, while Airbus and List teams setup regression baselines and training courses



2017

An enhanced support contract accompanies the deployment of NUPW to operational teams

Airbus engineers complete the design of a formal language for Low-Level Requirements, setting the stage for the design of innovative static and dynamic verifications

AIRBUS & CEA (together)

« That’s an R&D partnership! »