Différents cas d’application de l'Analyse Statique avec Frama-C dans un contexte industriel

DAS Systèmes Embarqués

28/06/2013

Agenda

- Use Case 1
  "Integrating Runtime Error Detection in the process with Frama-C"
- Use Case 2
  "Industrial development of Frama-C plugin based on semantic analysis"
- Conclusion

Use Case 1: Integrating Runtime Error Detection in the process with Frama-C

28/06/2013

Use Case 1
Introduction

- Objectives
  - Detecting runtime error by static analysis
  - Sound analysis expected
  - Detecting runtime-error by non-specialists
  - Detecting runtime-error earlier in the process
  - Quick and iterative process

Use Case 1
Introduction

- Examples of detections
  - Integer overflow

```
int f_overflow(int x)
{
  /*@ assert (x_0+1 ≤ 2147483647);
        // synthesized
     */
  return x+1;
}
```

  - Invalid pointer

```
int f_invalid_ptr()
{
  int* ptr =0;
  /*@ assert \valid(ptr);
        // synthesized
     */
  return *ptr;
}
```

  -

```
int tab[4] = {2,4,0,8};
int f_divided_0(int a, int b)
{
  int index = b&3;

  /*@ assert (tab[index] ≢ 0);
        // synthesized
     */
  return a/tab[index];
}
```

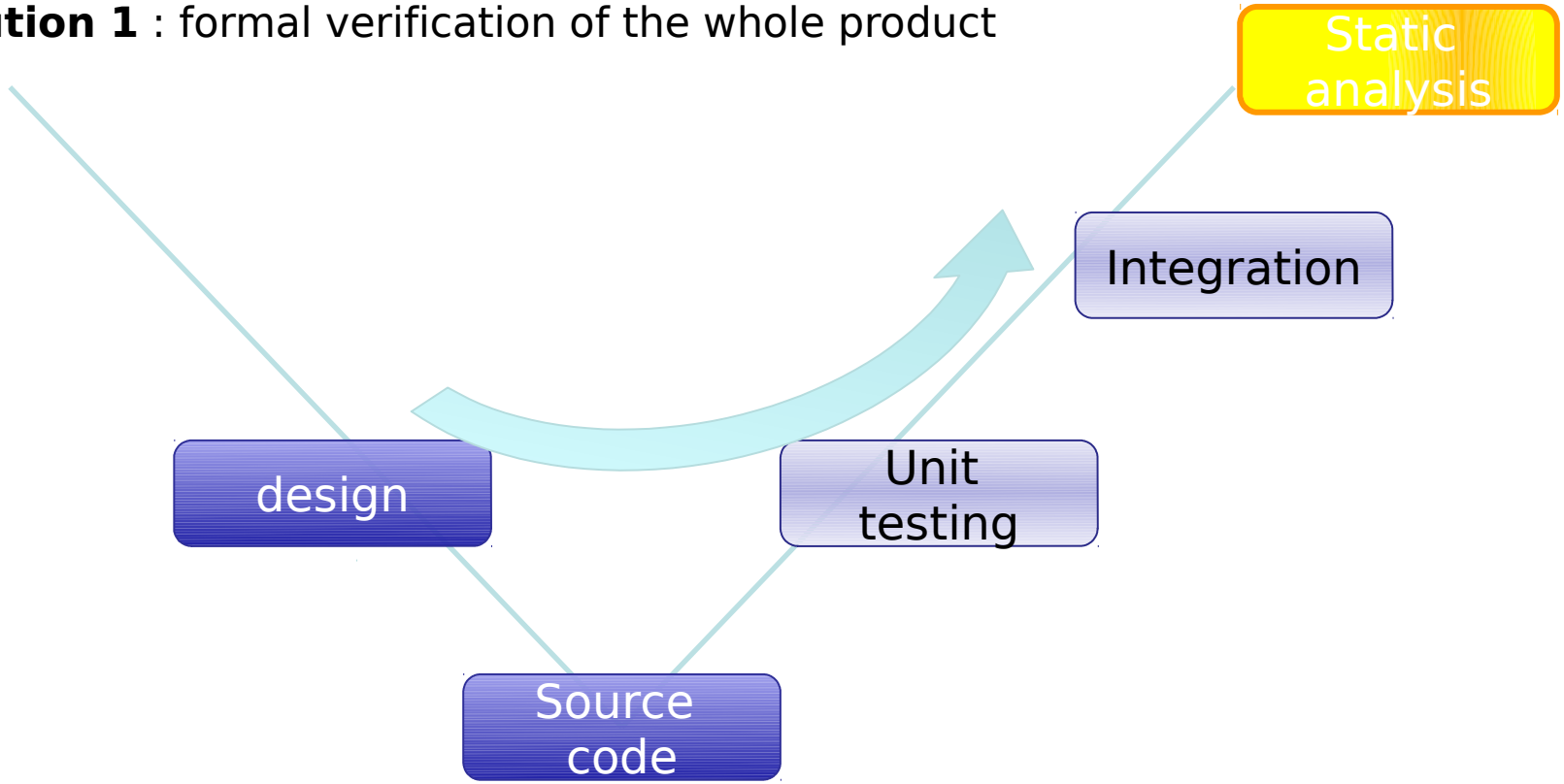
  -

```
int tab2[4] = {2,4,1,8};
int f_out_of_bound(int a)
{
  int index = a&7;

  /*@ assert (0 ≤ index) ∧ (index < 4);
        // synthesized
     */
  return tab2[index];
}
```

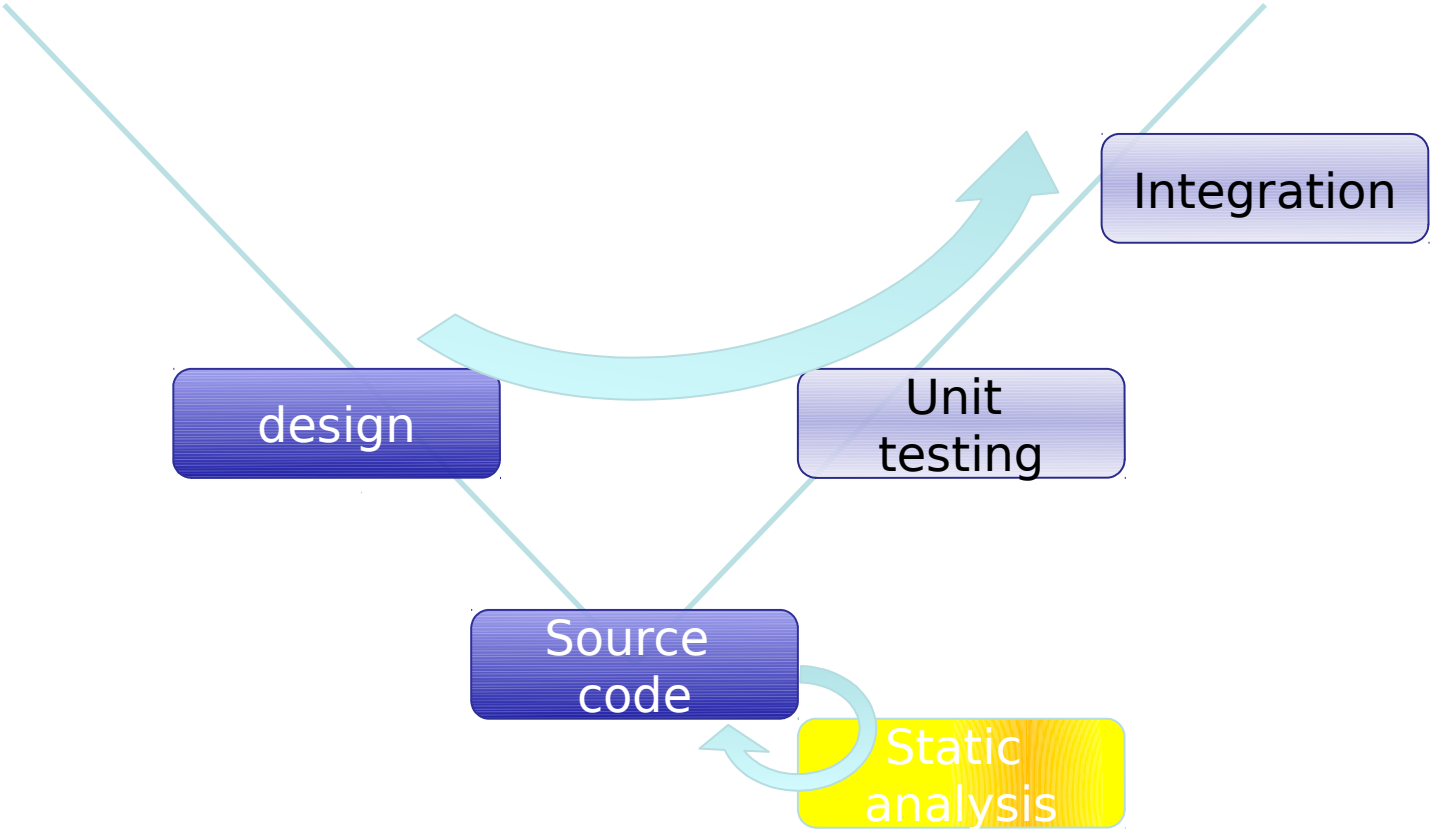
Use Case 1
Integration in the development process

- Process integration
  - **Solution 1** : formal verification of the whole product



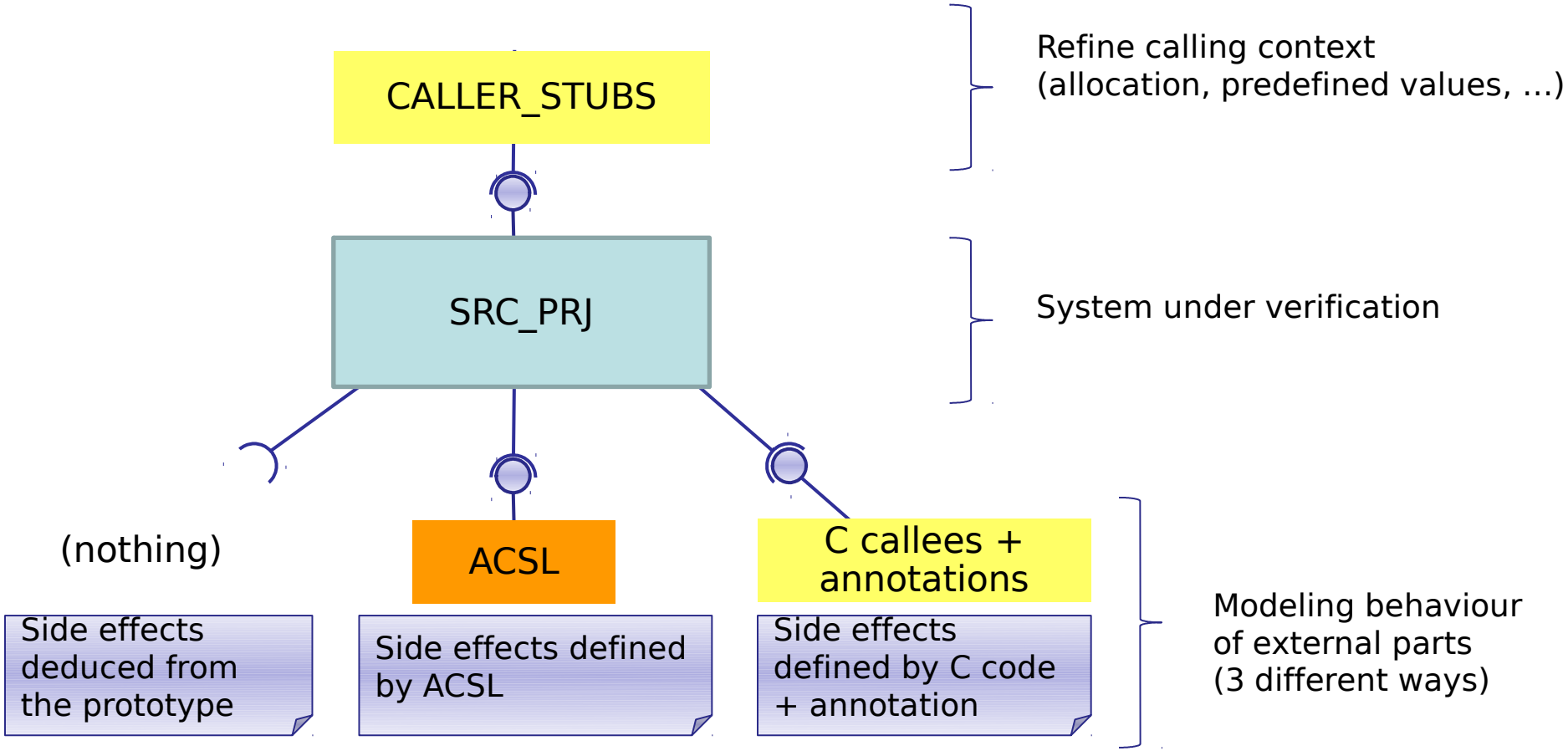
Use Case 1
Integration in the development process

- Process integration
  - **Solution 2** : formal verification integrated in the coding phase



Use Case 1
Technical overview

- Topology of a verification project



Use Case 1
Technical overview

- 2 different strategies

Top-Down

- Faster in case of few RTE end without problem of precision
- But difficult on complex programs
- Not adapted to multithread programs

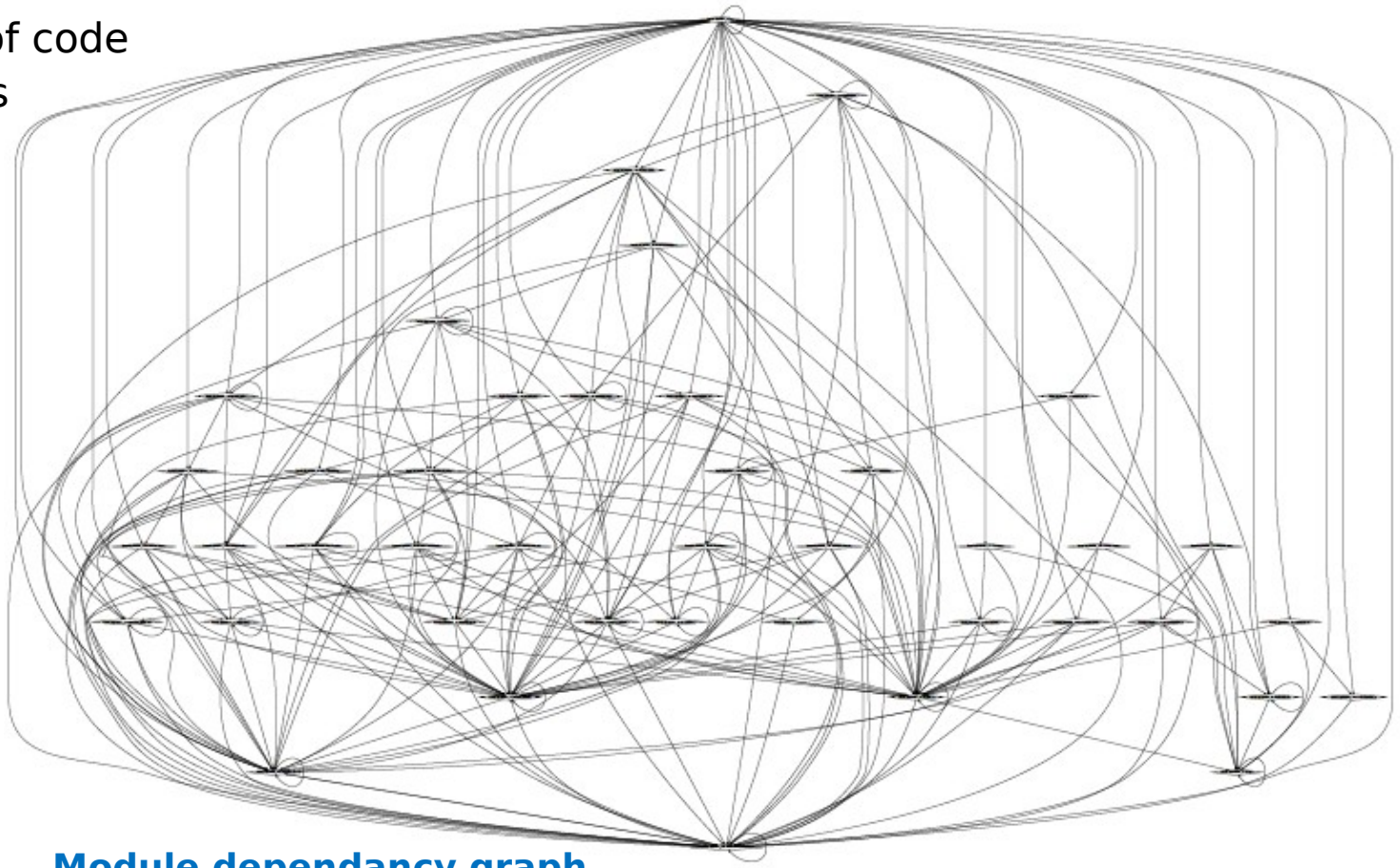
Bottom-Up

- Steady progress of the analysis
- Suitable for Safety verification intra-component (good use of C assertion in the code)
- Enabling verification on partial developments

Use Case 1
Technical overview

28/06/2013
Stephane Duprat

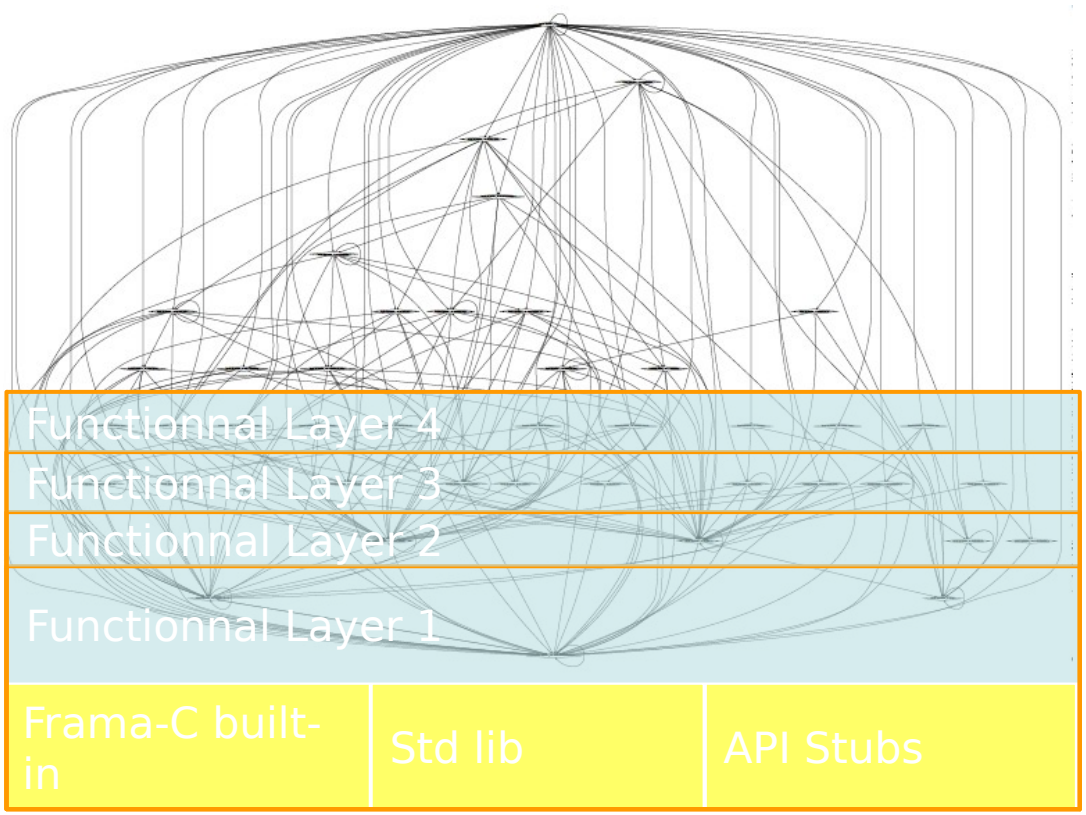
- Software:
  - 17 k lines of code
  - 42 modules



Module dependancy graph

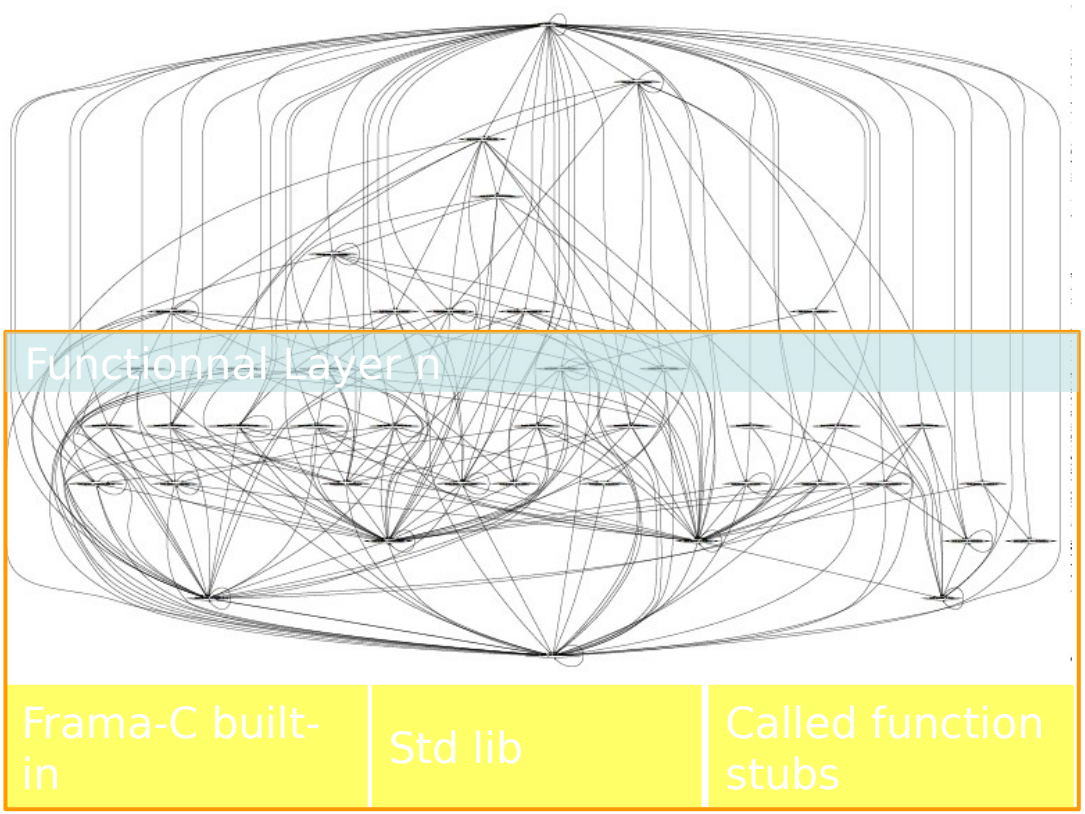
Use Case 1
Technical overview

- **The Bottom-Up strategy**
  - Starting with first layer analysis
    - Source code of first low level layer
    - + Frama-C builtins
    - + Standard lib stubs (ANSI standard library + specific ioctl commands)
    - + API stubs
  - Next layer analysis
    - Scope of previous layers
    - + new layer



Use Case 1
Technical overview

- Cutting projet
  - When ?
    - If whole project is too heavy
    - To analyse subproject independently
  - How ?
    - Taking functionnal layer
    - + Frama-C builtin
    - + stubs of called functions



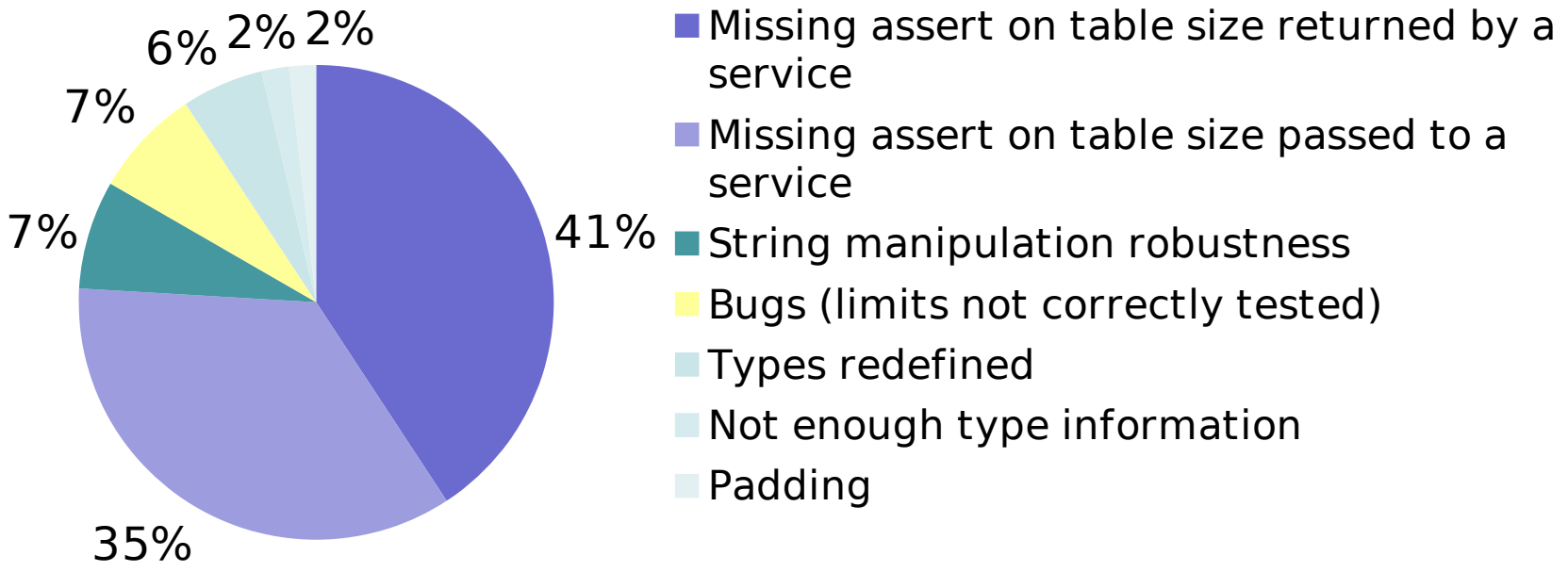
Use Case 1
Team organization

- Pros and cons of different team organizations

		Actors	
		Development team	Specialists
Process integration	Integrated in coding	Well synchronized Need some developers specialized in static analysis	The more efficient Need synchronization between actors
	Post-development	Not efficient	Efficient, but delayed result. Costs an extra verification phase

Use Case 1
Return of experience

- Issues detected



Use Case 1
Return of experience

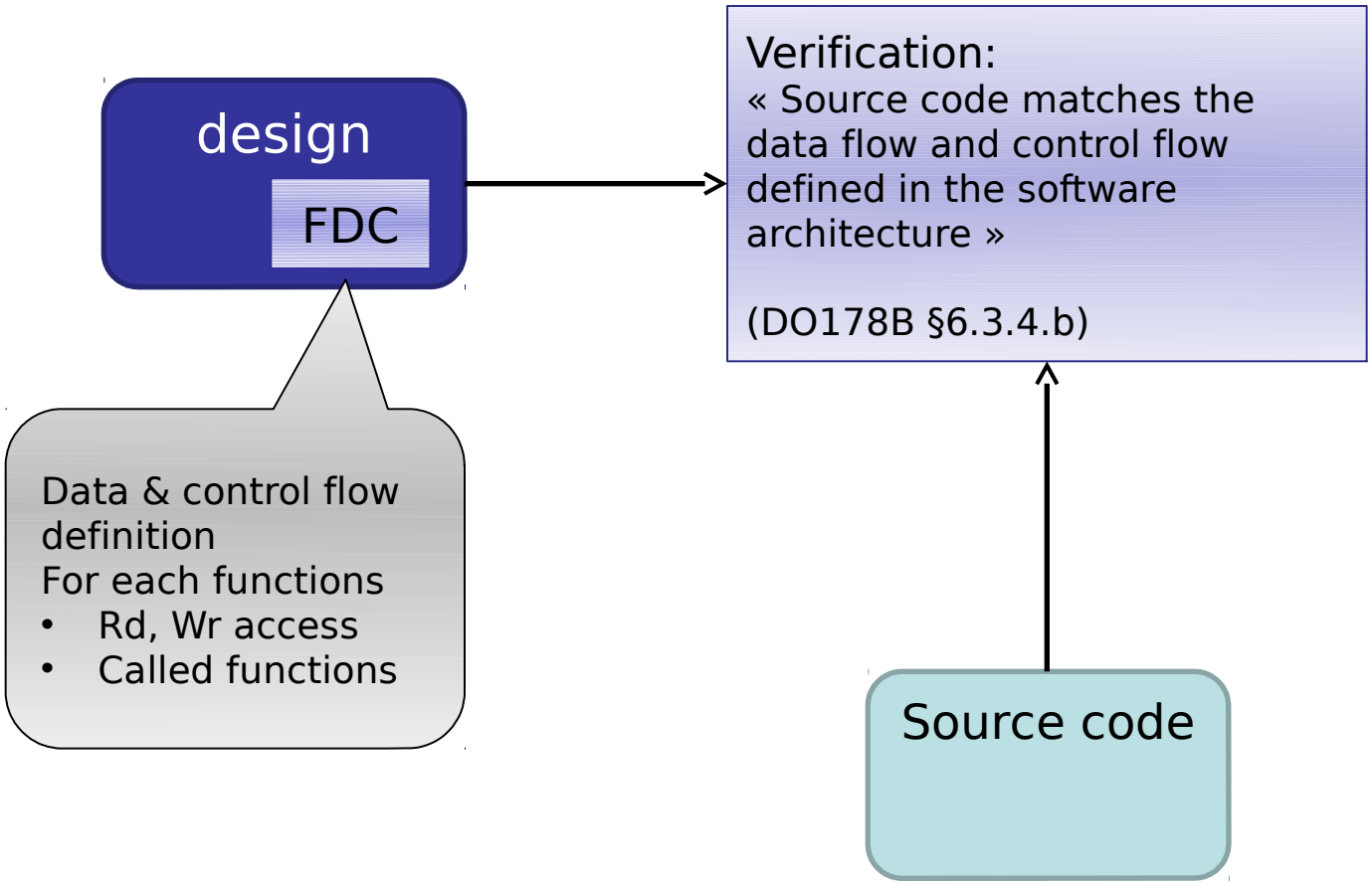
- Impact on costs
  - Quick improvement of maturity, reduce costs of tests
  - Linear ratios for project set-up in bottom-up
  - Reduce costs on defects on global lifecycle of the product
- Risk control
  - on independent development, on COTS, TMA
- Impact on organization
  - Secure development between
    - Different entities of skill in one company
    - Between customer and provider
  - Quality engagement objectively verifiable between customer and provider
  - Enhance skills of developers

Use Case 2 –
Industrial development of Frama-C plugin based on semantic analysis

28/06/2013

Use Case 2
Defining the verification needs

- Data & Control flow verification objectives

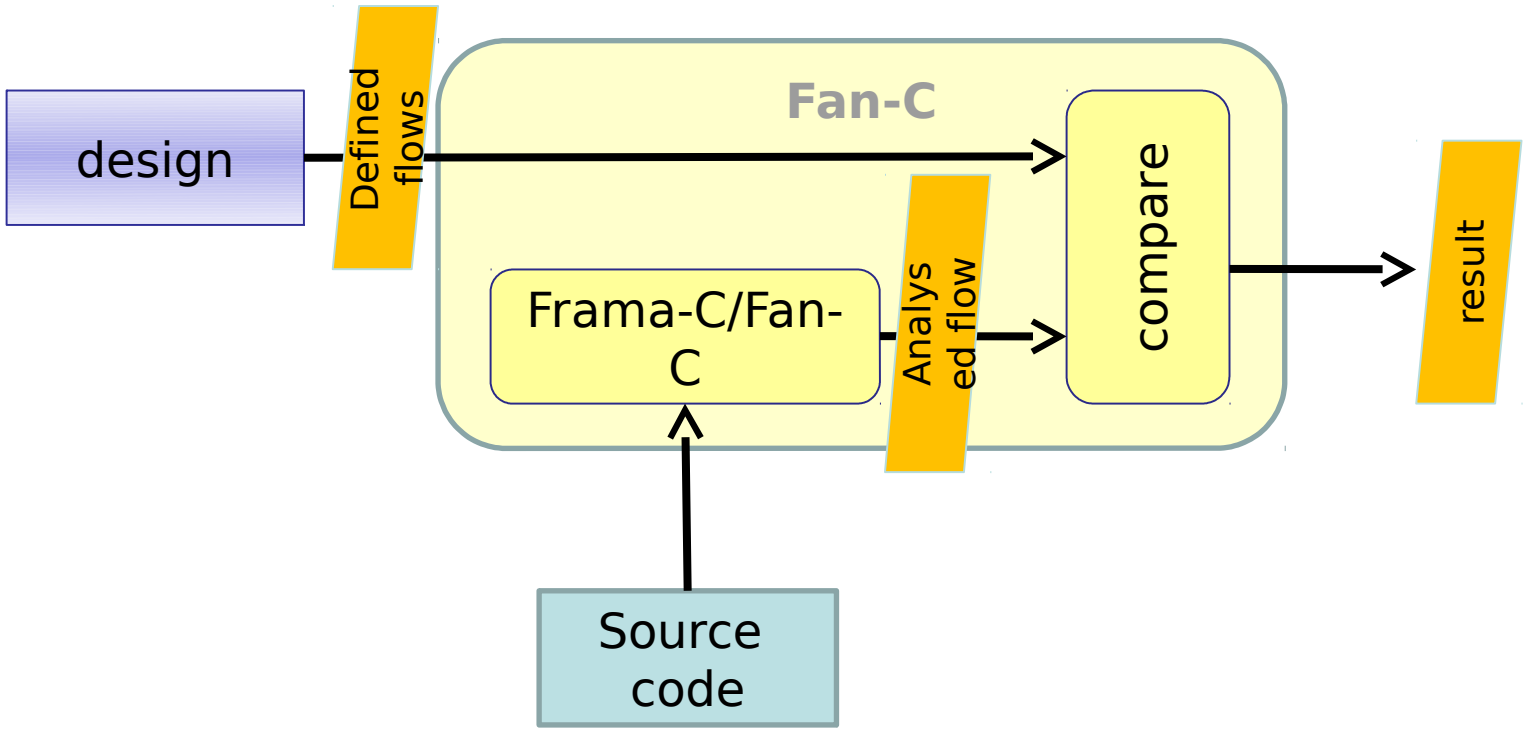


Use Case 2
Defining the verification needs

- Industrial context
  - Verification required by the process
  - Verification traditionally performed by code review
- Objective = automate the verification by a tool
- Challenges of the automated verification
  - Fit with Control and data flow conventions previously applied
  - Be able to analyze correctly data & control flow in source code
    (make a syntactic solution inappropriate)

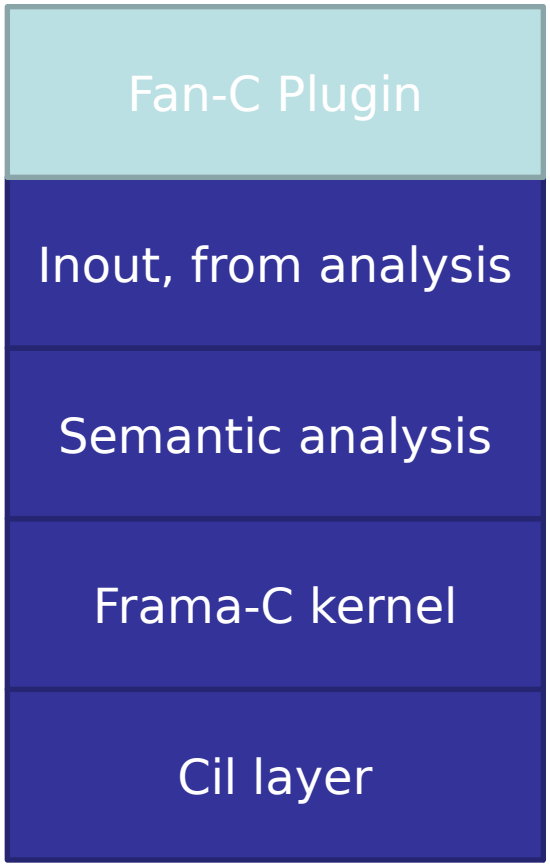
Use Case 2
Technical solution

- Input/output of the tool



Use Case 2
Technical solution

- Technological blocks



Generated ACSL, launch analysis, get results, consolidate and export in report

Developed part by the industrial

Analysis of locations in Rd, Wr

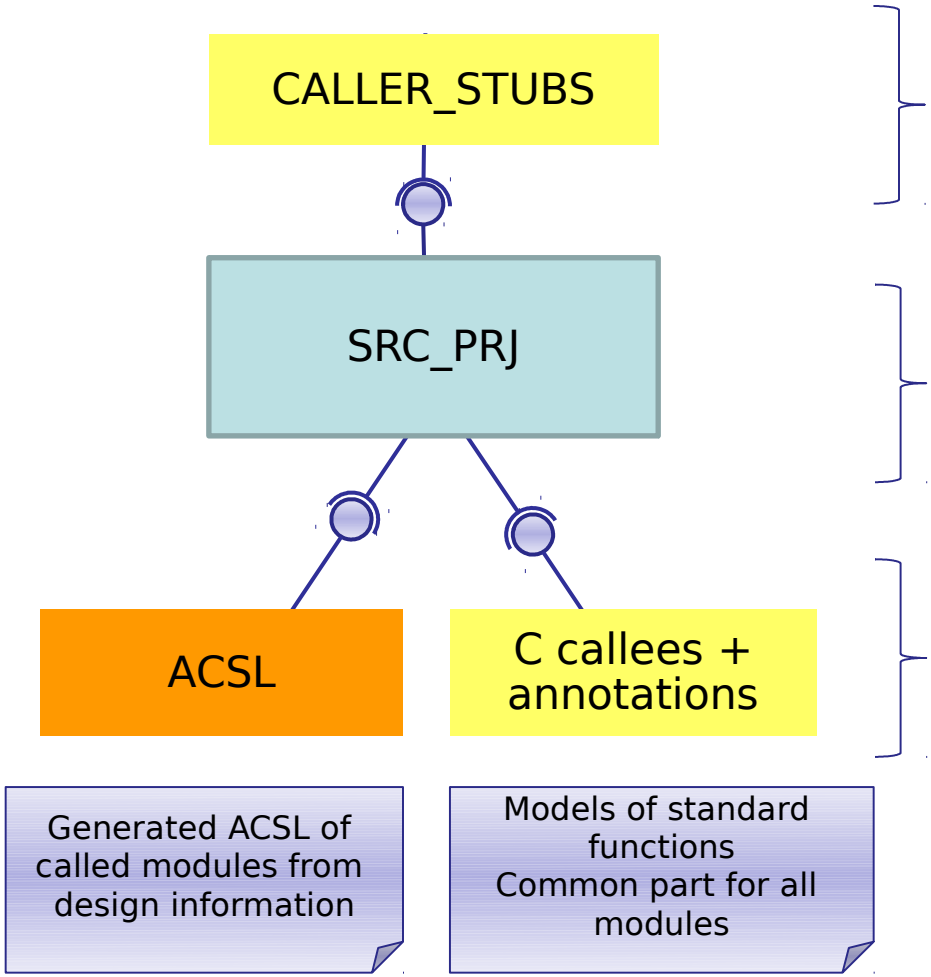
Make pointer resolution

Framework plugin

Standard framework

Syntactic layer

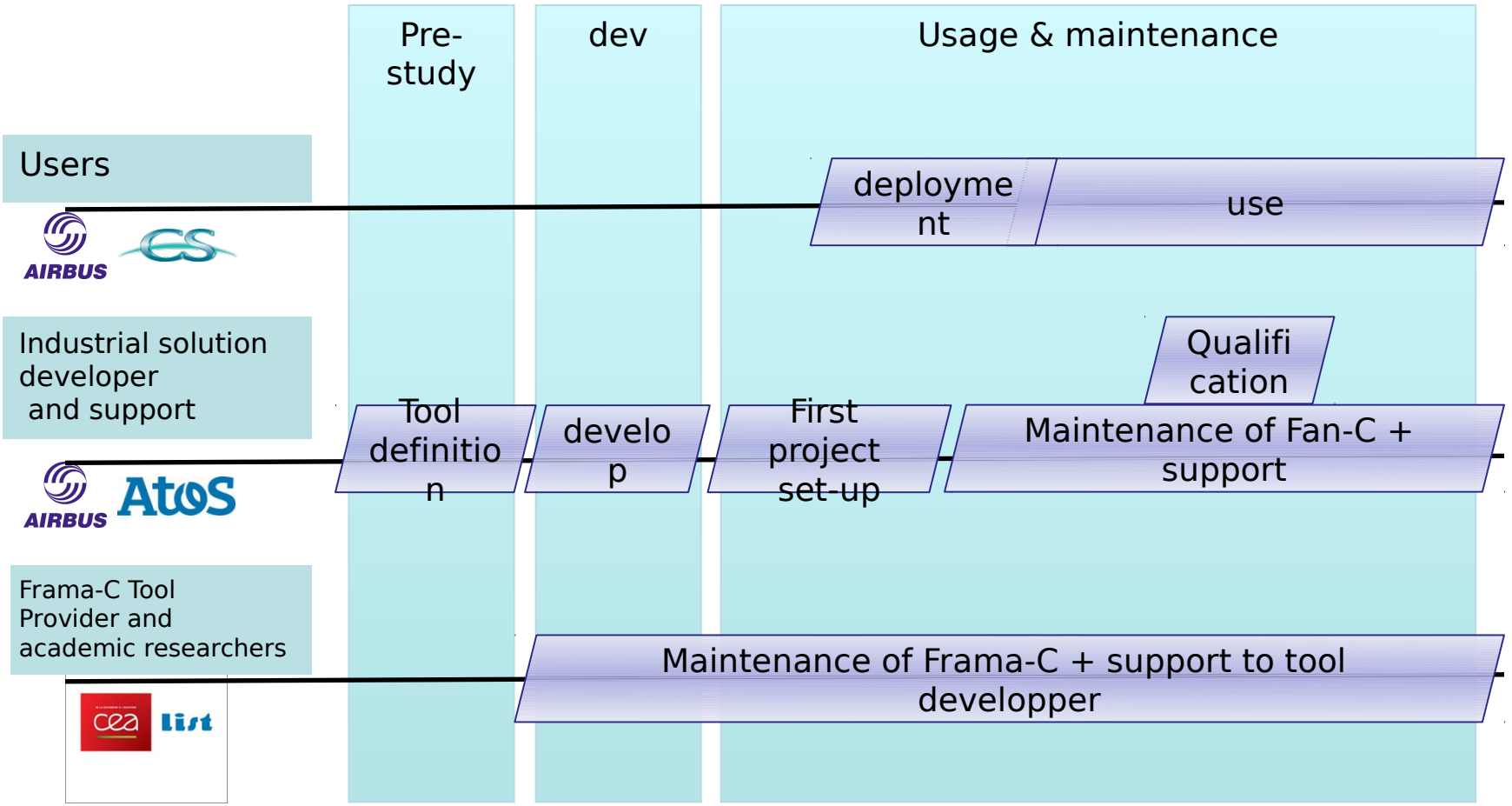
Use Case 2
Technical solution



Use Case 2
Organizational solution

28/06/2013
Stephane Duprat

- Actors and roles



Use Case 2
The industrial results

- Technical
  - High level of automaticity

- Users
  - Intellectual verification replaced by an automatic one « push button »
  - Bonus of static analysis
    - Detection of the use of pointer on local variables
    - Verifications on data alignments

Conclusion

28/06/2013

Global conclusion

- Static analysis adapted to multiple kind of verifications
- Static analysis
  - opening different methodologies,
  - enabling new ways of works
  - Quality assessment with high degree of confidence
- Tool mature enough for semantic analysis in industrial context
- Developing specific features of the tool is accessible for industrial actor (and not only researchers)
- Place of the specialists, techno providers

Glossary

- RTE: Run Time Error
- FDC: Flows Controls & Data