Specification of CBB Testing for Fractal

5th Fractal Workshop at ECOOP'2006

D. Deveaux - P. Collet

UBS/Lab. Valoria/ALCC team - UNSA/Lab. I3S/OCL team

daniel.deveaux@univ-ubs.fr philippe.collet@unice.fr

Ce travail est réalisé dans le cadre du projet SafeCode soutenu par l'ANR (ARA SSIA 2006-2009)

Agenda

- Goal
- Background
- Specification of the Framework
  - Principles
  - Implementation in Fractal
- Future Work

Goal: a Testing Framework for Fractal

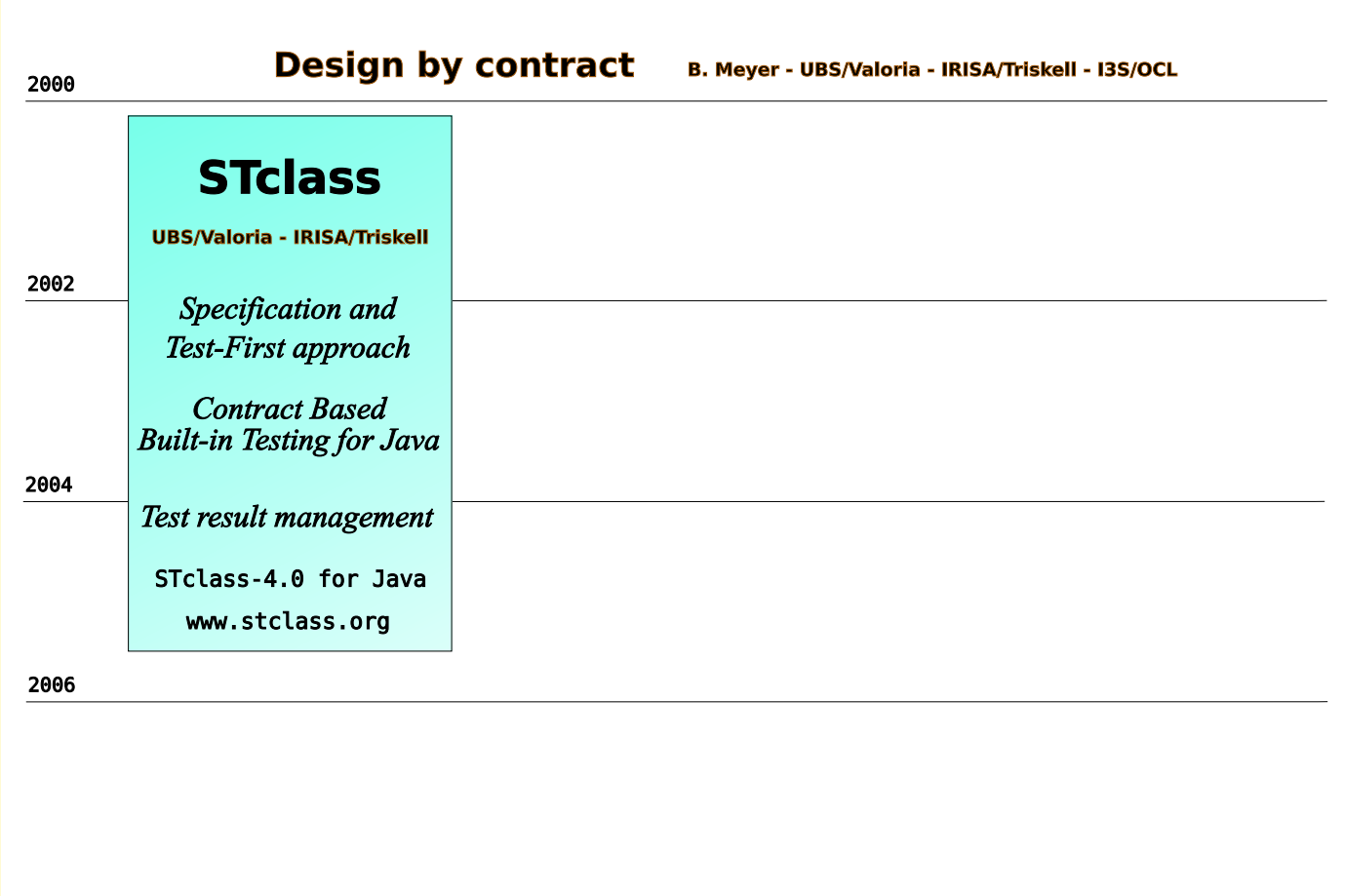
For several years, Contract Based Testing (CBT) is considered as one of the best testing techniques for OO-software [Binder96].

For components, contracts are at the center of many studies and several CBT proposals have be made recently:

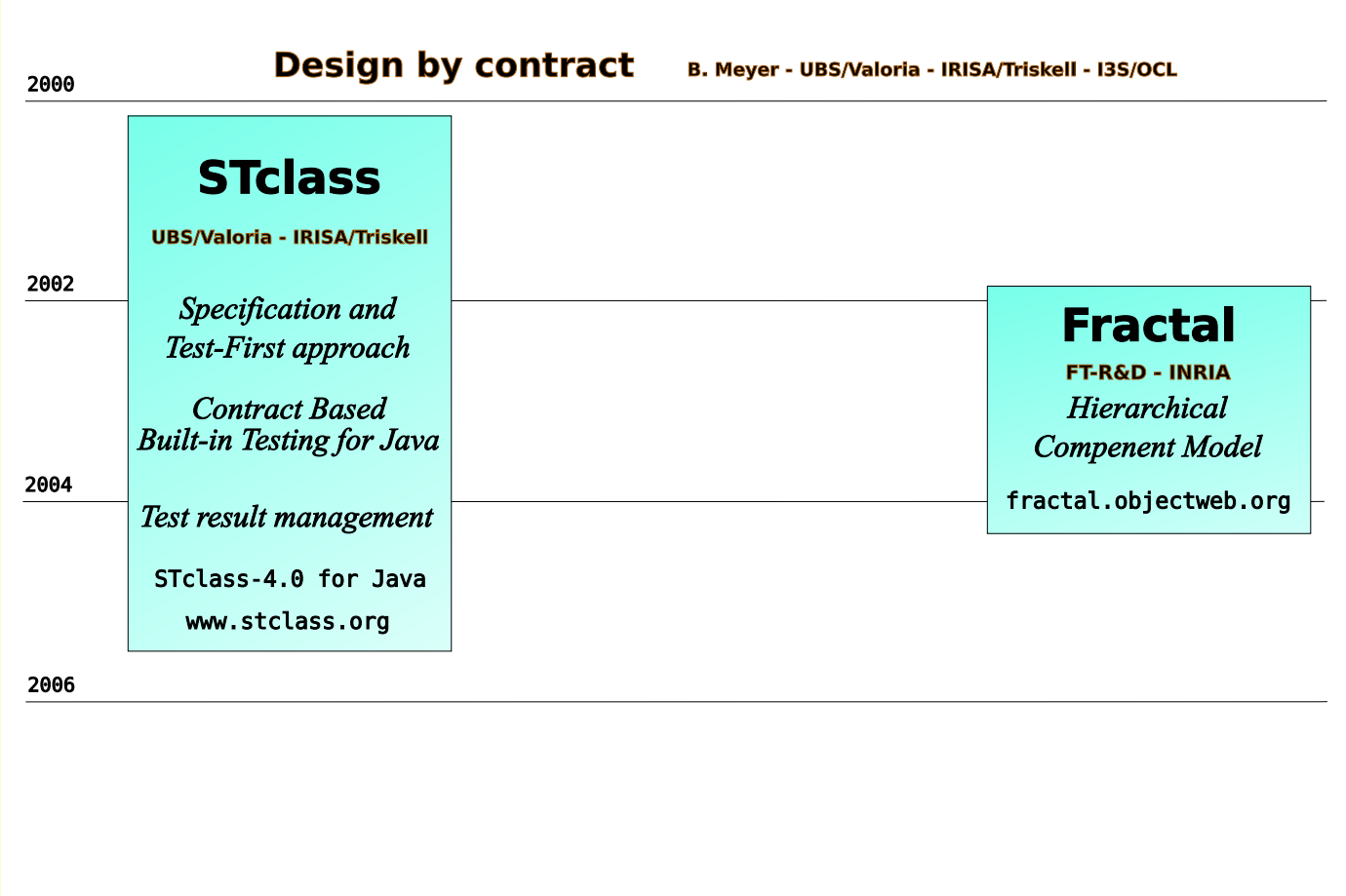
- methodological approaches [Gross05],
- practical frameworks [Valentini.ea05]

Such a framework has not been proposed for the Fractal component model.

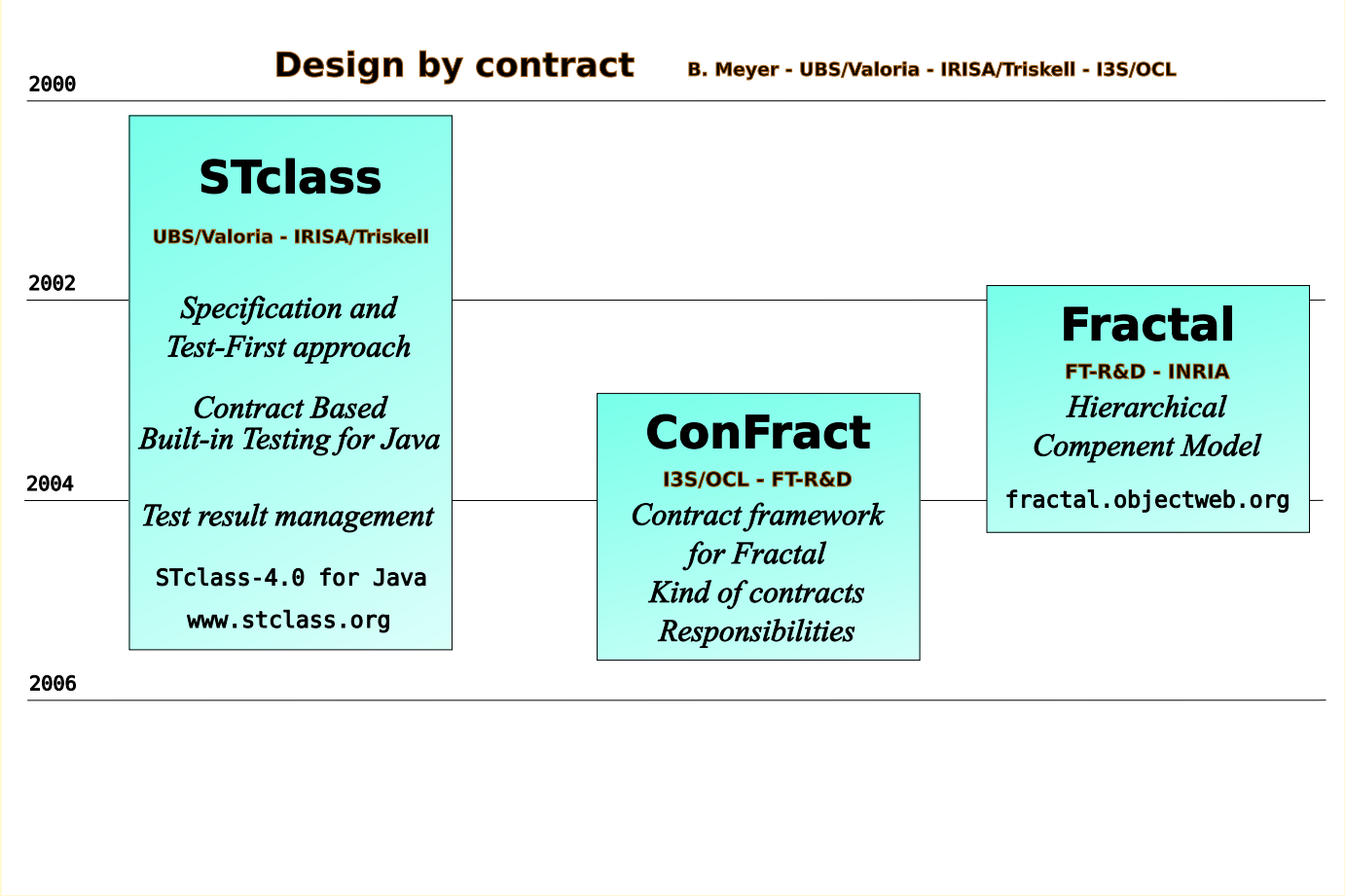
Background



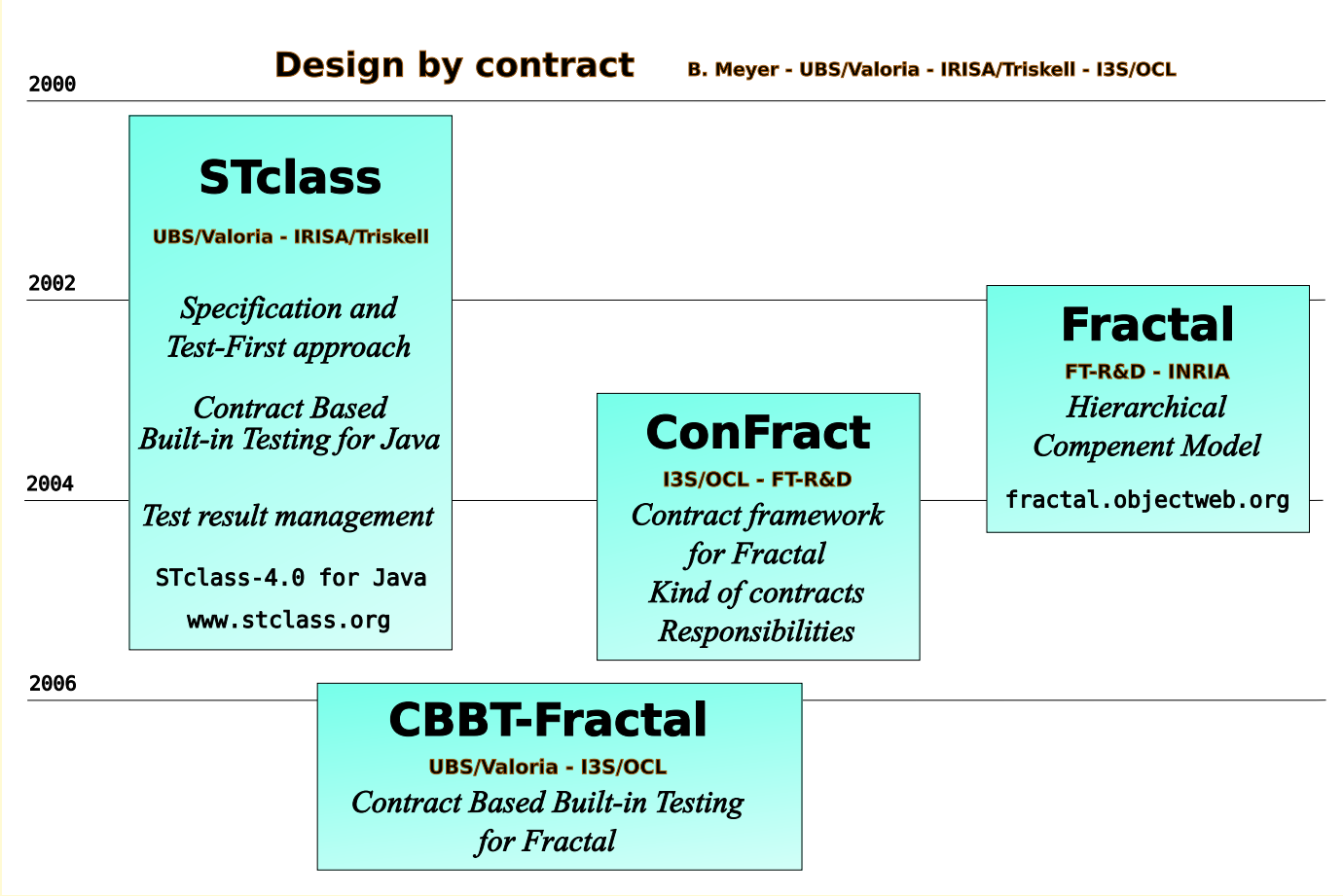
Background



Background



Background



STclass contributions

- **STclass** (2000-05 D.Deveaux, Y.Le Traon, JM. Jézéquel) supports a *Specification and Test first* approach,

STclass contributions

- **STclass** (2000-05 D.Deveaux, Y.Le Traon, JM. Jézéquel) supports a *Specification and Test first* approach,
- *Contract Based Built-in testing* at class level with contracts and tests inheritance for the Java language,
  - Postconditions and invariant make good and salient oracles.
  - Testing code consists only in simple method calls (scenario description)
  - Preconditions limit the scope of the test.

STclass contributions

- **STclass** (2000-05 D.Deveaux, Y.Le Traon, JM. Jézéquel) supports a *Specification and Test first* approach,
- *Contract Based Built-in testing* at class level with contracts and tests inheritance for the Java language,
  - Postconditions and invariant make good and salient oracles.
  - Testing code consists only in simple method calls (scenario description)
  - Preconditions limit the scope of the test.
- Why CBB-Testing rather than JUnit-Testing?
  - *separation of concerns*: **functional specification** in contracts, **dynamic specification** in senarii;
  - *better documentation;*
  - *modeling approach* rather than coding approach.

ConFract: Mastering Complexity

- **ConFract** (P. Collet, R. Rousseau) is a contracting system for the Fractal component platform

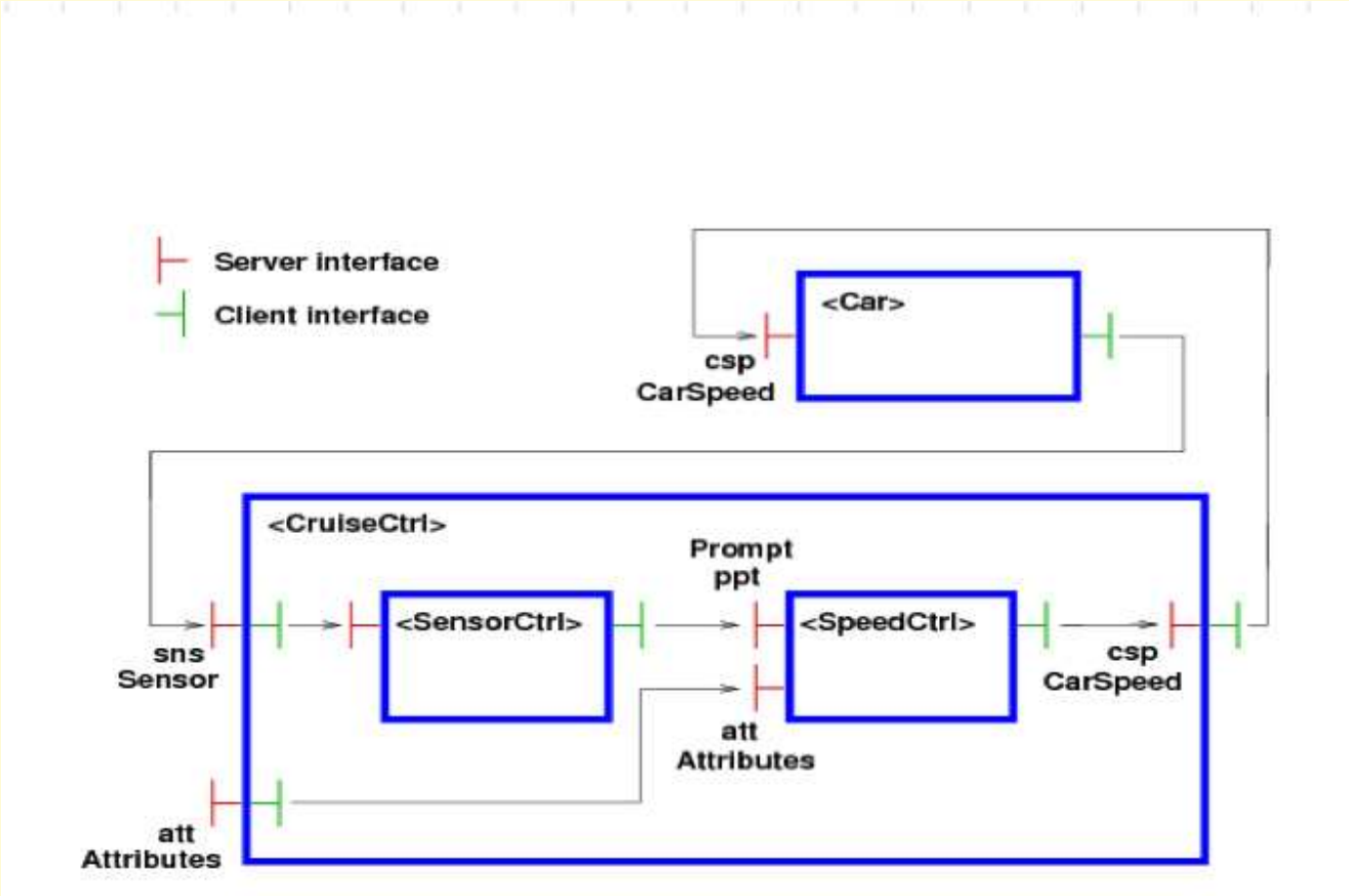
ConFract: Mastering Complexity

- **ConFract** (P. Collet, R. Rousseau) is a contracting system for the Fractal component platform
- **Usually, contracts are either**
  - *implicit* between software artefacts or,
  - some *interpretationof a specification* (responsibility, blame),

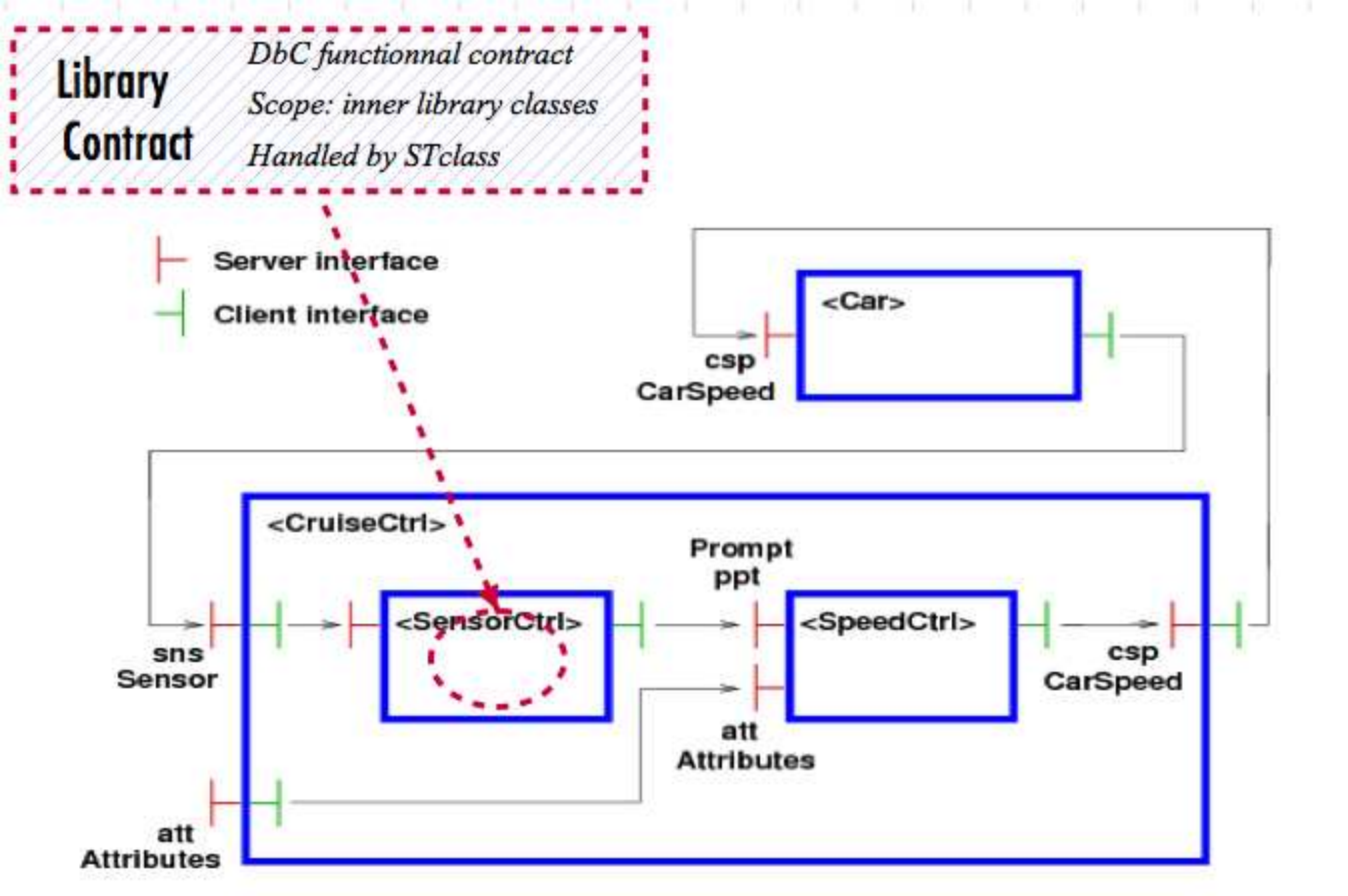
ConFract: Mastering Complexity

- **ConFract** (P. Collet, R. Rousseau) is a contracting system for the Fractal component platform
- **Usually, contracts are either**
  - *implicit* between software artefacts or,
  - some *interpretationof a specification* (responsibility, blame),
- **Confract: adapting programming by contract to Fractal**
  - Contracts are first class objects
  - Types of contracts (see next slide)
  - Responsibilities
    - components are participants in contracts
    - Each participant has a well-defined responsibility, guarantor and beneficiary

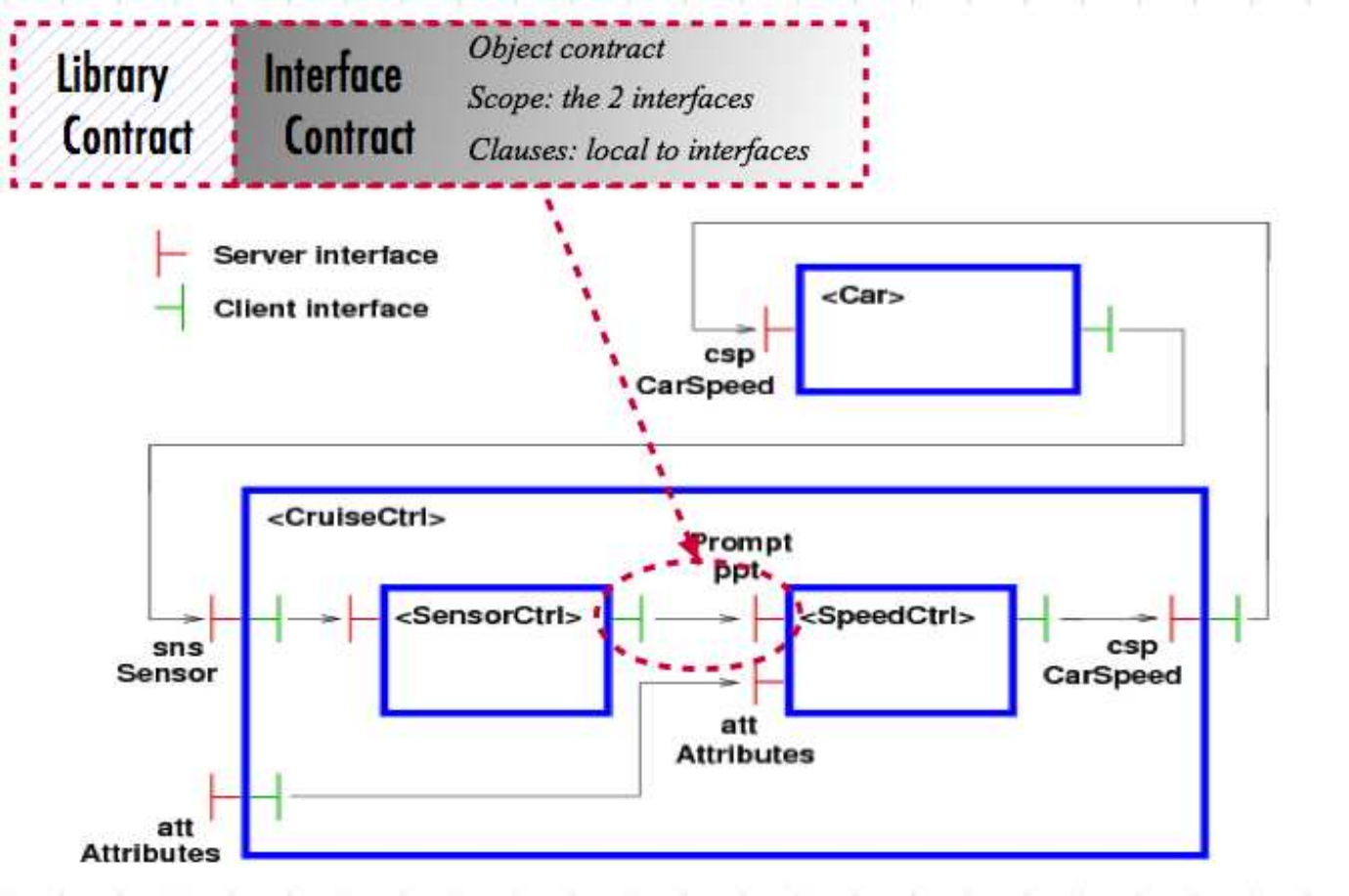
Types of Contracts



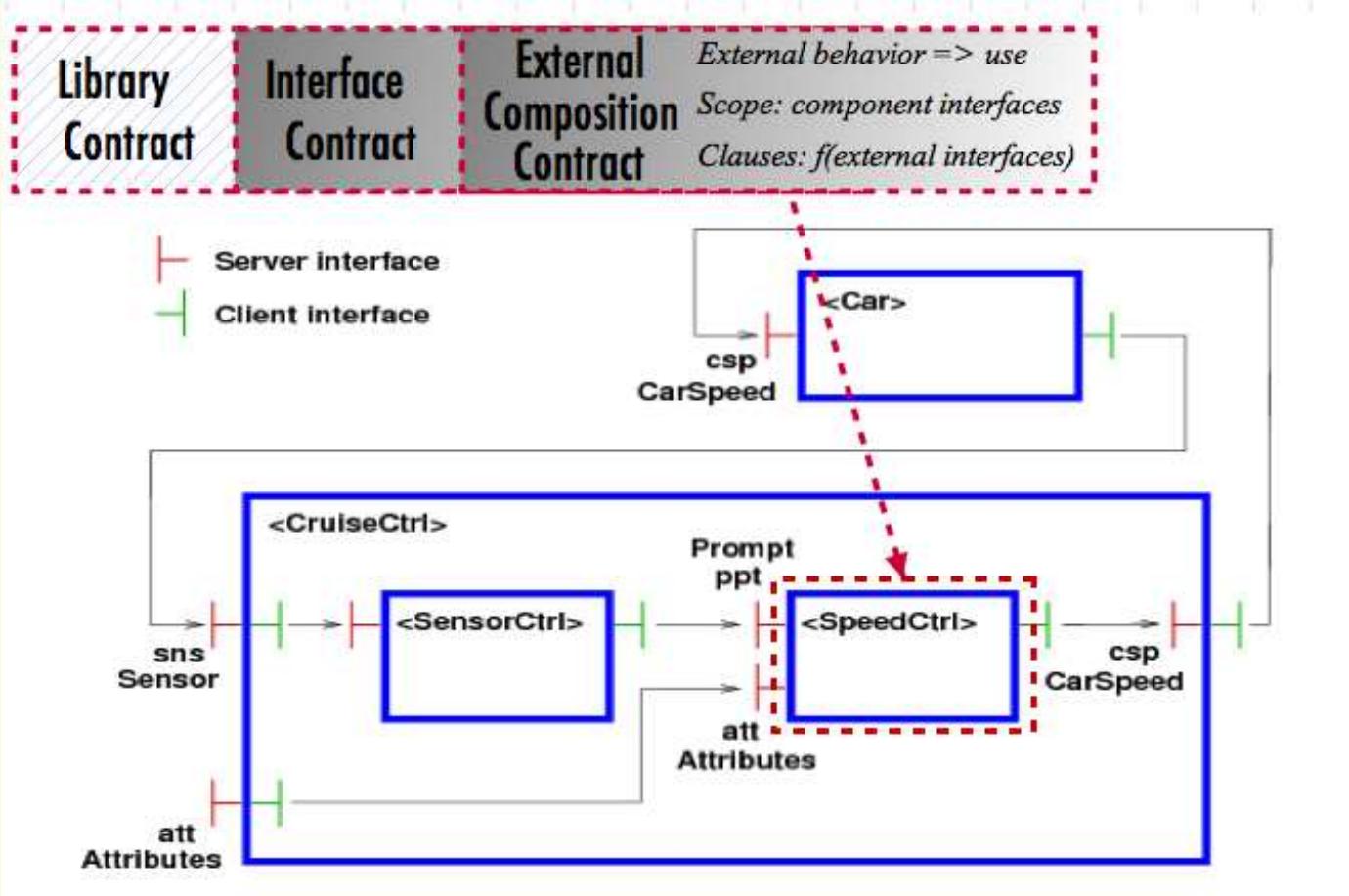
Types of Contracts



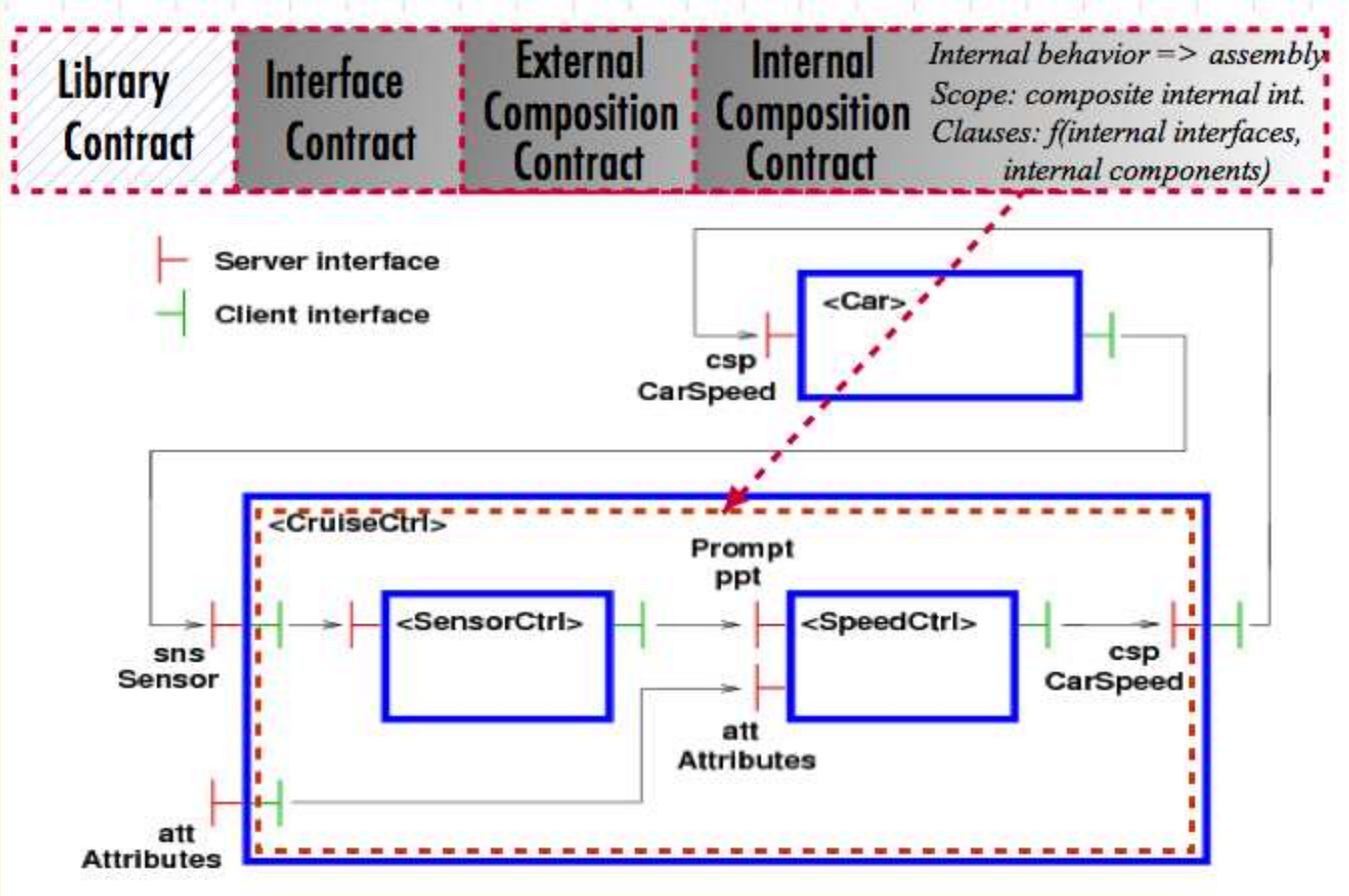
Types of Contracts



Types of Contracts



Types of Contracts



CBBT Framework Principles

- ***Writing tests only relies on basic concepts***:
  - testers should only use components, small Java (or other implementation language) and ADL code to define and run tests;
  - the framework manages all the complexity associated to test automation.

CBBT Framework Principles

- ***Writing tests only relies on basic concepts***:
  - testers should only use components, small Java (or other implementation language) and ADL code to define and run tests;
  - the framework manages all the complexity associated to test automation.

- ***Testing is Contract-Based***: see next slides

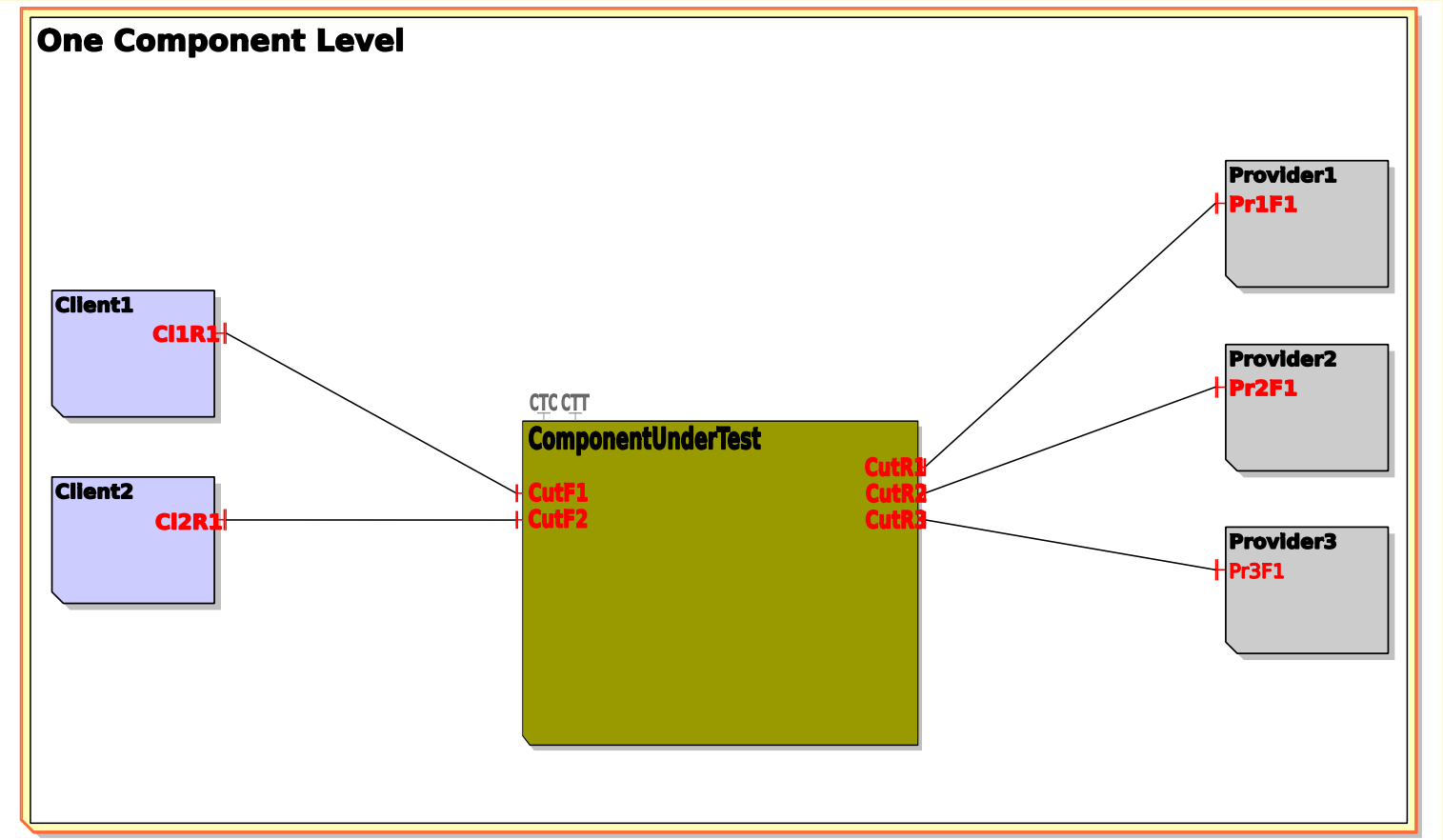
CBBT Framework Principles

- ***Writing tests only relies on basic concepts***:
  - testers should only use components, small Java (or other implementation language) and ADL code to define and run tests;
  - the framework manages all the complexity associated to test automation.
- ***Testing is Contract-Based***: see next slides
- ***Tests are built-in:***
  - each component contains its own testing information,
  - a test controller generates a test bed that surrounds the CUT.

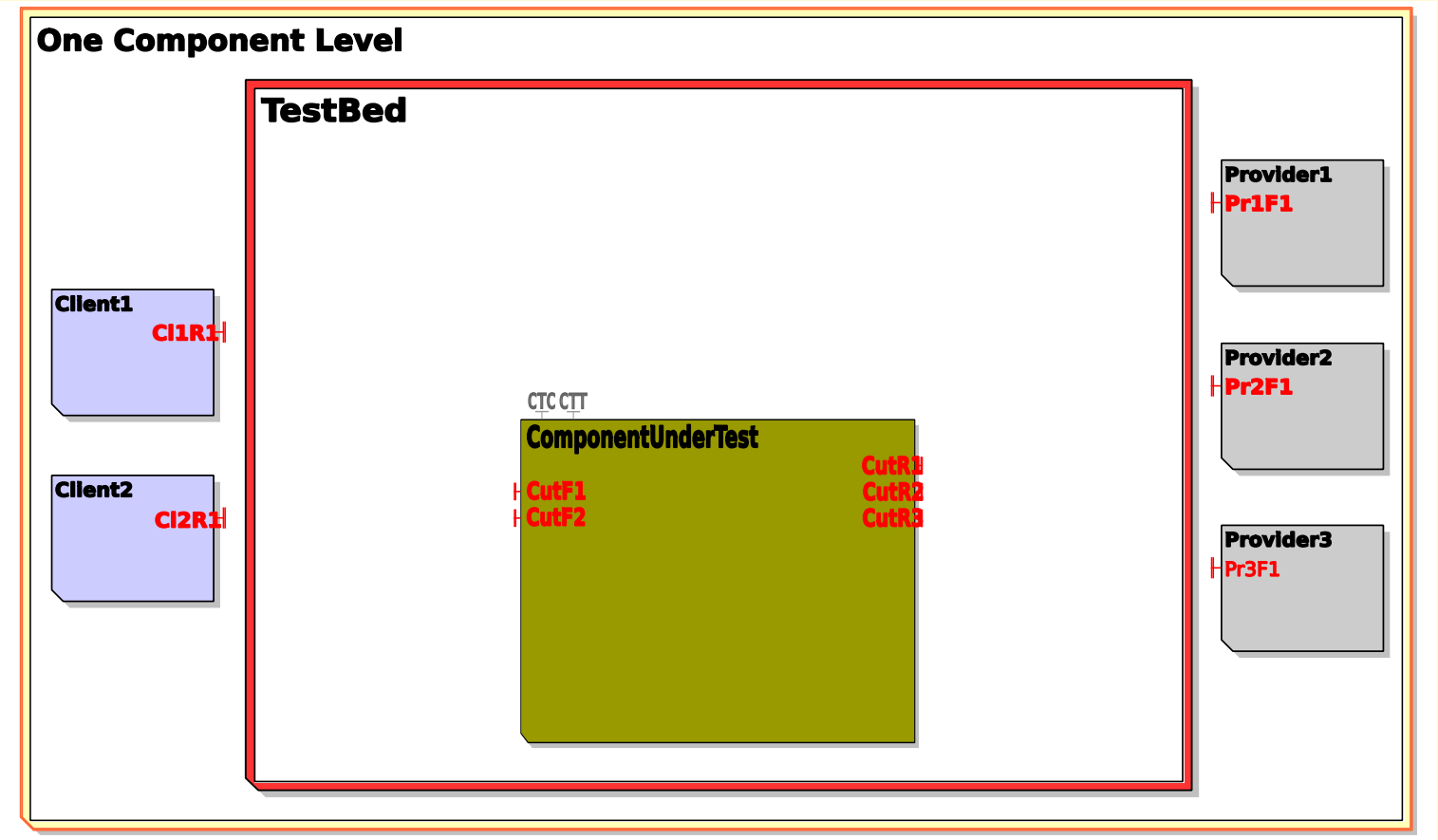
CBBT Framework Principles

- ***Writing tests only relies on basic concepts***:
  - testers should only use components, small Java (or other implementation language) and ADL code to define and run tests;
  - the framework manages all the complexity associated to test automation.
- ***Testing is Contract-Based***: see next slides
- ***Tests are built-in:***
  - each component contains its own testing information,
  - a test controller generates a test bed that surrounds the CUT.
- ***The Framework is Pure Fractal***: adaptation to different contracts models or different implementations

Built-in Test: a Dynamic TestBed



Built-in Test: a Dynamic TestBed



Contract Based Testing

ConFract responsibility model is reused and different kinds of test can be provided:

- Isolated testing or in situ testing,
- Black-box unit testing (TestUnit, TestCase, TestSuite)
- Gray-box testing
- Admission testing for the providers
- Test Reports

TestUnit, TestCase, TestSuite

- **TestUnit**: a scenario; cannot be executed out of a TestCase but can participate to several TestCases

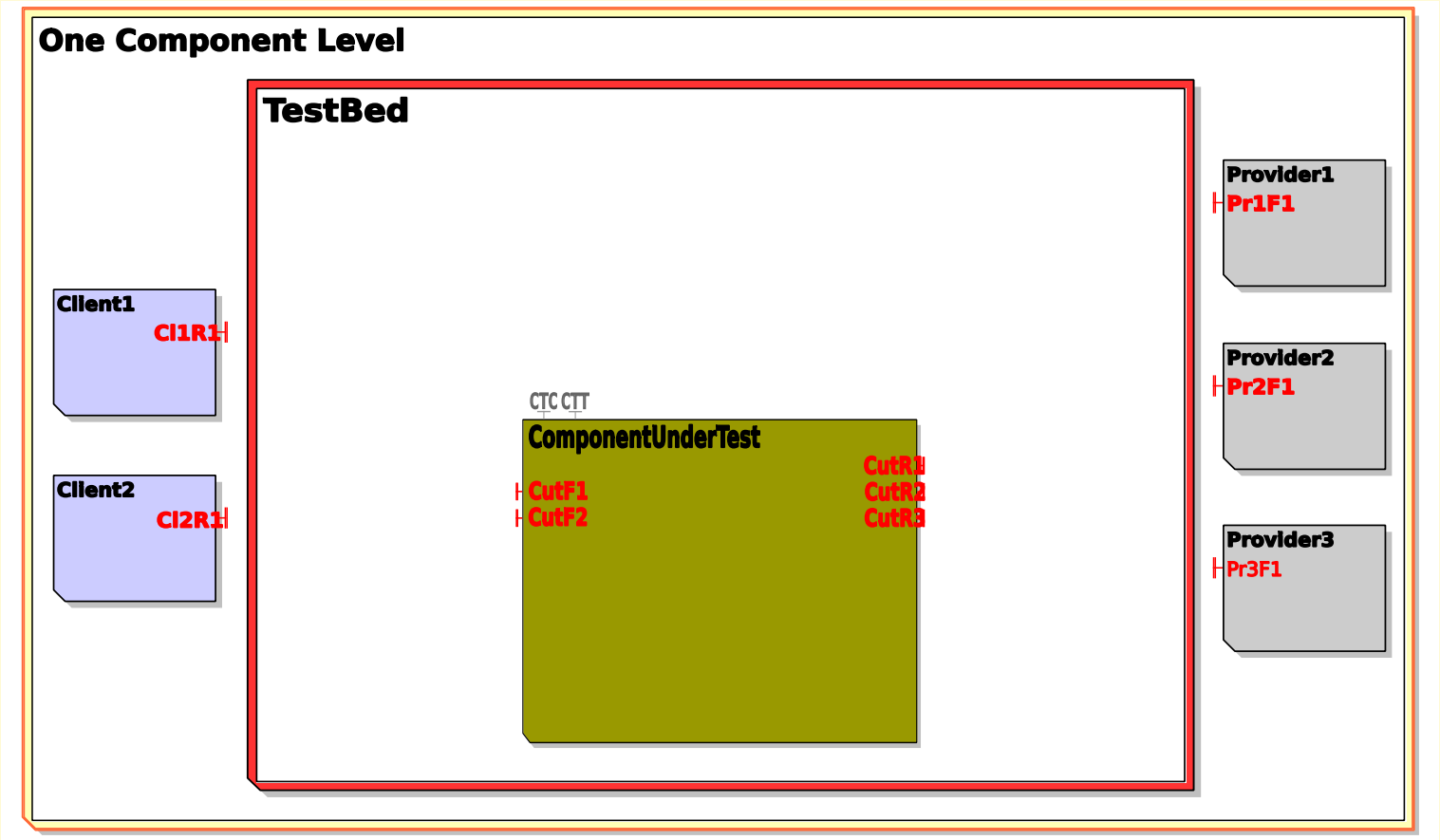
TestUnit, TestCase, TestSuite

- **TestUnit**: a scenario; cannot be executed out of a TestCase but can participate to several TestCases
- **TestCase**: environment of TestUnits
  1. parameters for testing stubs
  2. setup action
  3. list of TestUnits
  4. teardown action

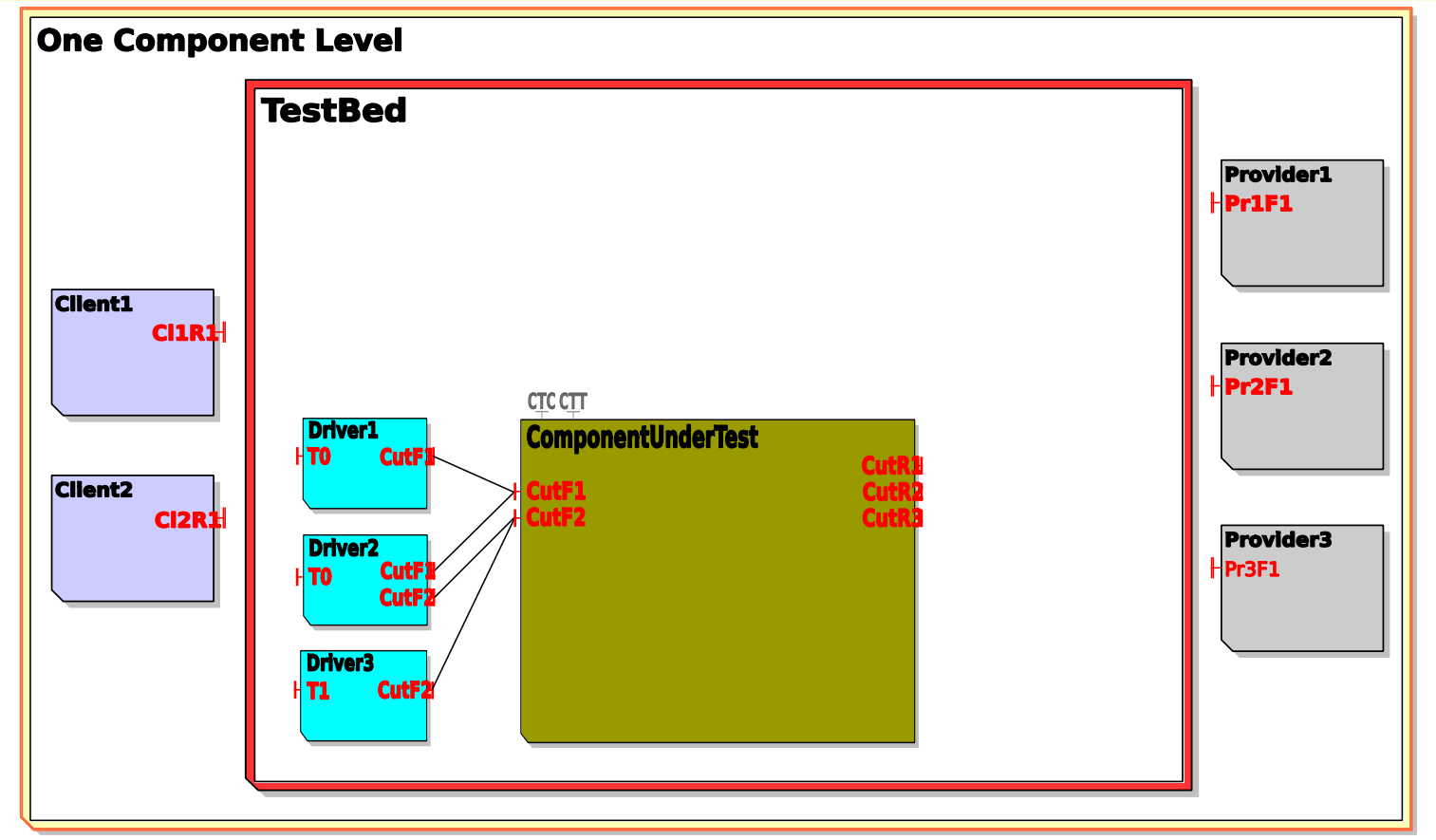
TestUnit, TestCase, TestSuite

- **TestUnit**: a scenario; cannot be executed out of a TestCase but can participate to several TestCases
- **TestCase**: environment of TestUnits
  1. parameters for testing stubs
  2. setup action
  3. list of TestUnits
  4. teardown action
- **TestSuite**: ordered list of TestCases, TestSuites or TestUnits to be activated (Composite pattern)

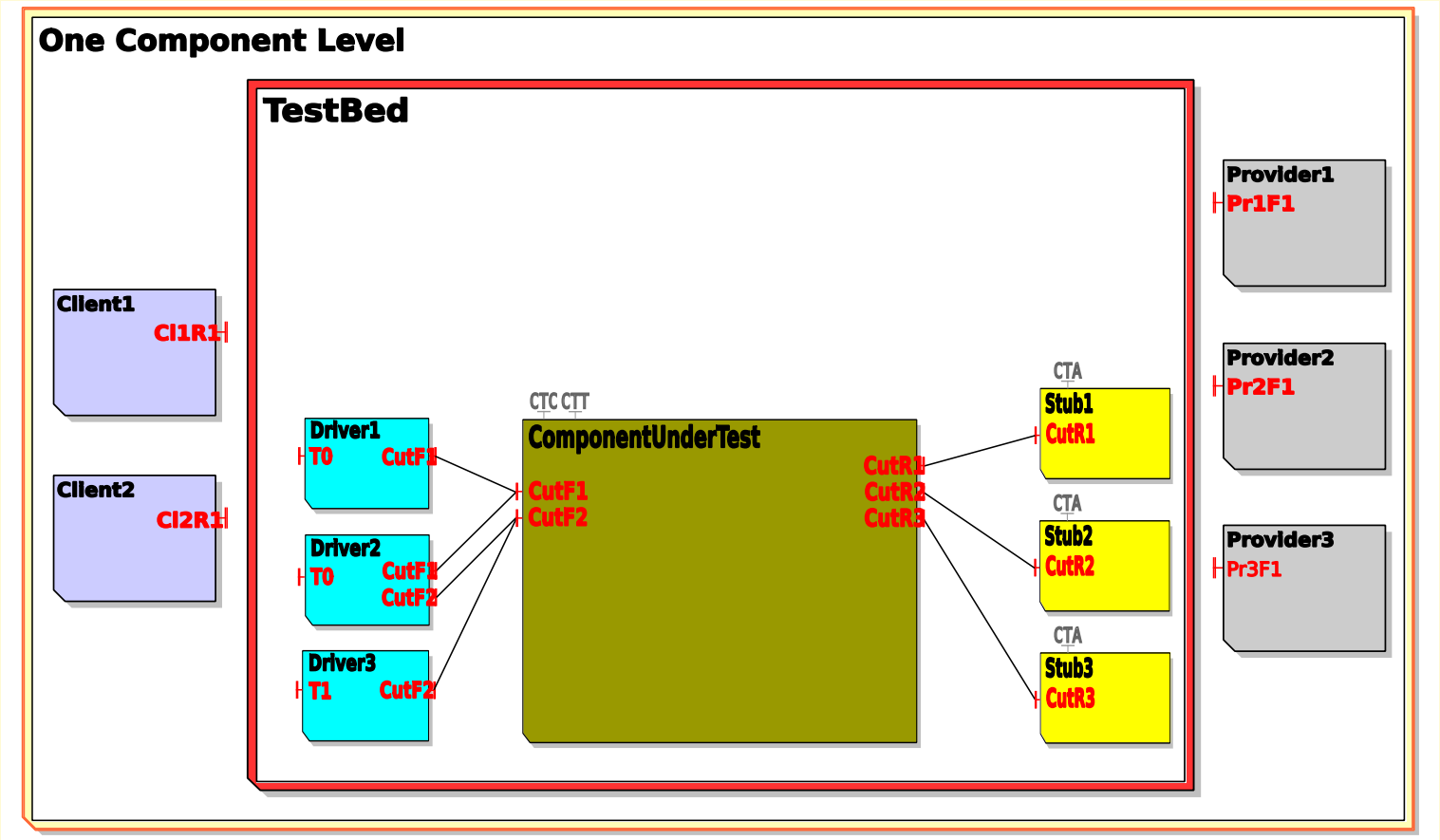
Populate the TestBed



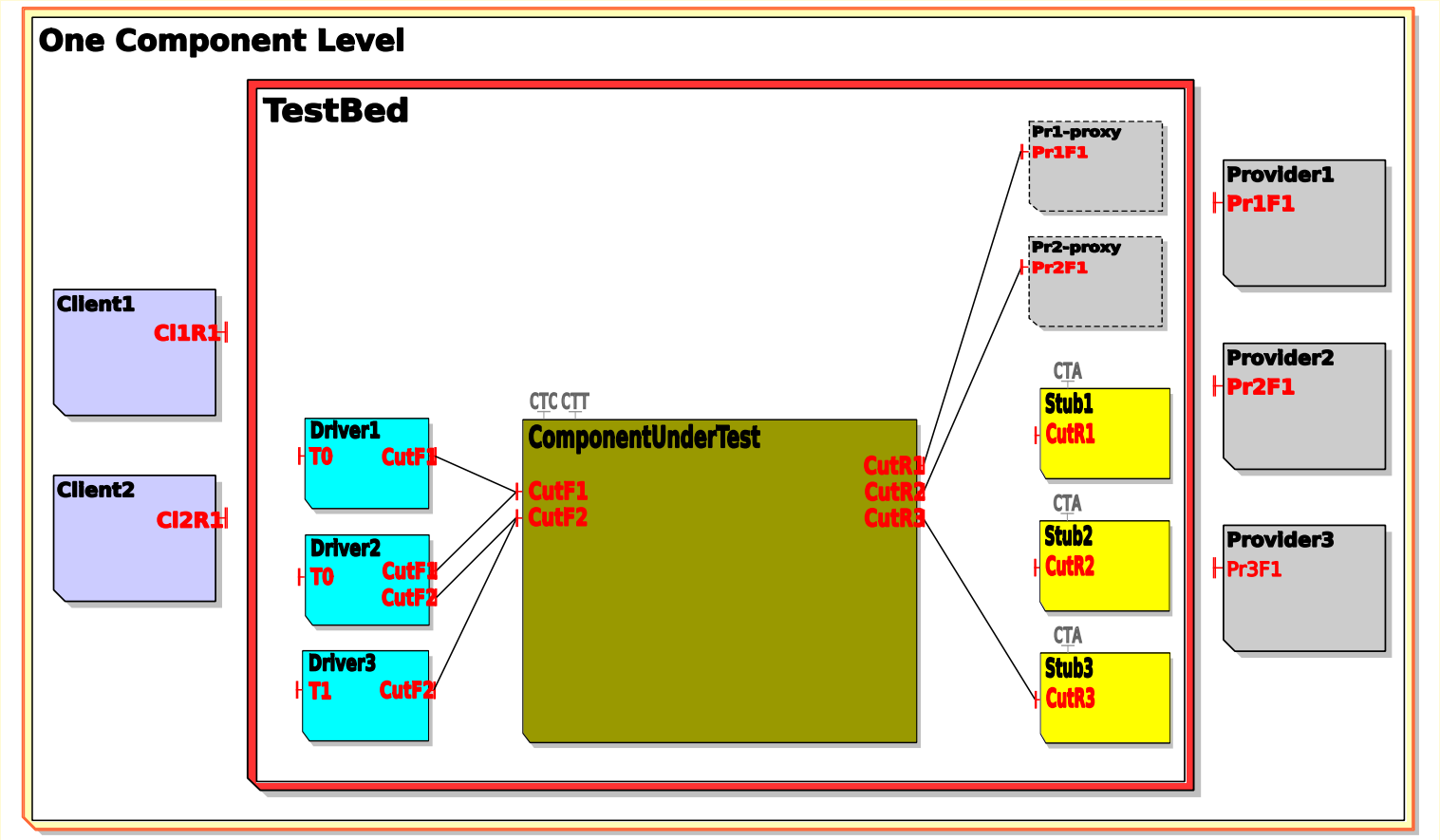
Populate the TestBed



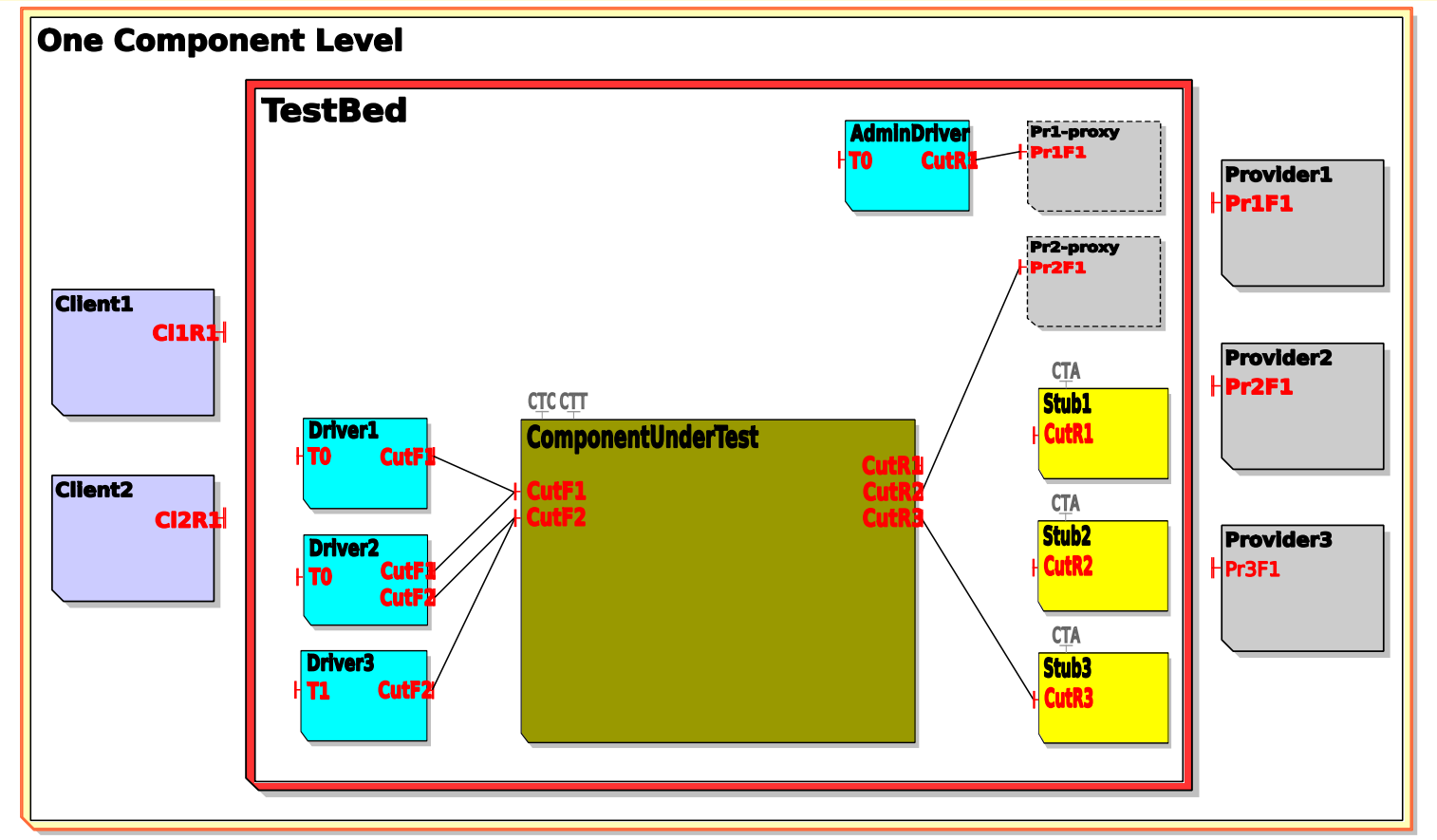
Populate the TestBed



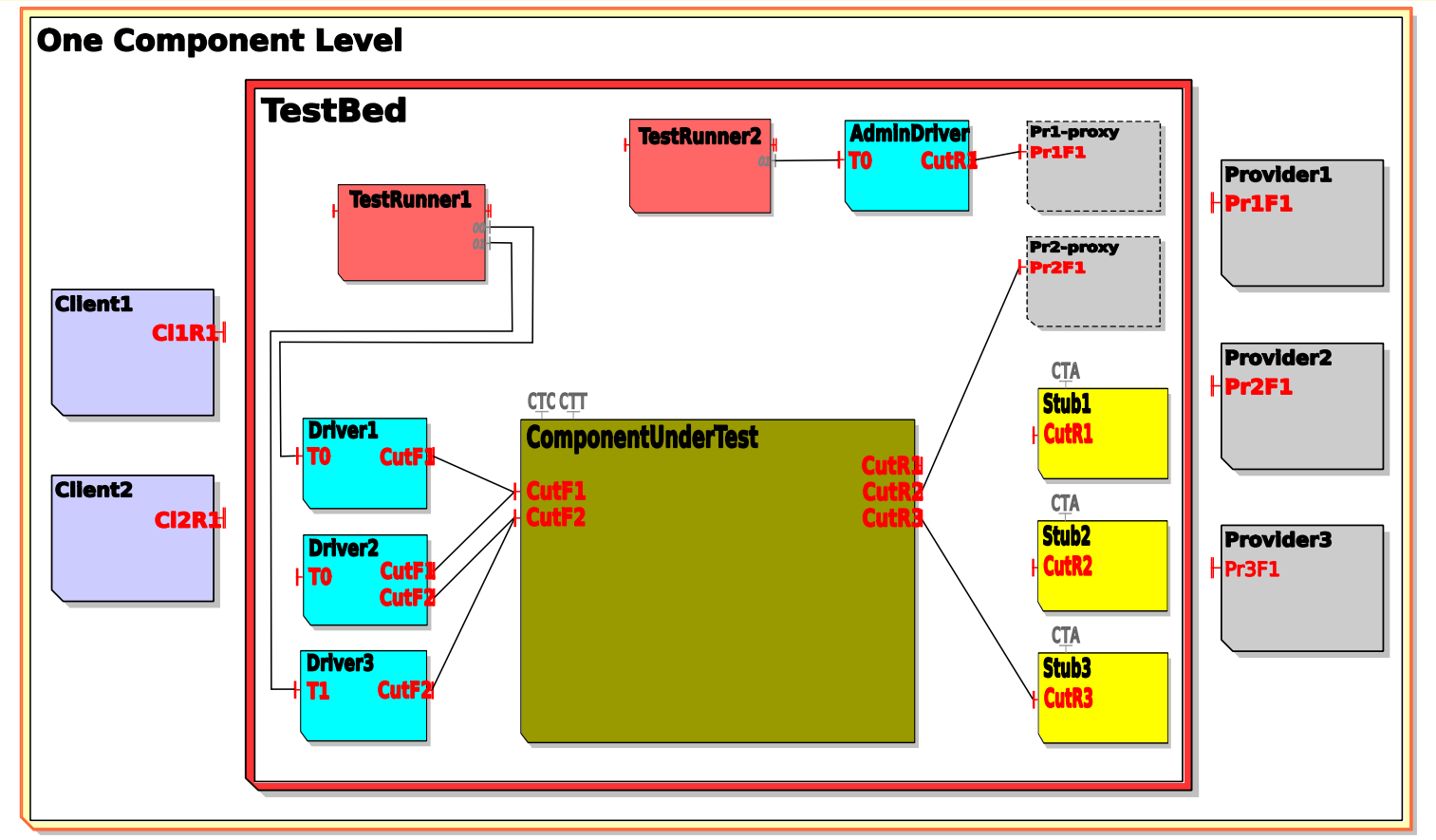
Populate the TestBed



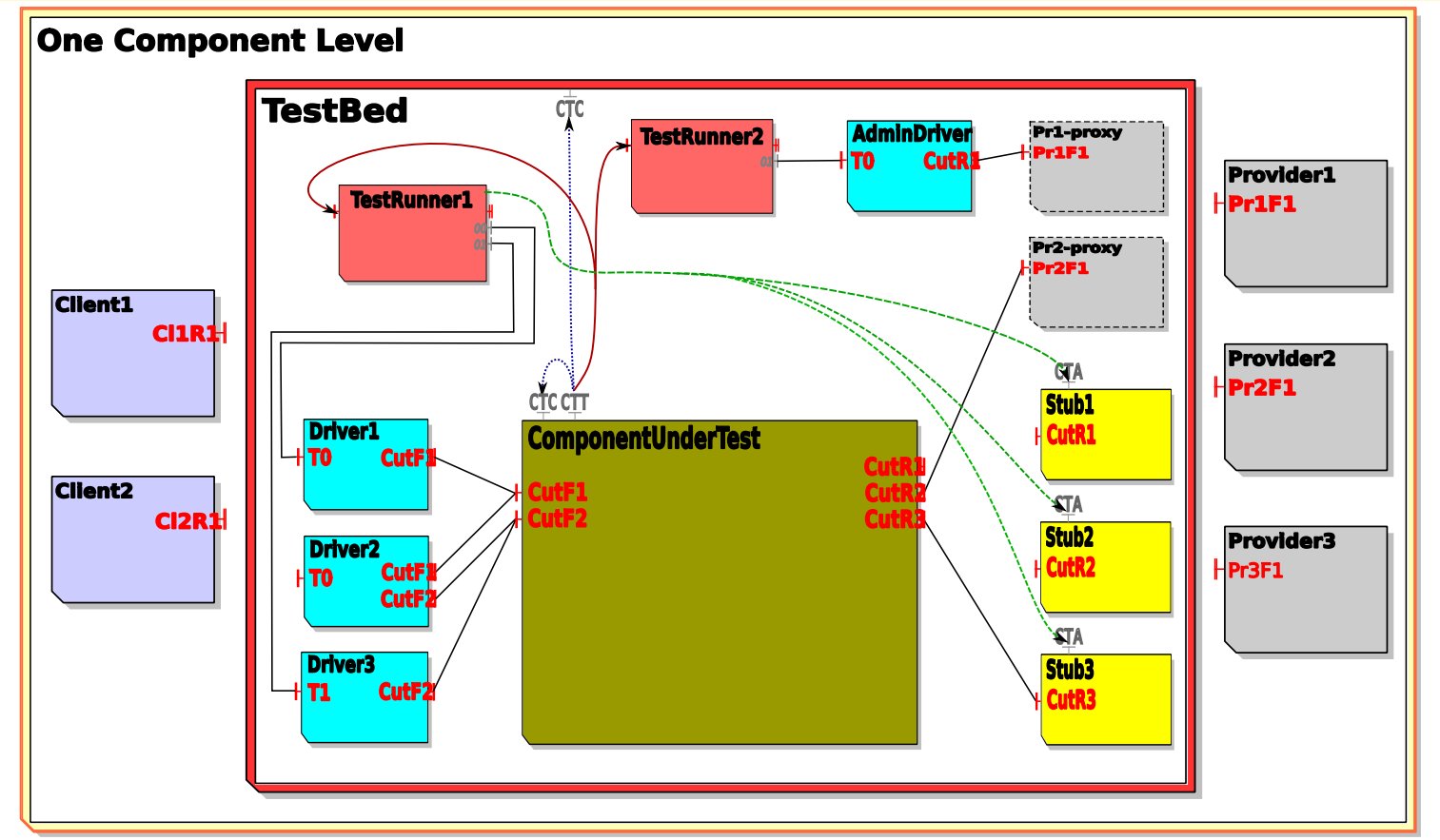
Populate the TestBed



Populate the TestBed



Populate the TestBed



Actual and Future Works

- Tests are defined in a declarative way;
- based only on Fractal components and a test controller;
- low dependency to ConFract: adaptation to other environments;
- not limited to unit-testing: support for admission, integration and regression test;
- possible adaptation to control hierarchical testing.

A prototype of this framework is under design and construction.

CBBT-Fractal is like a *"life-jacket"* for Fractal Components

Questions ? ...