Taming Complexity of Large Software Systems:

Contracting, Self-Adaptation and Feature Modeling

Philippe Collet

Habilitation à diriger des recherches

Université de Nice - Sophia Antipolis

6 décembre 2011

Université Nice Sophia Antipolis

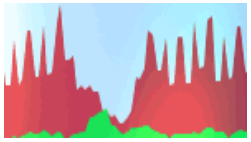
i3S sophia antipolis

cnrs dépasser les frontières

What this is all about

Contracting

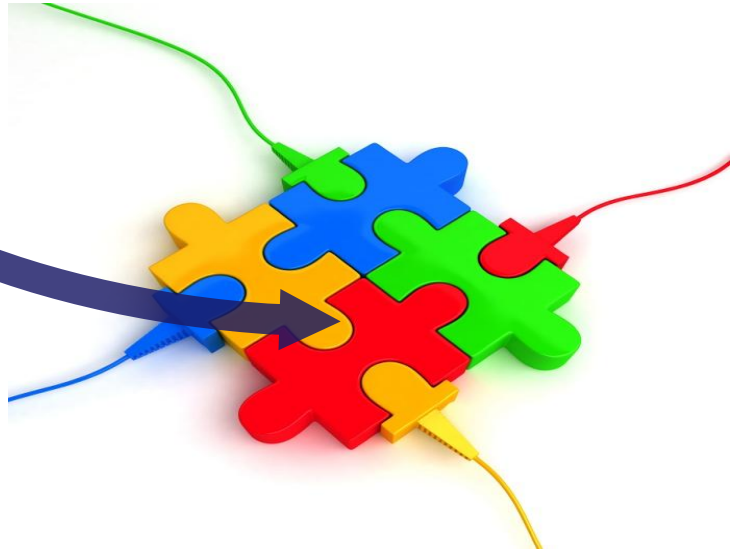
Large software systems



Adding self-adaptive capabilities

Software Engineering

- Systematic, quantifiable, disciplined approaches
- Master complexity to reduce costs

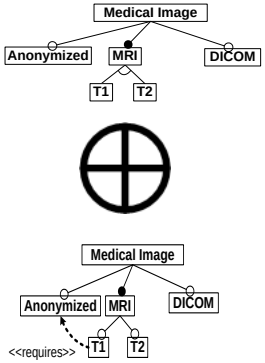


What this is all about

Software Product Lines

Large software systems

Composition



Variability models

Scientific Context and Approach

- **Ever-rising complexity of software**
  - Ultra-large scale (size, volume of data, decentralization, conflicting requirements, continuous evolution)
  - New software architectures (distributed components, services)

➡ **Finding the right trade-off between reliability and flexibility**

➡ **Providing well-grounded but pragmatic techniques and tools for software architects**

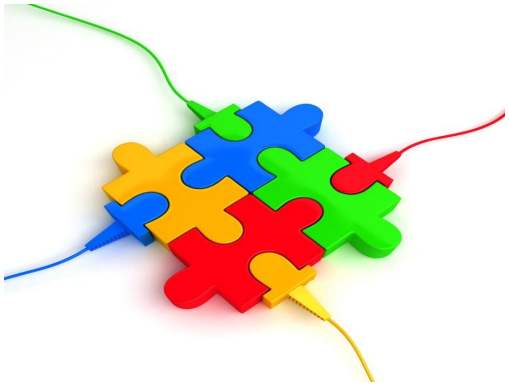
  - How to design dynamically reconfigurable components with confidence
  - How to deal with changes at reconfiguration / run times
  - How to manage variability in large systems of systems

Agenda

- Motivations
- Contracting
- Adding Self-adaptive Capabilities
- Feature Model Composition
- Conclusion and Perspectives

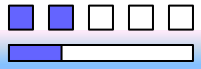
Software Contracting

- **Issues in Component-Based Software Engineering**
  - How to obtain confidence in component specification and assembly
  - How to take into account dynamic reconfigurations



- **A Solution**
  - Adapted forms of **contracts** for CBSE

- **Contracts?**
  - Specification and verification of properties on software entities, while attributing well-defined responsibilities [Design by Contract, Meyer88]
  - Executable assertions for Object-Oriented languages (Eiffel…)



ConFract: a contracting system

Spec.

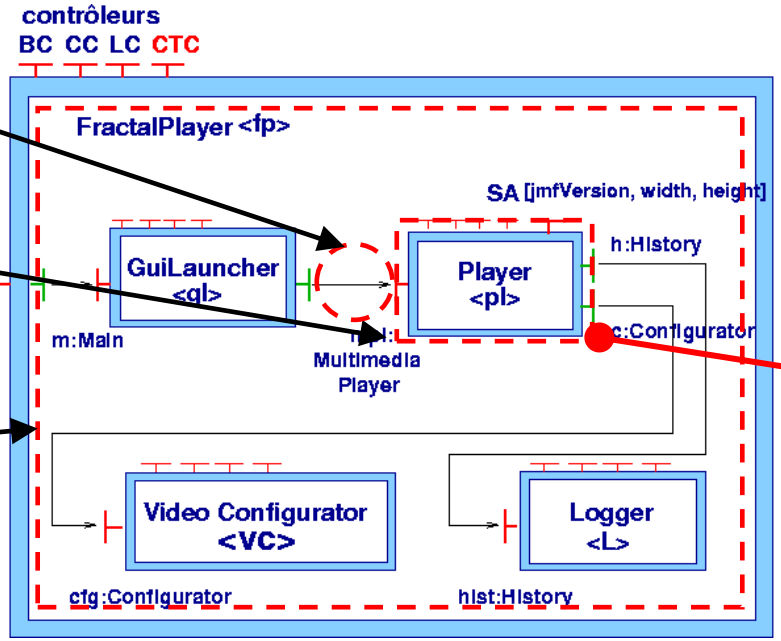
```
on <pl>
 context void mpl.start()
  pre c.expectedCPUUsage(getUrl().getDataSource(),
                <this>.attributes.getFrameRate()) <= 60
  post h.lastUrl().equals(getUrl())
```

Dynamic building

Interface Contract

External Composition Contract

Internal Composition Contract



Fractal hierarchical components

Contract Object

- Participants and responsibilities (guarantor, beneficiary)
- Moments of validity (checking driver)

Contract Management

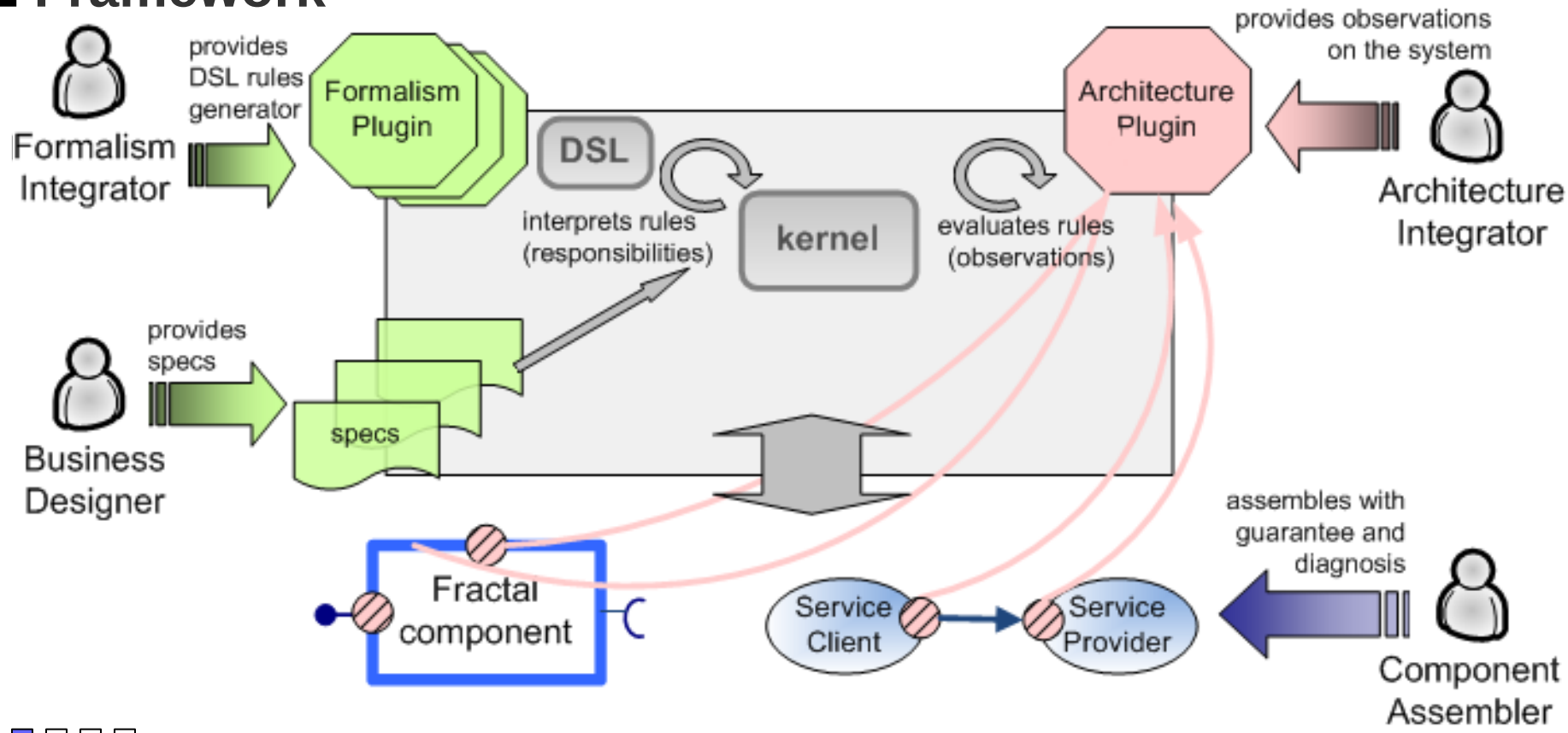
- Incremental construction
- Handling of dynamic reconfigurations

Collaboration France Télécom R&D

Interact: a Contracting Framework

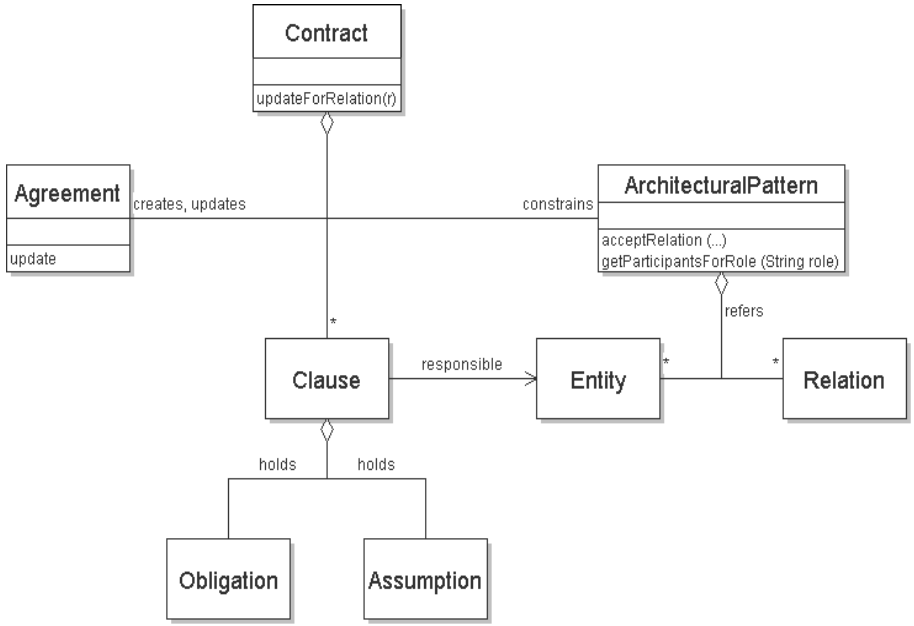
- **Issues**
  - How to integrate different formalisms
  - How to deal with different forms of architectures

- **Framework**



Interact: a Contracting Framework

- **A model for the *contracting kernel***
  - Abstraction
  - Extensible for languages
  - Extensible for platforms

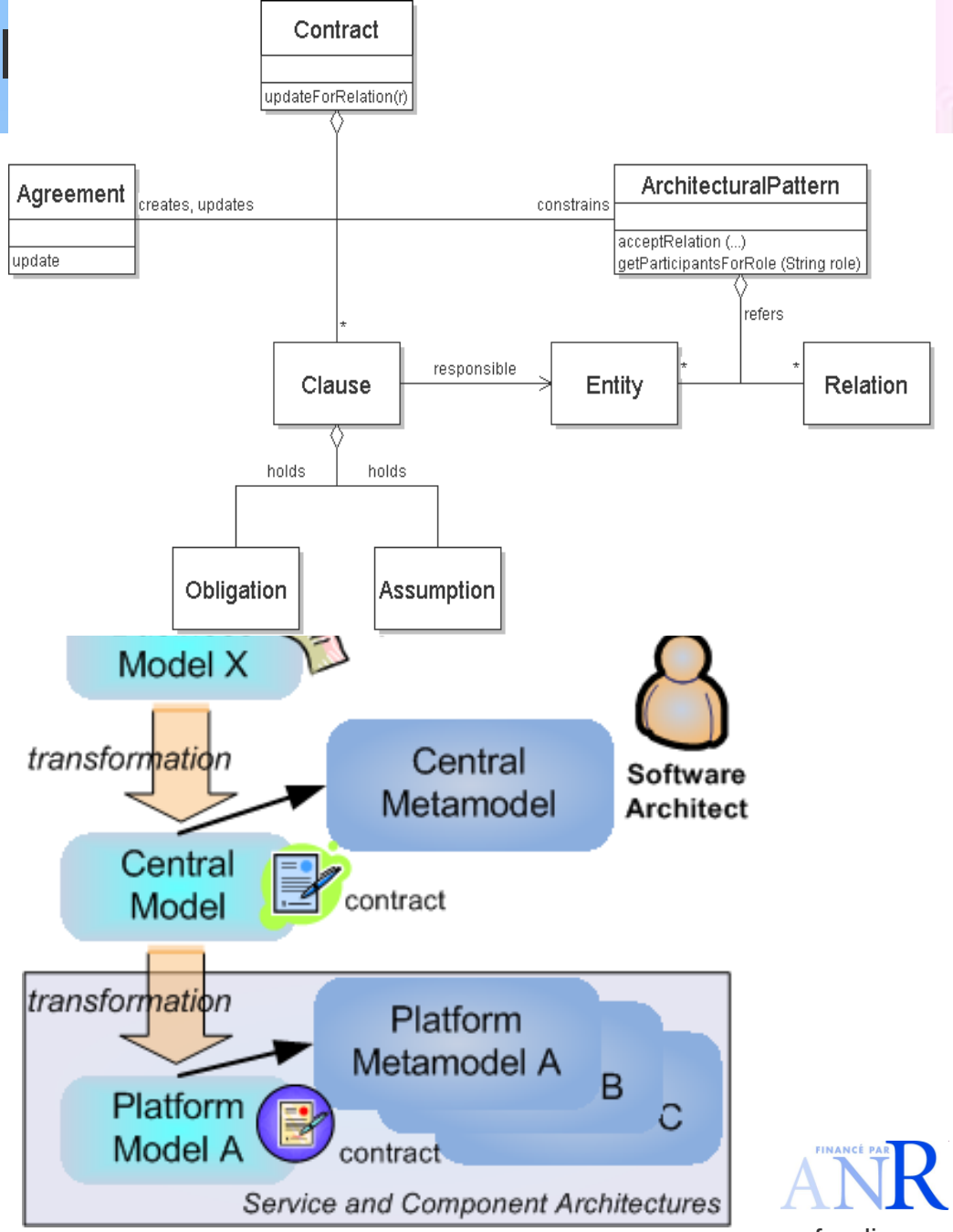


- **A generic and well-grounded *contracting kernel***
  - Assume-guarantee logic (Abadi/Lamport) for responsibility determination [Abadi & Lamport 90]
  - Support for horizontal and vertical compositions

Alain Ozanne PhD
- France Télécom R&D
- Collab. UPMC

Contracts in a Model

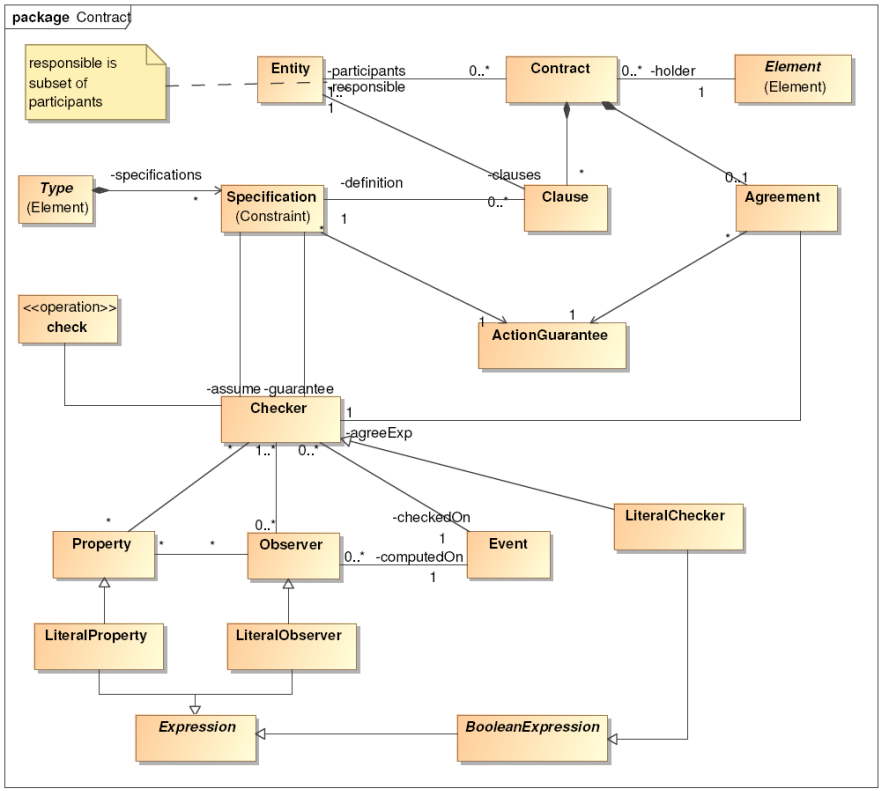
- **Issue**
  - How to automate contracting from business requirements to execution platforms
- **Solution**
  - End-to-end Model-driven engineering process for supporting contracts
  - The FAROS project



FINANCÉ PAR ANR
funding

Contracts in a Model-Driven Tool Chain

- **Extension of the contracting kernel**
  - Event definition
  - Observers and Checkers



- **Applications**
  - Validate the contracting kernel
  - 5 different software platforms (including ConFract)
  - 3 different case studies (including one using ConFract)

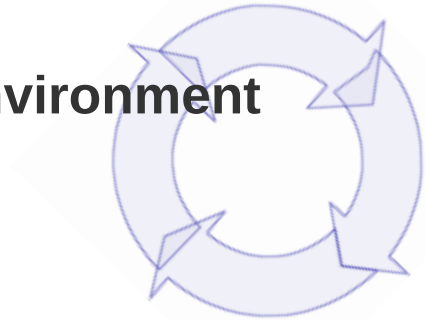
FINANCÉ PAR ANR
funding

Agenda

- Motivations
- Contracting
- Adding Self-adaptive Capabilities
- Feature Model Composition
- Conclusion and Perspectives

Self-adaptation

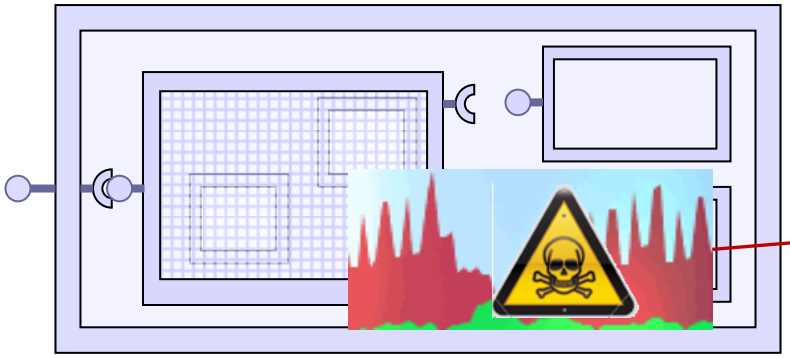
- **Issue**
  - How to master the dynamicity of large scale software systems
- **Self-adaptive system**
  - **Capacity to monitor its own behavior, and its environment**
  - Capacity to evaluate relevant states
  - Capacity to change behavior



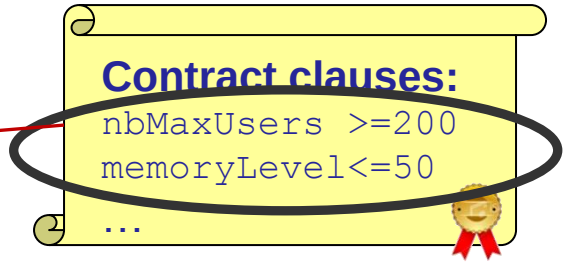
- **Focus**
  - Contracting mechanisms
  - Monitoring mechanisms

Making contracts *negotiable*

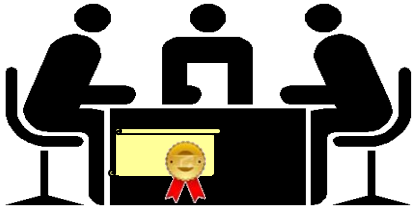
- Capacity to automatically reestablish violated contracts or to reconfigure the architecture



Negotiate on non functional aspects



Integrate negotiation into components

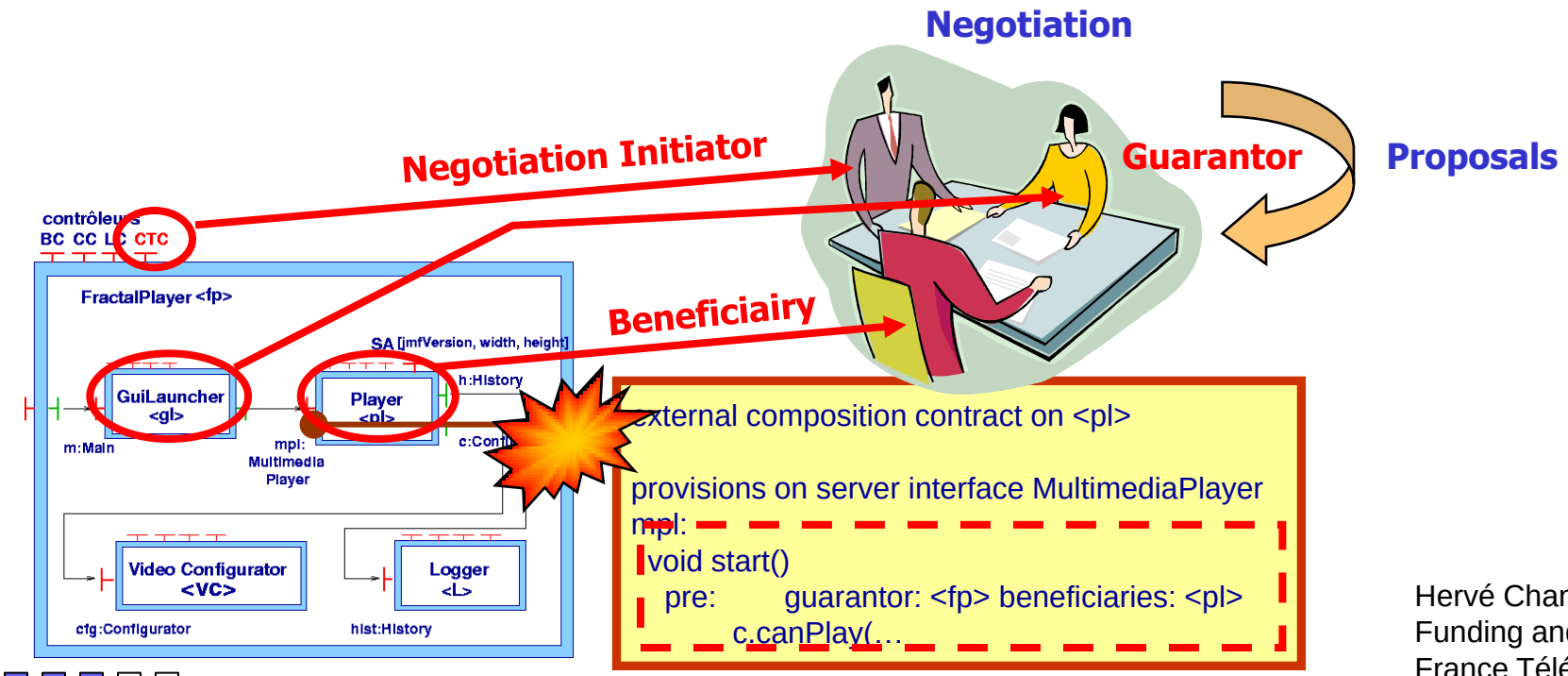


Exploit contract information

Drive negotiation processes

General negotiation model

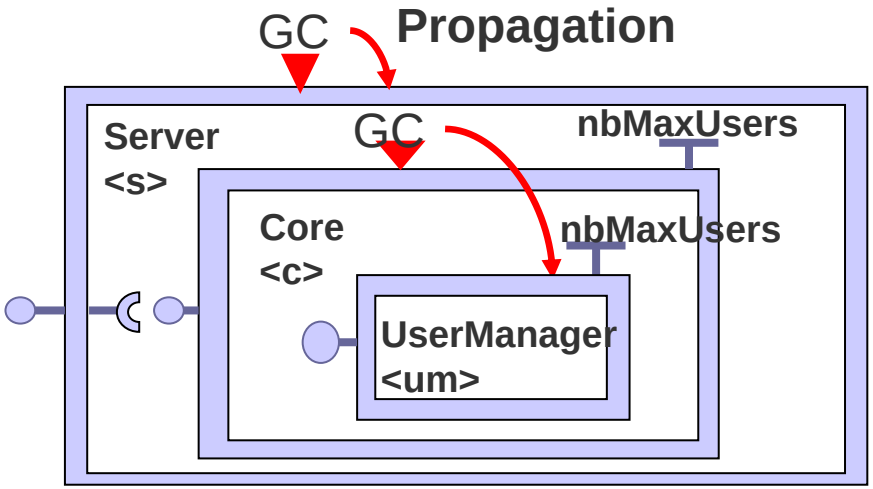
- Inspired by the *Contract-Net-Protocol* [Smith 80]
  - reusing responsibilities determined by ConFract
- Parameterized by negotiation policies (alternatives)
  - Concession based and effort based policies



Hervé Chang PhD
Funding and collab.
France Télécom R&D

Additional Capabilities

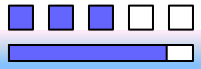
- **Patterns for describing compositional non-functional properties**
  - Classification of properties
  - Integration in component hierarchy
  - Exploitation in effort-based negotiation



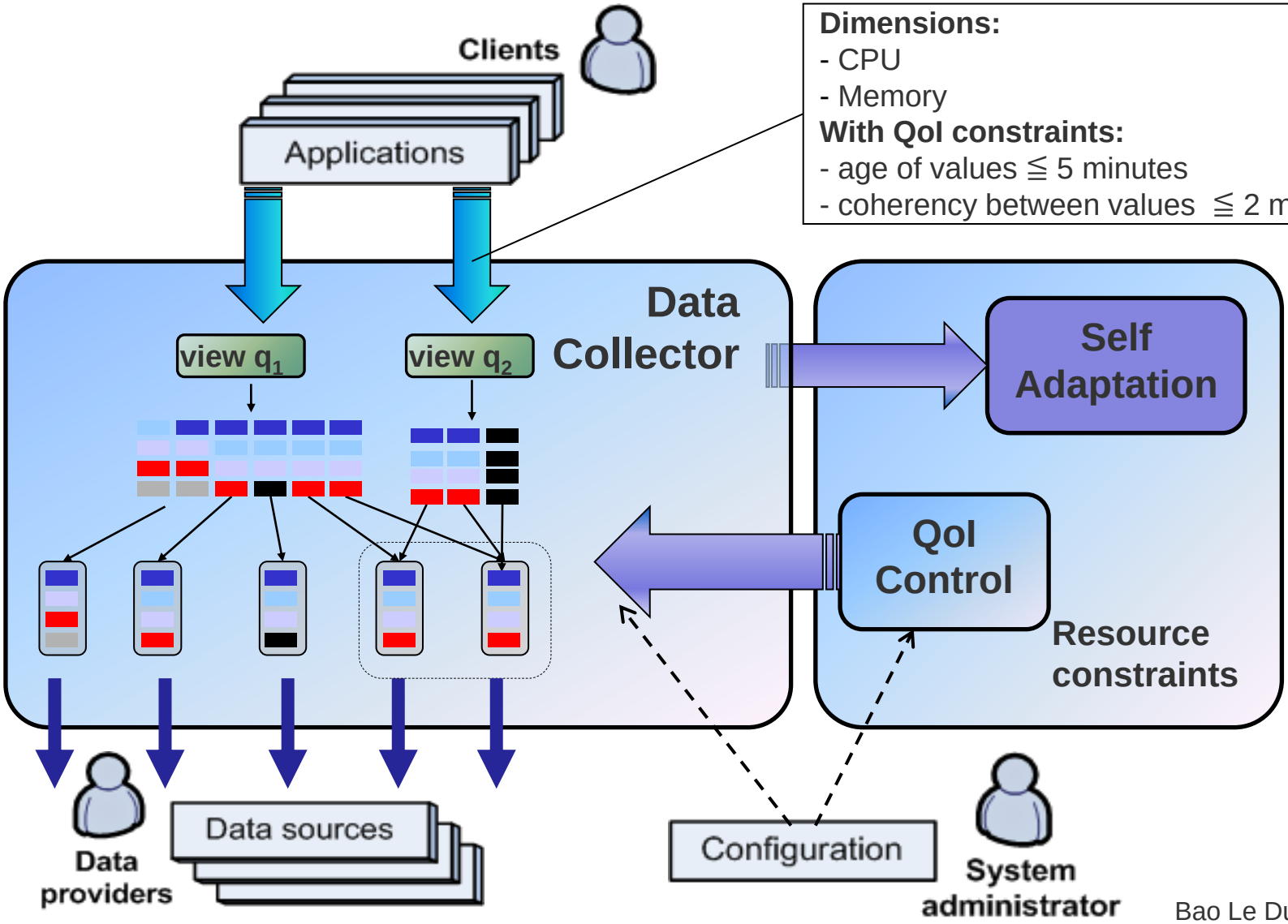
- **Self-adaptiveness of the negotiation system**
  - Negotiation mechanisms as components
  - Contracts on these components (timeout, oscillation detection)

Adaptive Monitoring

- **Issues**
  - Decision making systems are using Service Level Agreements
    - Quality of Service (performance, availability, etc.)
    - Quality of Information (coherency, freshness, etc.)
  - How to manage allocation of scarce resources (bandwidth, CPU, etc.)
  - How to manage changing situations
- **Solution: a framework**
  - Data richness (e.g., aggregation)
  - Quality of information (QoI) awareness (e.g., QoI specification and mechanisms)
  - Resource awareness and enforcement
  - Self-adaptation (e.g., runtime changes of clients, resources…)



ADAMO: QoI-aware Monitoring Framework



Bao Le Duc PhD
- France Télécom R&D
- Collab. UPMC

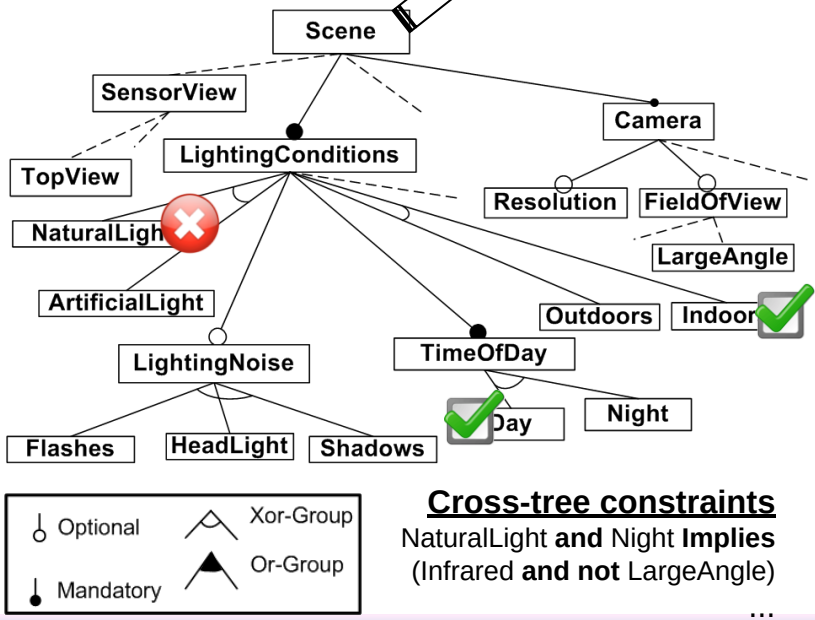
Agenda

- Motivations
- Contracting
- Adding Self-adaptive Capabilities
- **Feature Model Composition**
- Conclusion and Perspectives

Feature Modeling

- **Software Product Lines**
  - Factoring out commonalities for reuse
  - Managing variabilities for software mass customization

- **Feature Models**
  - Widely used
  - Formal Semantics
    - Propositional formula (^, v, ~, ⇔, =>)
  - Automated Reasoning Techniques
    - Satisfiability, configuration checking…
  - Tools
    - Language, editors...



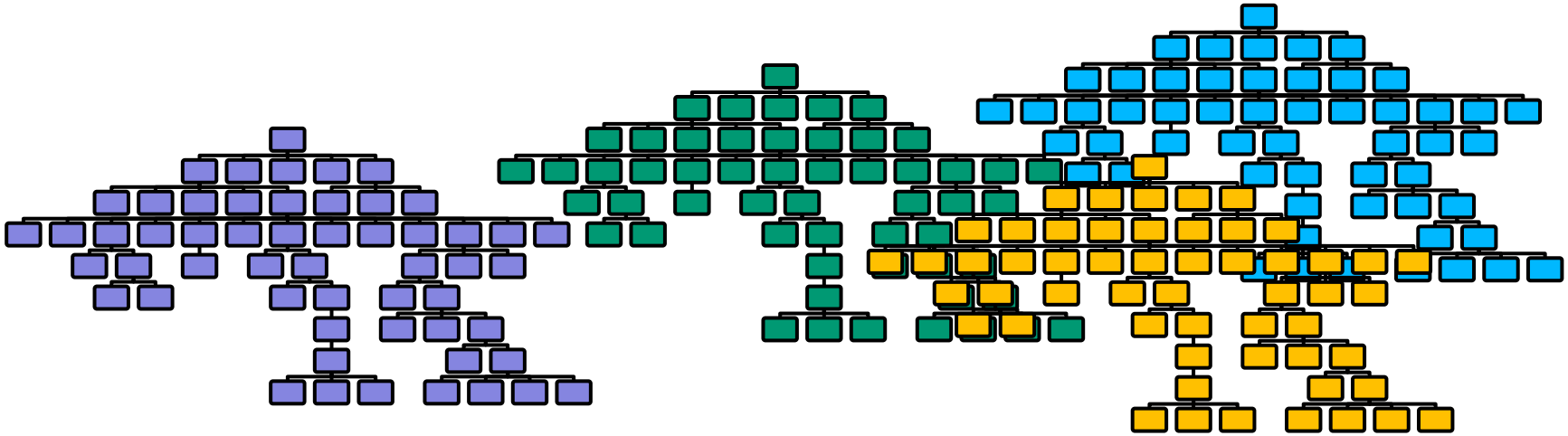
Cross-tree constraints

NaturalLight **and** Night **Implies** (Infrared **and not** LargeAngle)

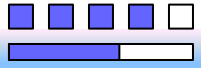
...

Composition of Feature Models

- **Issue**
  - How to manage large, complex and multiple feature models

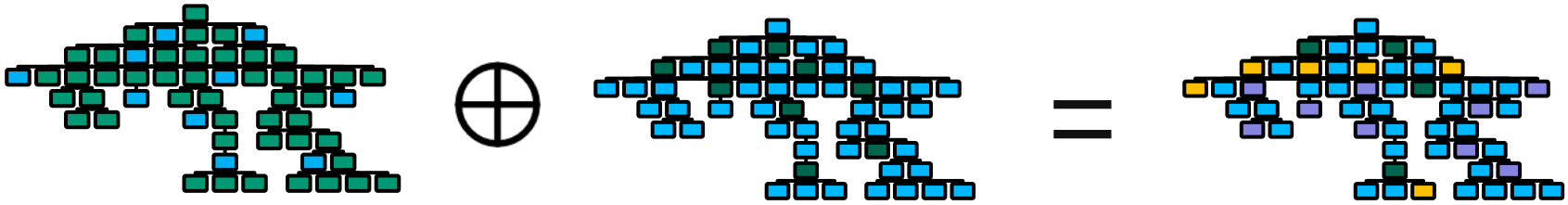


- **Solution**
  - Apply *Separation of Concerns*
  - Provide a set of composition / decomposition operators
  - Ground the operators on a sound basis (semantic not syntactic)
  - Reuse / extend automated reasoning techniques

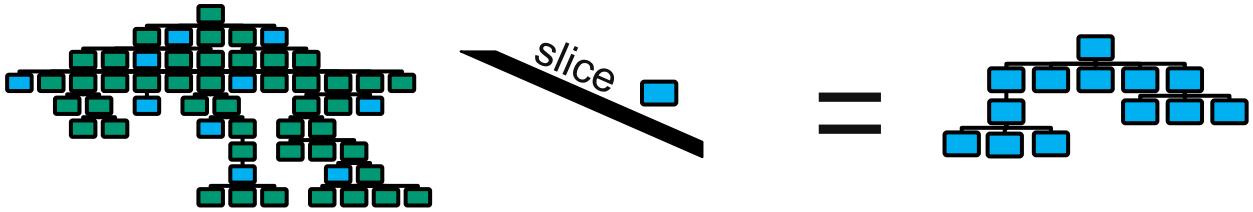


Composition operators

- **Insertion, aggregation, merge (union, intersection)**



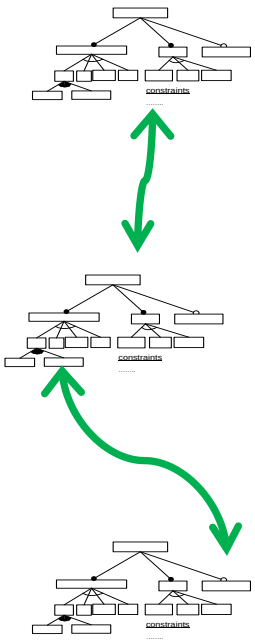
- **Slice (= projection)**



- **Resulting support**
  - Well-defined semantics
  - Guaranteeing semantic properties by construction (configuration set)
  - Producing more compact feature models
  - Efficiently implemented (good scalability w.r.t. existing techniques)

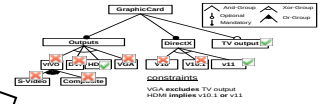
Mathieu Acher PhD
- MESR funding

Domain Specific Language and Applications

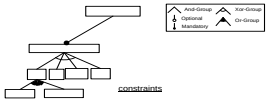


```
// foo.fml
fm1 = FM (“foo1.tvl”)
fm2 = FM (“foo2.m”)
fm3 = merge intersection { fm1 fm2 }
c3 = counting fm3
renameFeature fm3.TV as
“OutputTV”
fm5 = aggregate { fm3 FM
(“foo4.xml”) }
assert (isValid fm5)
fm6 = slice fm5 including fm5.TV.*
export fm6
```

FAMILIAR



True/False
8759
“OutputTV”, “TV”

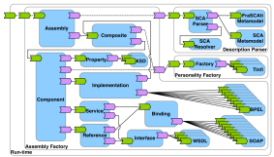


Interoperability **Language facilities** **Environment** eclipse

- **Applications**
  - Consistent assembly of variable service workflow

  - End to end variability handling in video-surveillance processing chains

  - Reserve engineering of architectural feature models



Agenda

- Motivations
- Contracting
- Adding Self-adaptive Capabilities
- Feature Model Composition
- Conclusion and Perspectives

Results

2002 2005 2008 2011

Contracting system

Alain Ozanne

Contracting Framework and Models

Components & services

Contracts and Tests

Not discussed

Hervé Chang

Components & services

Contract Negotiation

Non-functional Properties

Bao. Le Duc

Self-adaptive Monitoring

Mathieu Acher

Software Product Lines

Feature Model composition

Supporting DSL

Research Roadmap

Contracting

1 ongoing PhD

Model-driven Construction of Self-adaptive Systems

SALTY

FINANCÉ PAR ANR

Checking and Contracting Self-Adaptive Systems

Scalable Feature Model Composition

yourCast

FINANCÉ PAR ANR

Relation of Features to Other Models

Software Product Lines of Self-Adaptive Systems

Main publications

- **2 international journals: SQJ, JSW**
- **1 national journal: L'Objet**
- **More than 20 international conferences: ASE, CBSE, SC, SAC, SEAA, SEKE, ECMFA, SOFSEM…**
- **And other publications such as**
  - International workshops
  - Registered / publicly available software
  - Contract and project deliverables…

Questions

applications architecture case checking component
composition concerns configuration constraints
contract control data engineering feature fractal
framework implementation interface language level management
model monitoring negotiation operator process
properties provided provision qoi reasoning resource responsibilities
self-adaptation software specification support systems
testing variability