Les concepts de LfP

F. Kordon & F. Gilliers

LIP6

Fabrice.Kordon@lip6.fr, Frédéric.Gilliers@lip6.fr

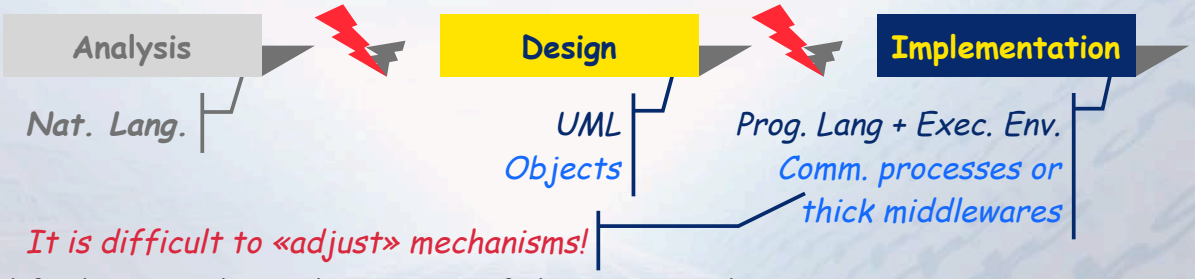
Introduction

- Idées de base (à l'origine de MORSE)

- Chronic Software Crisis
  - Size of systems increases
  - Need for quality increases (in some domains;-),
  - It is also necessary to consider «time to market» aspects
- Is «prototyping» a solution ?
  - IEEE: *A type of development in which emphasis is placed on developing prototypes early in the development process to permit early feedback and analysis in support of the development process*
  - Modeling + «evaluation» + Program synthesis + deployment
  - Towards model based development
- Industrial needs
  - Level 1: accelerate development, integrates «know-how» in tools (key for productivity)
    - *Telecommunications, domestic embedded systems*
  - Level 2: increase quality by using formal verification techniques
    - *Mission-critical and/or high-confidence systems (avionic, transport, etc.)*

Observations concerning distributed systems

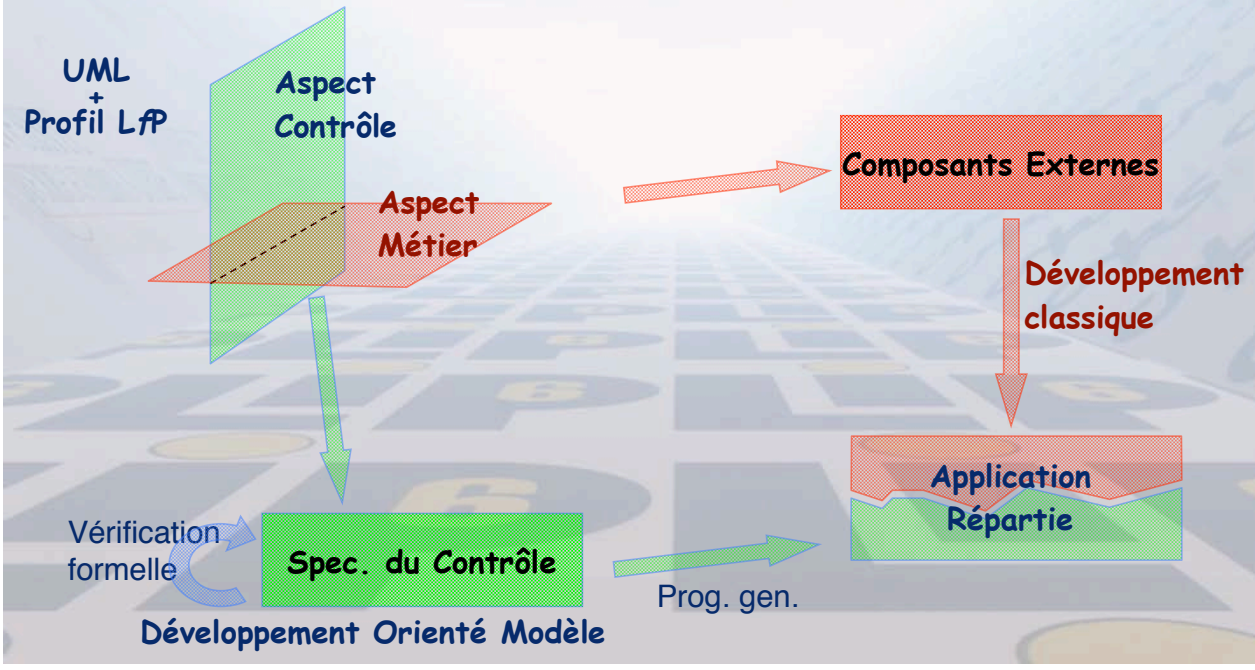
- Disruption in the software life cycle



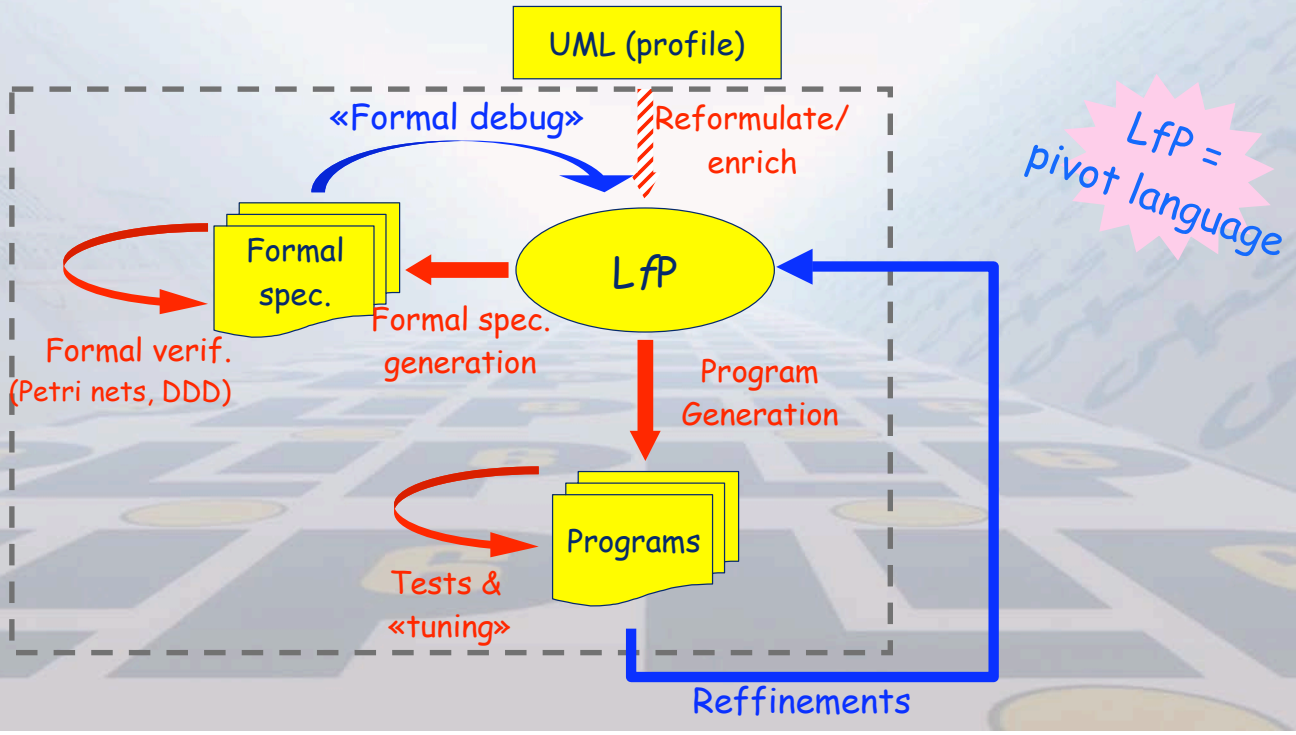
- Need for better and smoother capture of the execution dynamics
  - Execution of a system has to be very precise
    - *Current solution: UML, but connection between diagrams is not formally defined*
  - Implementation is delicate
    - *It is of interest to automate the most difficult part (execution control)*
  - There should be a «central model» for both evaluation/verification and implementation

- Perspectives with regards to maintenance
  - Models are easier to maintain
  - Intensive use of automated processes (flexibility of prototyping)

Approche par prototypage évolutif



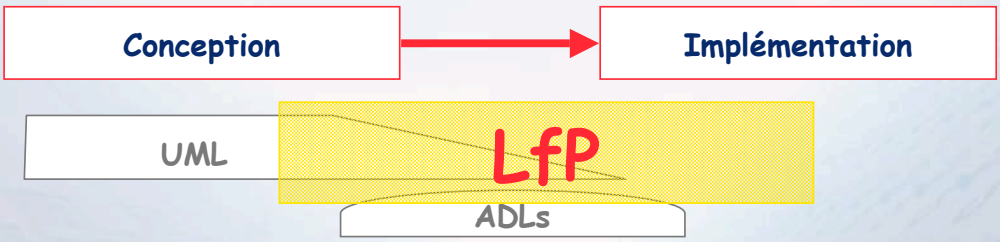
Model Based Methodology



Caractéristiques de l'approche MORSE

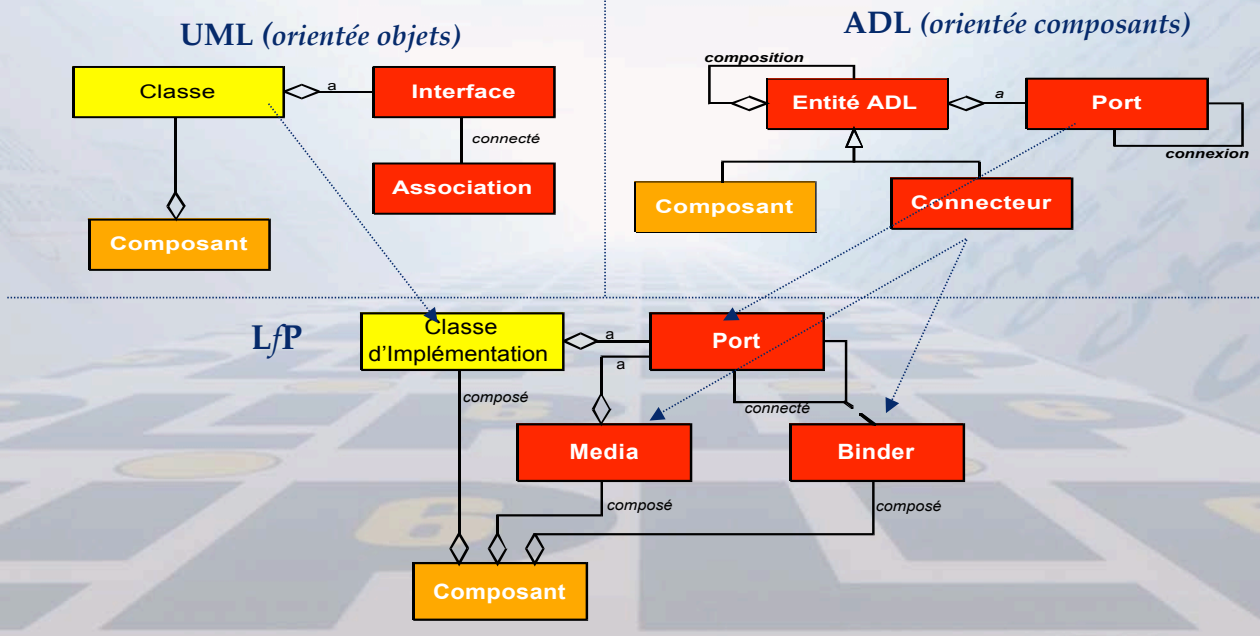
- Séparation entre aspect contrôle et aspect métier
  - Traitement des interactions entre les composants
  - Calculs séquentiels
- Permet
  - Réutilisation des composants métier déjà développés
  - la vérification => on ne peut vérifier que l'aspect contrôle
- Profil LfP pour UML
  - Permettre la structuration de la spécification
  - Rajoute les informations nécessaires pour la génération du modèle LfP
- Conséquence :
  - Définir l'interface entre les deux aspects de l'application

LfP: positionnement



- Style architectural «neutre» → inspiré de RM-ODP
- Intégration à une démarche UML
- Intégration d'éléments d'un ADL
- + Implémentation détaillé

UML & les ADL : Style architectural de LfP



Language overview

- LfP (language for prototyping)
  - Architectural views ⇨ ensure traceability
    - *Deduced from UML + identification of communications elements*
  - Behavioral views ⇨ describe behavioral contracts
    - *Partially deduced from sequence diagrams + connection to state diagrams*
  - Property views ⇨ expected properties (guide for verification)
    - *Properties must be embedded into the specification*
  - Deployment view ⇨ for program synthesis (directives for code gen.)
    - *Link to the target architecture, detailed code generation directives*
- Now strongly linked to a UML-profile (UML-M)
  - Expression in UML-2.0 and UML-1.3
    - *Due to tools limitations…*

An LfP Example : a simple chat system

A chat system (client/server)…

… and using a communication system

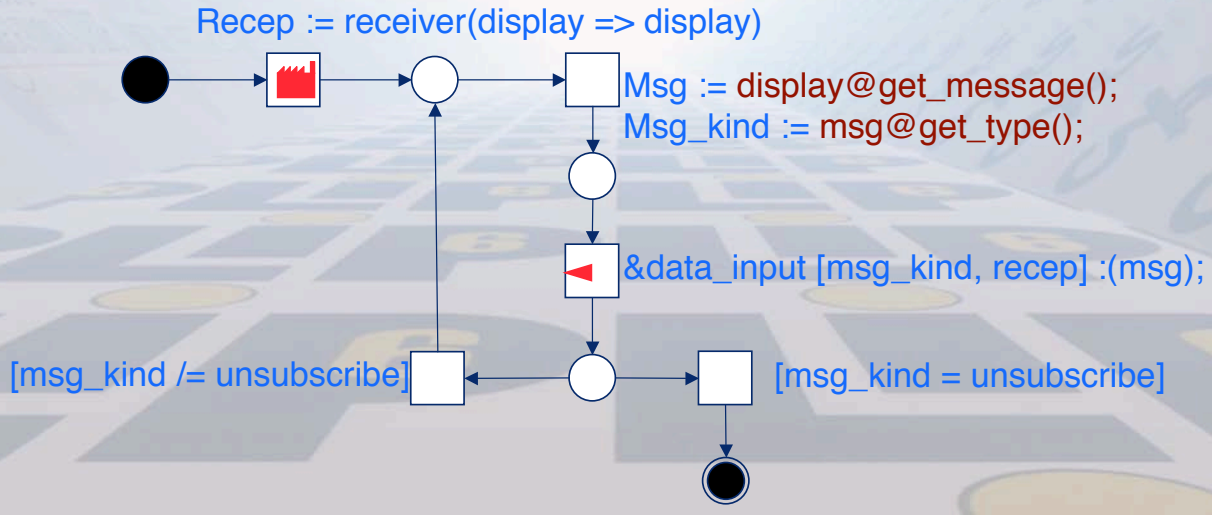
… relying on a graphical API…

```
Type GUI is opaque
  function get_message return t_message ;
  procedure display_message (msg : in t_message) ;
End ;
```

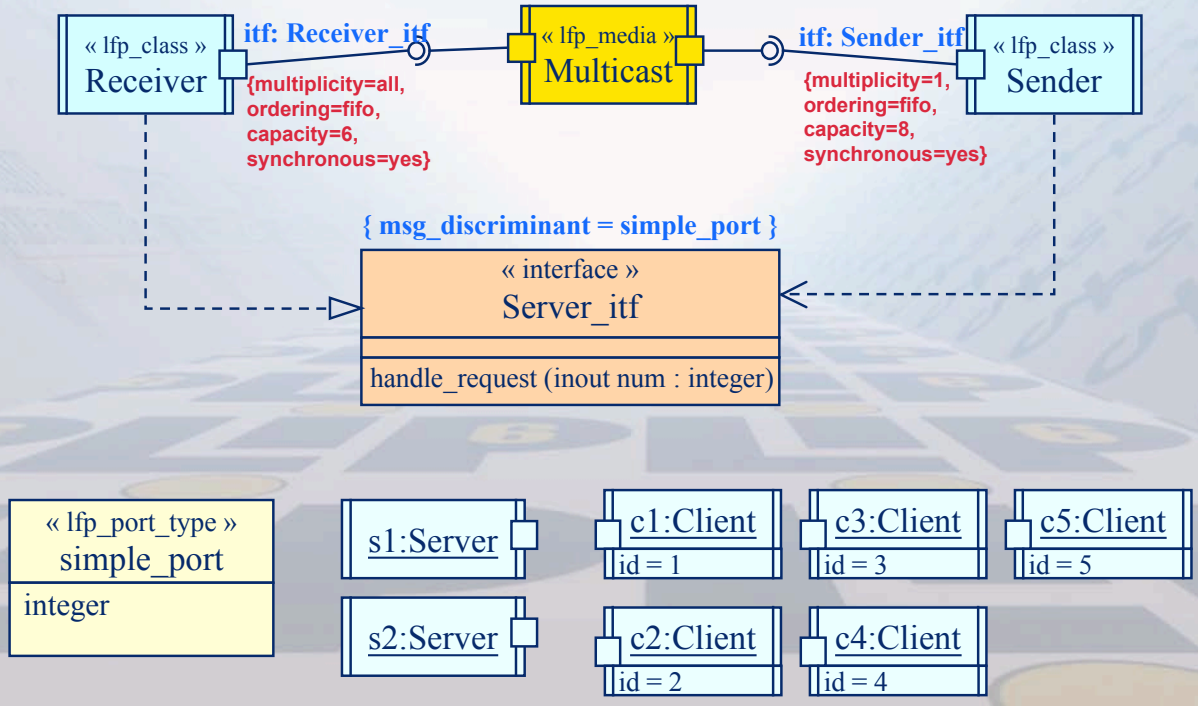
```
Declarative part
```

Behavior of class sender

- Behavior described using structured hierarchical automata
  - Description of classes and media
  - Global view for a class (its «protocol»)
  - Behavior of class's methods (here, client)

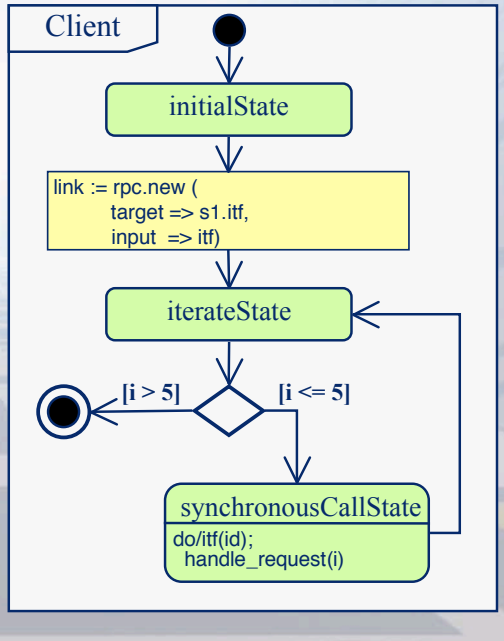


A way to model the architecture in UML



A way to model behavioral diagrams In UML

- UML 2.0
  - State charts have ports!!!
  - Useful because we need to identify queues associated to «entry points»



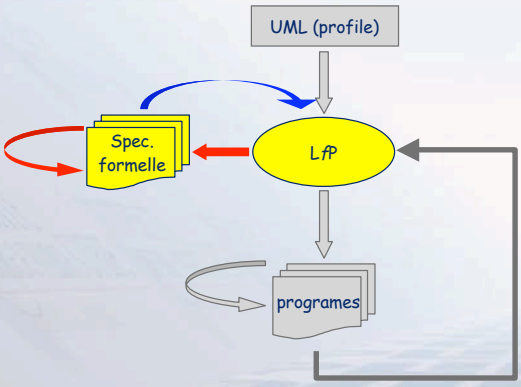
Properties view

- Objective: embed and stick properties to the specification
  - Have a «natural way» to express them (for engineers)
- Types of properties
  - Use of collaboration and sequence diagrams
    - *Deduce Temporal Logic formulæ: at least, the specified scenarios have to be considered*
  - Use of OCL
    - *Deduce invariants in the system: unexpected configuration do not occur*
  - Use of more specific Temporal Logic formulæ
    - *To explicitely express causality*

underlying techniques are based on Model Checking from Colored Petri Nets (well formed)

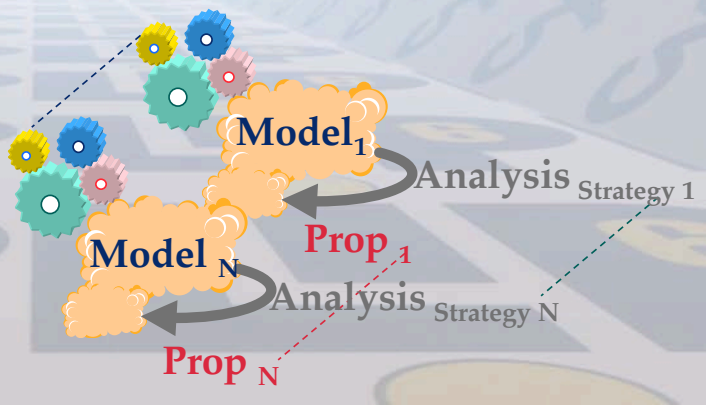
- Towards a «push-button approach»

Linking to formal methods

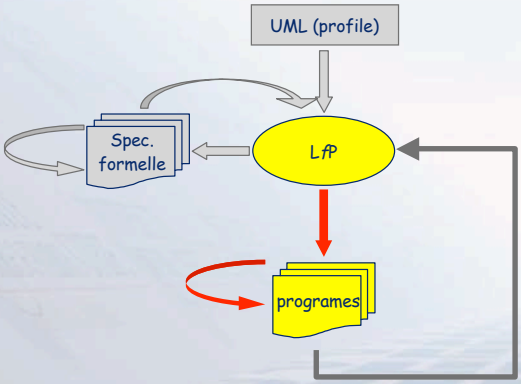


- Various formal specifications
  - Petri Nets
  - Data Decision Diagrams
- Warning: integration of external components
- Automatic translation
  - Automatization being performed

- Require intensive optimizations
  - Use of expected properties
  - Use of the specification structure
  - Reductions during generation
- Several models must be generated



Relation to programs

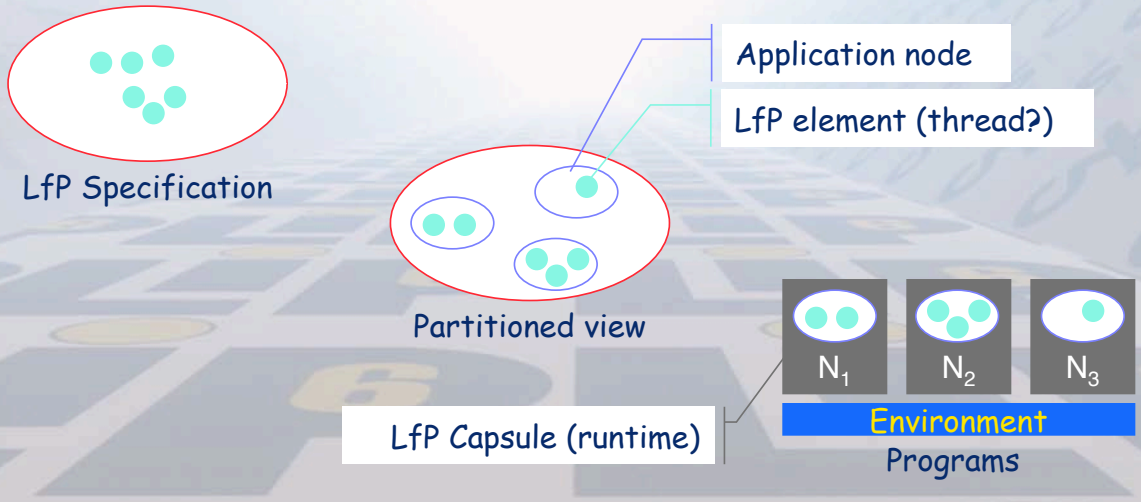


- Requires a generic prototype architecture
  - Integrates a communication pattern with external components

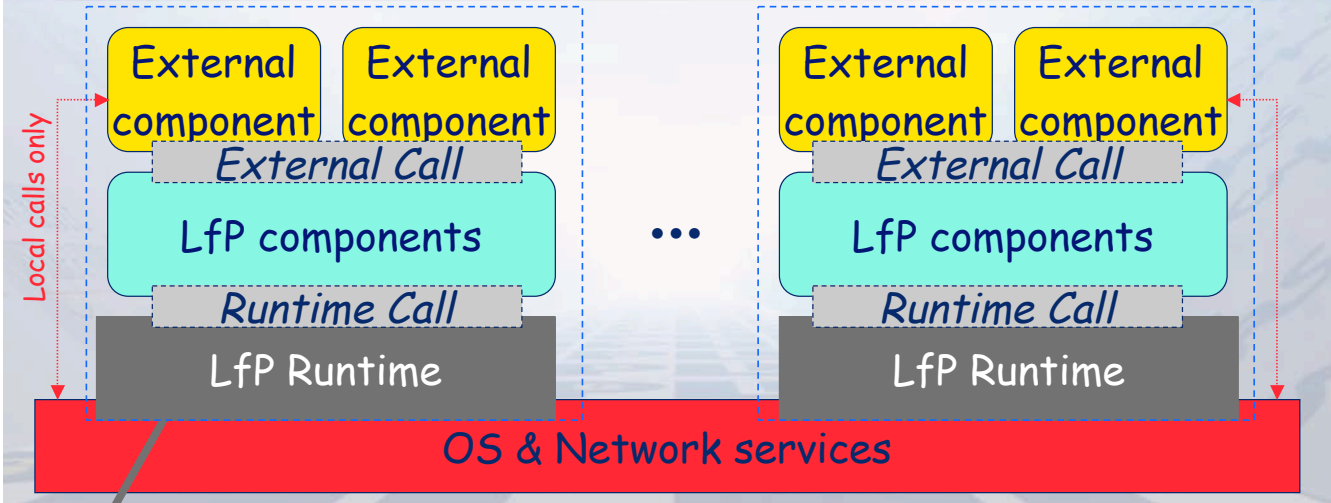
- Requires a set of services (runtime)
  - Similar to programing languages;-)
  - Provides support functions to operate LfP specifications
- LfP runtime and middleware?
  - Similar objectives
  - Require facilities for deployment
  - Discussed later

From the model to the program

- LfP contains a deployment view
  - Yet experimental in its syntax (XML data associated to the specification)
- Generation approach



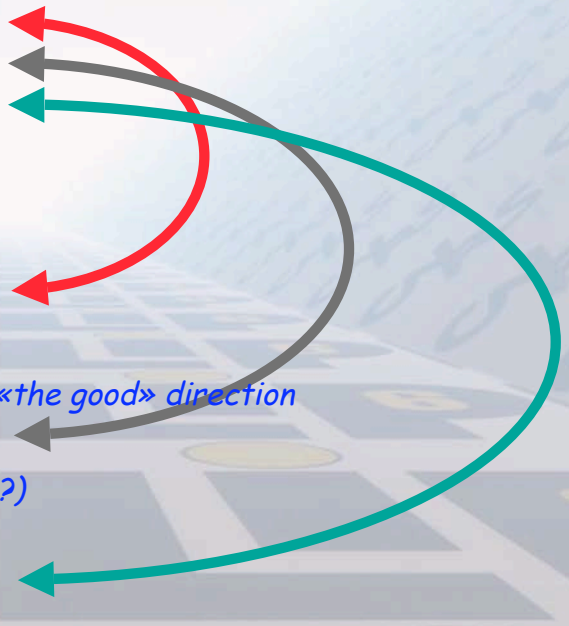
Implementing an LfP Model



Virtualization of the «execution environment» (~ middleware)
What services for specific needs (mobility, adaptability, FT, etc.)
What services for deployment (heterogeneous hosts, unused services, etc.)
What services to configure nodes (no useless code, performances, etc.)

CONCLUSION

- Distributed applications are a difficult task
  - Handling complexity of interactions
  - Handling deployment onto machines
  - Handling configuration (on a node)
    - *Certification, real-time, etc.*
- Integrated methodology can help!!!
  - Modeling and formal methods
    - *Experimentation on LfP*
    - *Why not UML? goes somewhat in «the good» direction*
  - Architecture languages:
    - *Software or hardware (need both?)*
  - Middleware manufacturing
    - *Middleware «à la carte»*



PROJECTION DANS LE PROJET MORSE

- Ce qui reste de LfP
  - Usage de LfP en interne (rôle pivot)
    - *Langage intermédiaire*
    - *Liaison avec la vérification*
    - *Liaison avec la production de code pour les aspect contrôle*
  - UML-M (langage vu par les utilisateurs de la méthode)
    - *Profil sur UML-2.0*
    - *Prise en compte de l'aspect architectural et comportemental*
    - *Vers un usage des diagrammes de séquences pour les propriétés «temporelles»*
    - *Vers un usage d'OCL pour des propriétés «d'accessibilité»*
- La méthodologie
  - Démarche par prototypage

ET MAINTENANT...

- La phase passionnante
  - Éprouver l'ensemble dans un contexte industriel
  - L'épreuve du feu?
    - *Des études de cas ont été traitées*
    - *Elles ont validé les fonctionnalités*
    - *Elles n'ont pas encore validé le passage à l'échelle*