

From Software Engineering To Linguistic Engineering

Laurence Ayache & Pierre Crescenzo & Jean-Louis Paquelin

ALLI
I3S - LIM
CNRS - UNSA

ayache@lim.univ-mrs.fr & Pierre.Crescenzo@i3s.unice.fr & jlp@alli.lpl.univ-aix.fr

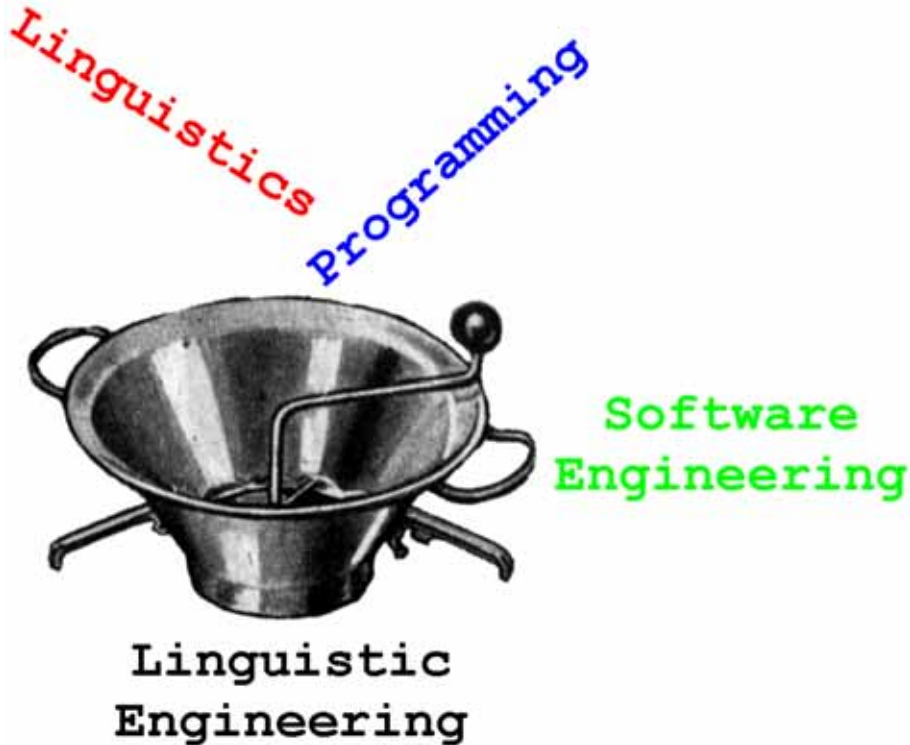
Plan

Goals

Focus

Elements of solution

Introduction



Goals: Modelization

- Near from the theory to strictly respect its semantics
- Near from the real world to naturally handle data

$\text{plus}(\text{succ}(\text{succ}(\text{succ}(\text{zero}))), \text{succ}(\text{succ}(\text{zero}))) = \text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{succ}(\text{zero}))))))$

$$3 + 2 = 5$$

Goals: Expressiveness (1)

- Higher-level concepts

```
print_integers_tree(An_integers_tree)
```

```
print_reals_tree(A_reals_tree)
```

```
print_trees_tree(A_trees_tree)
```

...



```
print_tree(A_tree, An_element_printing_routine)
```

Goals: Expressiveness (2)

- Active constraints

Passive constraints	Active constraints
X is a free variable	
X < 4 ? <i>Success: X is free</i>	X < 4 ! <i>Success: X < 4</i>
X ≥ 2 ? <i>Success: X is free</i>	X ≥ 2 ! <i>Success: 2 ≤ X < 4</i>
X = 7 ? <i>Success: X = 7</i>	X = 7 ! <i>Failure: 2 ≤ X < 4</i>

Goals: Expressiveness (3)

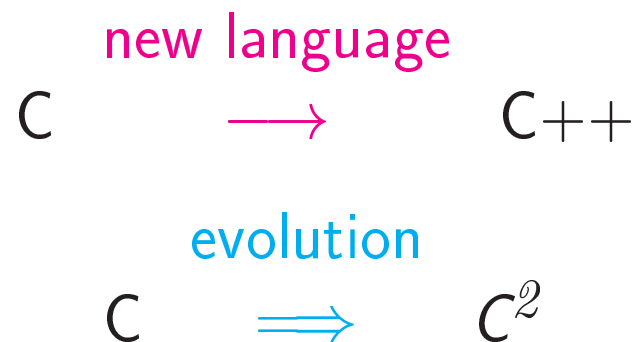
- Query language
- Logic relations
- Type relations

A **noun** is a **word**.

A **verbal phrase** is at least constituted of a **verb**.

Goals: Evolutivity

- To add concepts or to follow linguistic-theories evolutions
- To correct **Anima**'s imperfections
- To be used as test bed



Goals: Integration, Introspection and Control (1)

- Tools (debugger, profiler, ...) must be at Anima's level

C++ (source level)	C (target level)
<code>O = new Circle;</code>	<code>if(!(_23 = (T_42 *)malloc(sizeof(T_42) + 15))) { _cpp_errno = _NONEW_12; _internal(_ERROR_PRINT, _NONEW); }</code>
Logic (source level)	Prolog (target level)
$p(a) \vee p(b) \rightarrow p(c)$	<code>p(a) :- p(c). p(b) :- p(c).</code>

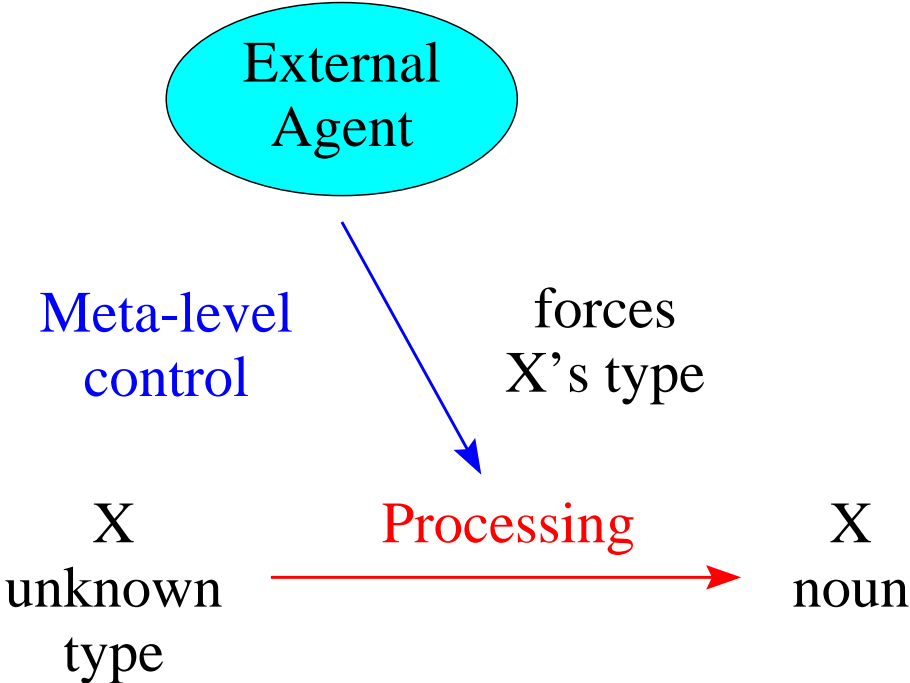
Goals: Integration, Introspection and Control (2)

- Programmers' tools have to exist as well as users' ones

Programmer level	User level
Debugger	Inspector
Profiler	Statistics
Controller	Tracer

Goals: Integration, Introspection and Control (3)

- Control meta-level is desirable



Goals: Software Engineering (1)

- Legibility
 - Natural syntax
 - Users' knowledge respect
 - Other Software-Engineering principles respect
- Reliability
 - Validity
 - Robustness

Goals: Software Engineering (2)

- Extensibility
 - Easy adaptability on changing
- Reusability
- Compatibility
 - Interfacing ability
- Performance

Goals: Software Engineering (3)

- Portability
- Verifiability
 - Easier collection procedures
 - Easier guarantee procedures
 - Easier proving
 - Easier testing
 - Easier debugging

Goals: Software Engineering (4)

- Integrity
 - Components (programs, data, documents, ...) protection
 - Usage simplicity
 - Easy **Anima** learning and operating
 - Easy results interpretation
- Cause and consequence of all other Software-Engineering principles

Focus: Prolog

- Not very legible but NLP-widely-used syntax
- No type system and sometimes no constraint solver
- Not very configurable and extensible
- No appropriateness with recent grammatical theories
- No control meta-level
- In relation to software engineering: not very maintainable, a bit reusable (but not reused), reliable

Focus: LIFE

- Complex and not very legible syntax
- Good expressiveness and higher-level concepts
- Existing type system and constraint solver
- Not very configurable and extensible
- Enough appropriateness with recent grammatical theories
- Disused and not very performant interpreter and compiler
- No control meta-level
- In relation to software engineering: not very maintainable, a bit reusable (but not reused), not reliable

Focus: Oz

- Disconcerting but clear syntax
- Good expressiveness and higher-level concepts
- Existing inadequate type system but good constraint solver
- Configurable and extensible but at C++ level
- No appropriateness with recent grammatical theories
- No control meta-level
- In relation to software engineering: enough maintainable, a bit reusable (but not reused), reliable

Elements of solution: Modelization

- Good expressiveness
 - depends on higher-level concepts
 - The most important: constraints
- Encapsulation
 - provides good control and autonomy

→ Constrained objects

Elements of solution: Expressiveness

Good expressiveness relies on

- Higher-level concepts
- Active constraints
- Query language
- Logic relations
- Type relations (Inheritance link, *Clientèle* link, ...)

and is implemented by

- Objects
- First-order discipline

Elements of solution: Evolutivity

- Meta-Object Protocol
 - Basic-mecanisms transparent box
 - Basic-mecanisms modification ability

Elements of solution: Integration, Introspection and Control

Analysis + Time + Work

=

Debugger, Profiler, ...

⇓

Integration + Introspection + Control

Elements of solution: Software Engineering (1)

- Legibility
 - Encapsulation
 - Overloading
 - Polymorphism
 - Modularity
- Reliability
 - Specification definition and implementation
 - Designers' experience and appraisal

Elements of solution: Software Engineering (2)

- Extensibility
 - Modularity
 - Autonomy
- Reusability
 - Of existing languages
 - Relies on object technology

Elements of solution: Software Engineering (3)

- Compatibility
 - Objects as exchange standard
 - Autonomy
- Performance and Portability
 - Well-known solutions
- Verifiability
 - Well-known and well-tried algorithms
 - Helped with good tools (debugger, profiler, ...)

Elements of solution: Software Engineering (4)

- Integrity
 - Robustness
 - Designers' talent
- Usage simplicity
 - Application-domain (NLP) knowledge

But few direct control

Conclusion and Future

Many existing partial solutions but no integration



Integration under control of Software-Engineering through a coherent multi-paradigm language: **Anima**