Formal Specification, Rapid Prototyping and Program Calculation — an Industrial Experiment using the CAMILA/SETS Approach

UNU/IIST — Seminar n.º 1
5–16 May 1997
Macau

J. Nuno Oliveira
University of Minho / INESC (Group 2361)
Braga
Portugal

email: jno@di.uminho.pt
http://www.di.uminho.pt/~jno

Abstract

- This talk reports on UM/INESC 2361 group's experience on using formal methods throughout an Eureka project whose partners were software companies neither trained nor interested in such methods.
- The main target of the consortium was the design and implementation of an object-oriented meta-CASE system which should include mechanisms for re-using arbitrary information concerning large software systems.
- Encharged of specifying the system's overall architecture from the early stages of the project, INESC researchers decided to produce a formal specification specially concerned with keeping the overall behaviour of the system unambiguous.
- Lack of communication with INESC partners was solved by deriving a rapid prototype from the system's specification, disguised behind a UI layer running on Windows 3.1. Early feedback on the hard technical subtleties of the design (as unveiled by the formal specification) could be obtained by our partners by observing the prototype's behaviour.
- Eventually, all main implementation decisions could be validated by calculation. In the system's final prototype, running on Windows 3.1, all "formal" code gave room to C/C++ code.

The Industrial Case Study

Project Profile:

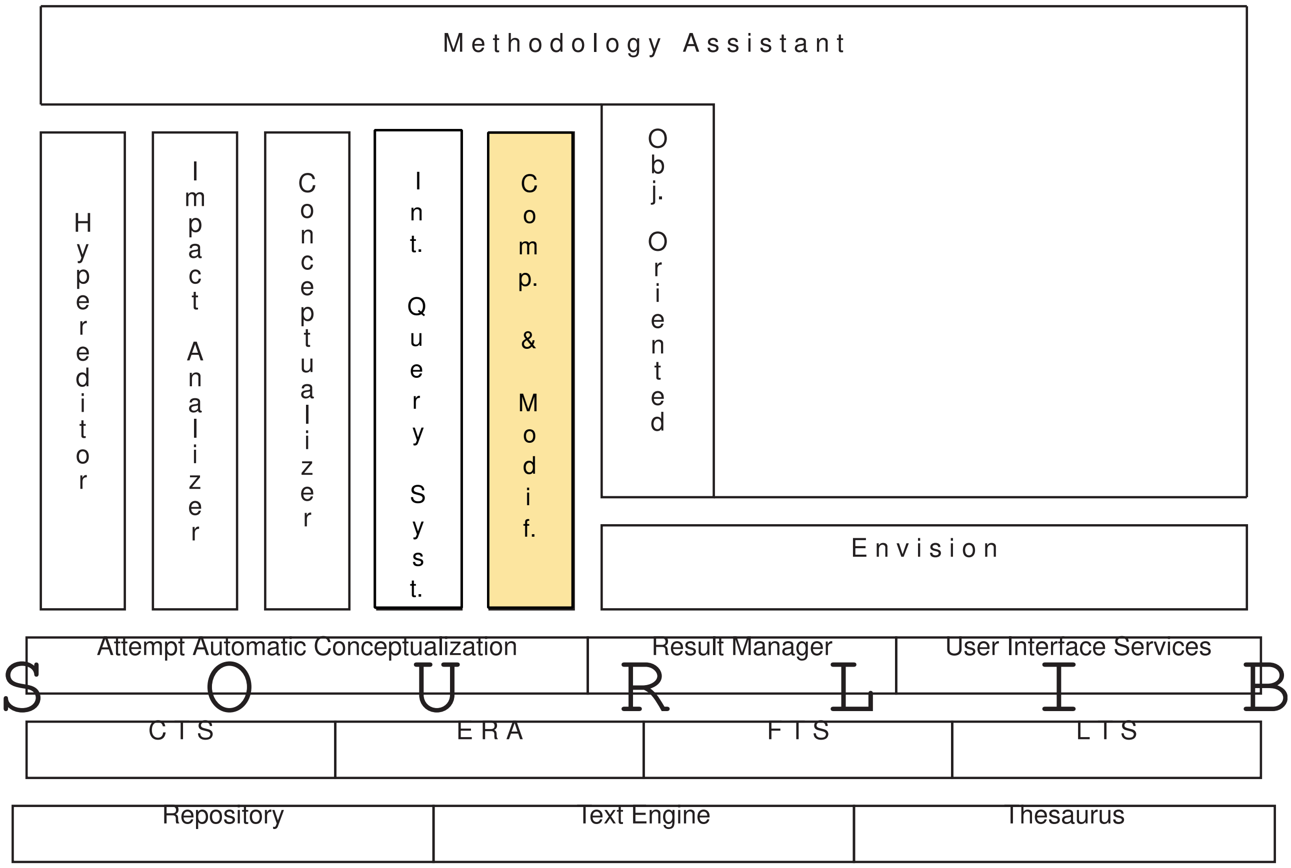
- R&D Contract *Eureka 379* — **SOUR**
- Partners: *Systena* (Olivetti Group - Rome), *Sss* (Olivetti Group - Bari, Italy), *Inesc* (Group 2361 - Braga, Portugal).
- Execution time: 1989–94.
- Budget: $\simeq$ 500 M Liras.

Main targets:

- **System**: CASE system able to promote industrial software re-use (SOUR=Software Use and Reuse).
- **Tools**: integrated support for the characterization, comparison, modification and re-use of code or other technical documentation.
- **Strategy**: “faceted” classification based on fuzzy-logic
- **Target platform**: Olivetti’s *Osa/R*(epository) running on *Windows 3.1*.

INESC Workpackages (see block diagram):

- *Con*(ceptualizer);
- *Iqs* (Intelligent Query System);
- *CMod* (Comparator & Modifier);
- *RM* (Result Manager).



Project Context

- Software re-use.
- Olivetti concern: a huge legacy of chaotic documentation about banking software
- INESC group 2361 motivation: application of formal methods to real life problems.

About INESC 2361

Where we are

- INESC's Group 2361 (*Formal Methods for Software Development*) is hosted by the Department of Informatics of the University of Minho, sharing its premises in the University's new campus at Gualtar, in the suburbs of Braga.

Main Target

- R&D to promote the adoption of *Formal Development Methods* in large-scale, industry-level software development.

Strategy

- A stepwise approach to software development based on calculating programs from formal specifications, animated by rapid prototyping tools and embeddable with conventional technology.
- Emphasis on interactive and/or distributed applications.
- Formalisms leading to automation

The Liaisons

- University of Bristol (Dept. of Engineering Mathematics)
- University of Zielona Gora (Poland)
- Universidade Federal de Pernambuco (Recife, Brazil)
- OGROUP (Olivetti Group, Bari, Italy)

Summary of Recent R&D Activity

- Formal Specification Case Studies
  - The Bristol Liaison
    - Barbosa
  - The Olivetti Liaison (SOUR)
    - Oliveira
    - Martins
    - Ribeiro
    - Neves
  - A Building Specification Language Experiment
    - Oliveira
- Dealing with human interaction
  - Fuzzy query languages
    - Martins
    - Ribeiro
  - Formalization of the UI-layer
    - Martins
    - Campos
- Rapid Prototyping
  - Embedding of formal models in pre-existing conventional systems
    - Neves
- CASE support
  - Graphical editor based on hardware-software analogy
    - Gulamhussen
  - Fuzzy classification of software components
    - Oliveira
    - Neves
  - Classifying Internet Objects
    - Neves
    - Oliveira
  - Formal software classification techniques
    - Oliveira
    - Ferreira
  - A SETS-based reusability engine using genetic algorithms
    - Neves
- Program Calculi
  - Handling “in loco” invariants in SETS
    - Oliveira
  - Extending SETS to Process Refinement
    - Barbosa
  - Relating SETS to other Program Calculi
    - Rodrigues

Background

- Traditional *software engineering* is admittedly an under-developed branch of Computing Science.
- Structured Programming (1970's), Declarative Programming (1970-80's) and Object-Oriented Programming (1980's) have failed to resolve the "software crisis".
- "Traditional" Formal Methods are not effective enough.
- Industry has been slow in adopting formal methods.

Main Concerns

- **Simplicity** — industry will never absorve formal methods based on complex mathematical theories.
- **Compatibility** — formal methods cannot replace traditional methods altogether.
- **Tools** — formal methods should be supported by tools and environments.
- **Flexibility** — formal method tools should be easily portable across different machine platforms and be able to communicate with existing (traditional) tools.
- **Modularity** — or common sense applied to software development; warning: informal modularity is worse than formal monolithic development.
- **Reusability** — requires a formal classification scheme; otherwise, repositories become full of things we will never find.
- **Back to "good engineering habits"** — in school Physics we are taught a universal strategy for problem solving:
  - understand the problem,
  - build a mathematical model of your understanding of it,

– reason in this model,
– upgrade your model, if necessary,
– calculate a solution

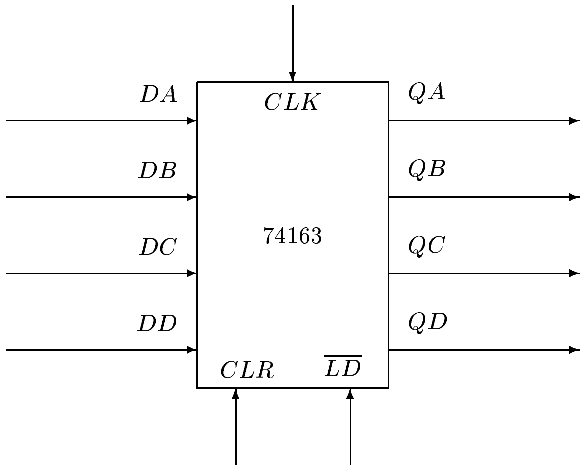
Why don't we do likewise in software development?

Main Ideas of CAMILA/SETS

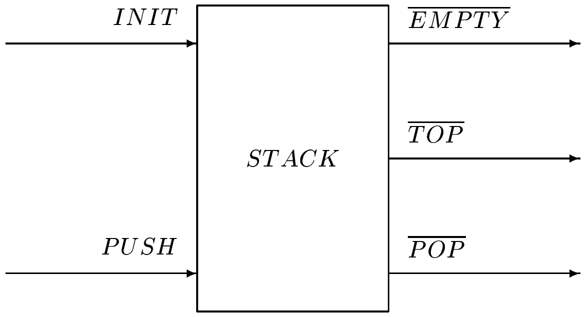
- **Simplicity** — the SETS notation is based on naïve set theory and the lambda-calculus.
- **Compatibility** — traditional notations such as DFDs and ERA-diagrams can be inverted to the SETS notation by calculation.
- **Tools** — CAMILA is a platform for supporting large-scale development of software using the SETS methodology (see slide 22); the toolkit is still in development but already comprises a rapid-prototyping tool, a structure-editor and a UI semi-automatic generator.
- **Flexibility** — the rapid-prototyping tool of the CAMILA environment is available on Unix, MS/DOS and Windows 95 and can be easily interfaced with other tools via C/C++; any CAMILA prototype can be a X-windows server or function like a Windows 95 DLL.
- **Modularity** — a version of the CAMILA rapid prototyping language is modular and may be based on a repository of software components formally classified using the SETS calculus.
- **Reusability** — in a top-down development style, the SETS calculus can be used to horizontally refine a given component and find isomorphic versions of its sub-components in the repository.

  In a bottom-up style, SETS component aggregation can be expressed by "software-circuit" diagrams, using a graphical notation suggestively resembling the conventional hardware notation.

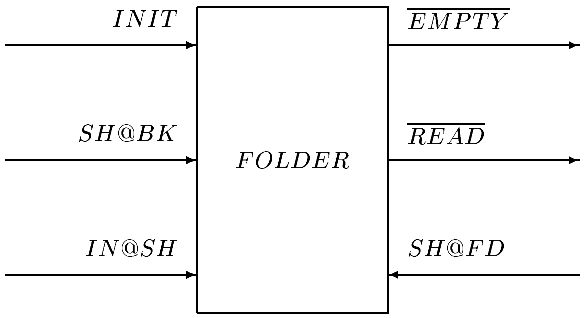
Compare, for instance...



with...

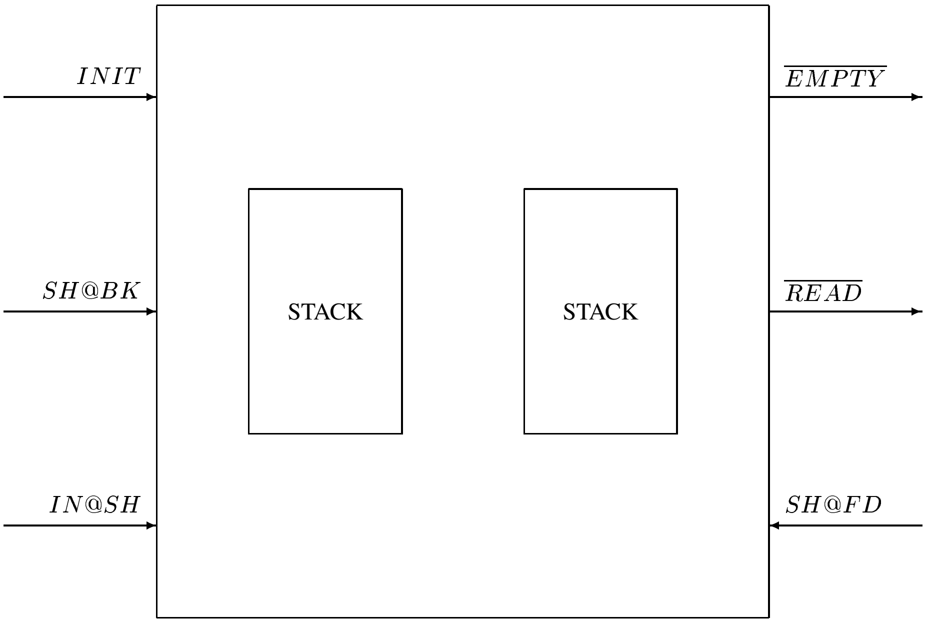


Suppose what we want is...

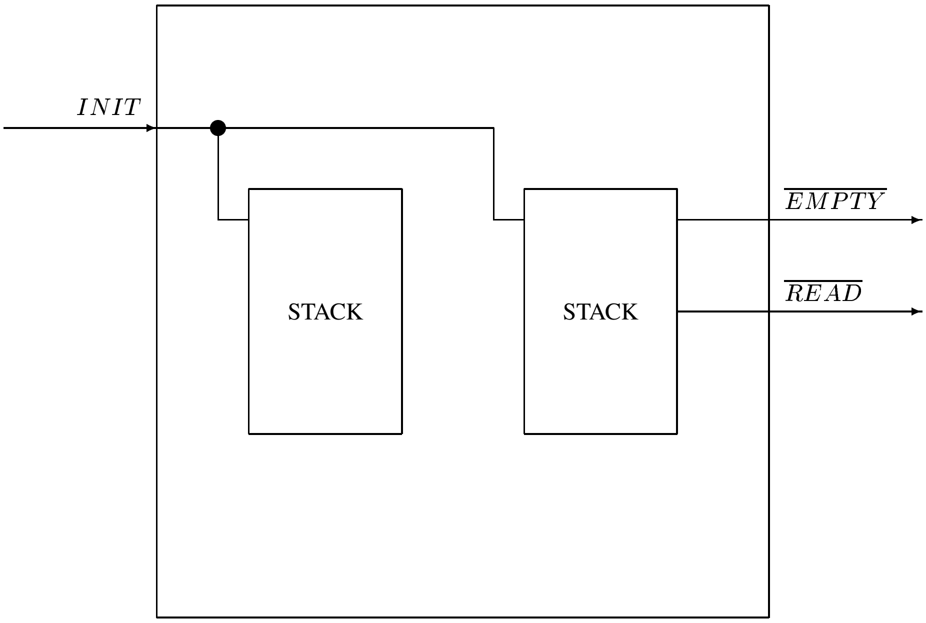


Reusability:

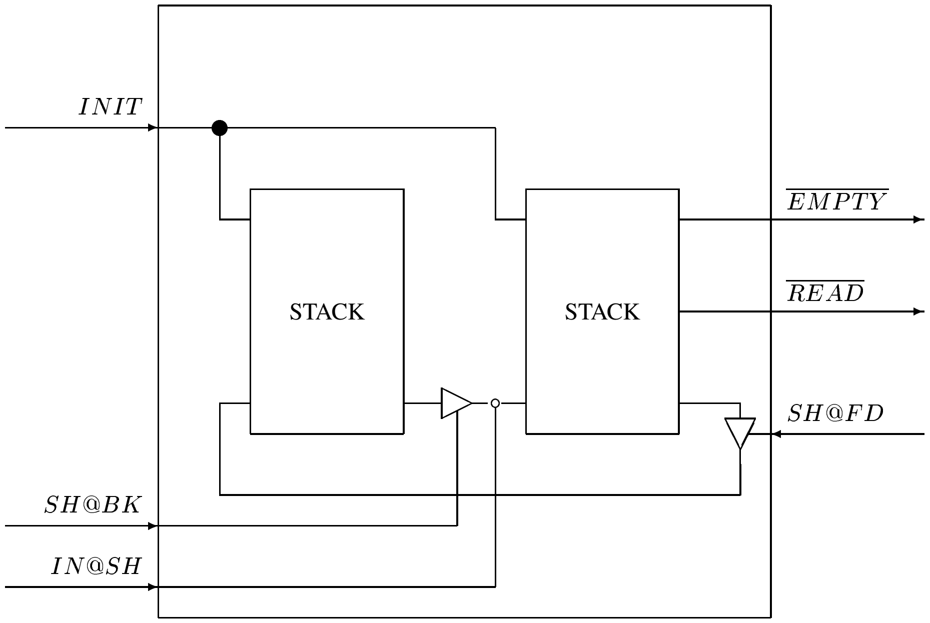
Do not start from scratch: "zoom in" e reuse $STACK$:



Some obvious connections first...



...then the elaborate ones...



...and that's all.

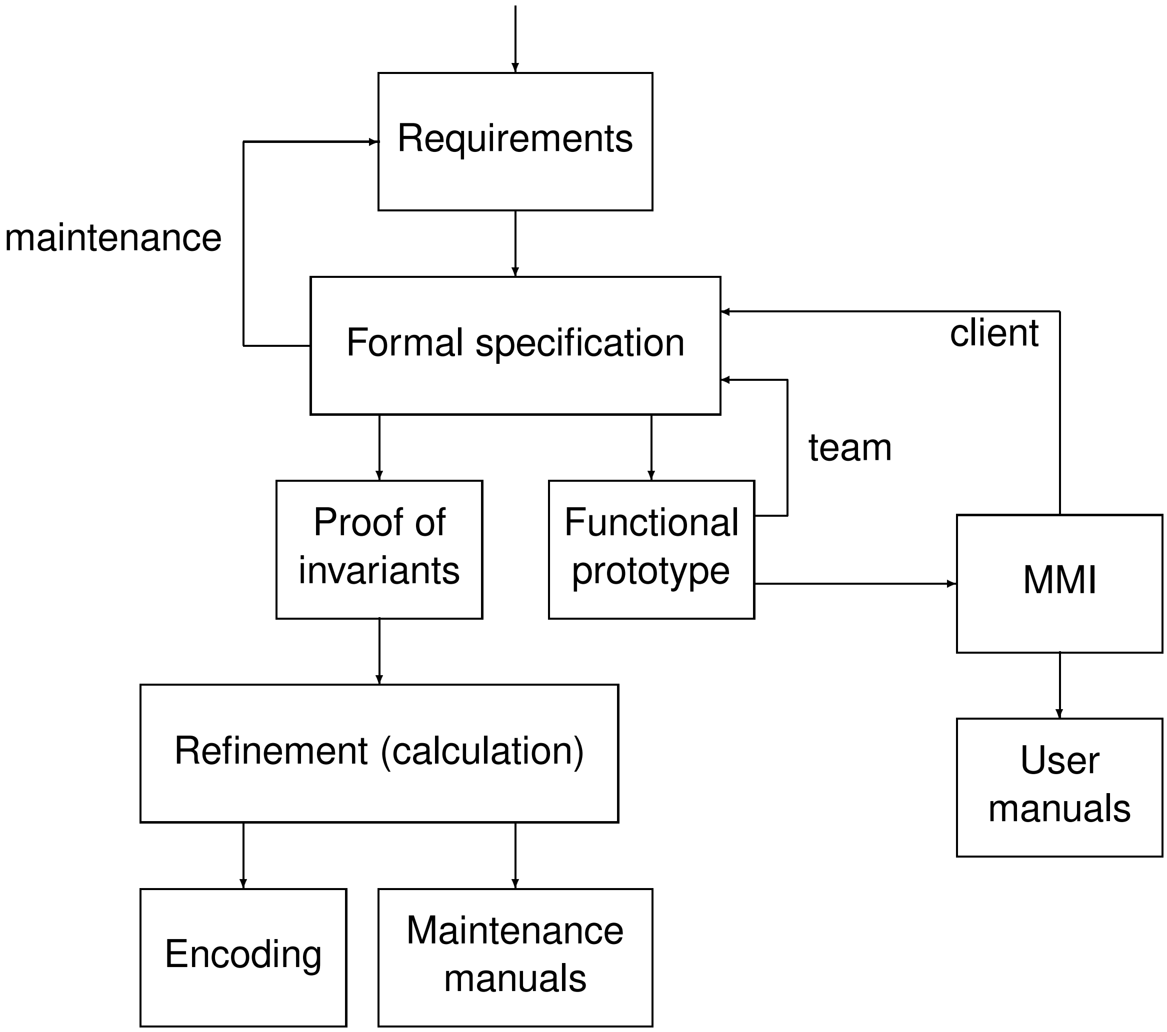
In summary:

$$\text{Software component} = \begin{cases} \text{Elementary component } (\textit{e.g. } STACK) \\ \textit{or} \\ \text{Software component network } (\textit{e.g. } FOLDER) \end{cases}$$

However, diagrams should never replace formalisms — every "software-circuit" has a proper formal semantics and is just a shorthand for some piece of mathematics.

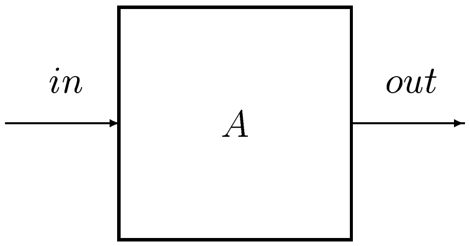
- **Back to "good engineering habits"** — the SETS reification calculus is used throughout development to calculate possibly distributed solutions (implementations) to specification models — *cf.* the software life-cycle of next slide.

Software Life-cycle using CAMILA/SETS:

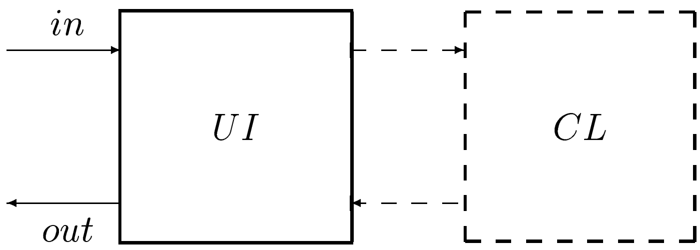


Client-server software architecture

The *end user*'s viewpoint…



…is different from the system's internal architecture:



A	-	application
UI	-	user interface (client)
CL	-	computational layer (server)

Seeheim "Separation Principle"

- UI and CL should be developed in isolation
- CL provides API (application program interface) for "batch service"
- (UI will be eventually discarded)

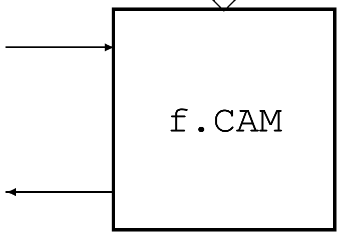
Separation Principle versus Formal Specification and Rapid Prototyping

- Formal specification of CL first
- First version of UI hides rapid prototype of CL formal specification

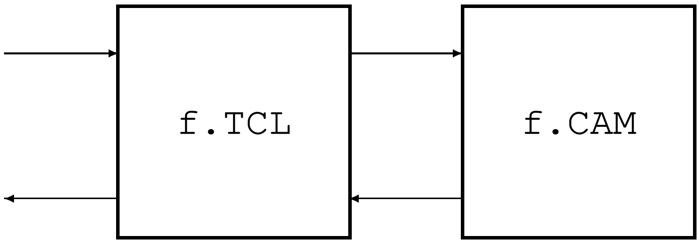
- Improve UI and/or CL in isolation
- Calculate CL implementation and replace prototype

f.DOC = Pretende-se uma aplicação para a automatização do sistema de gestão da produção da fábrica .. bla. . . bla. . . bla. . . . .

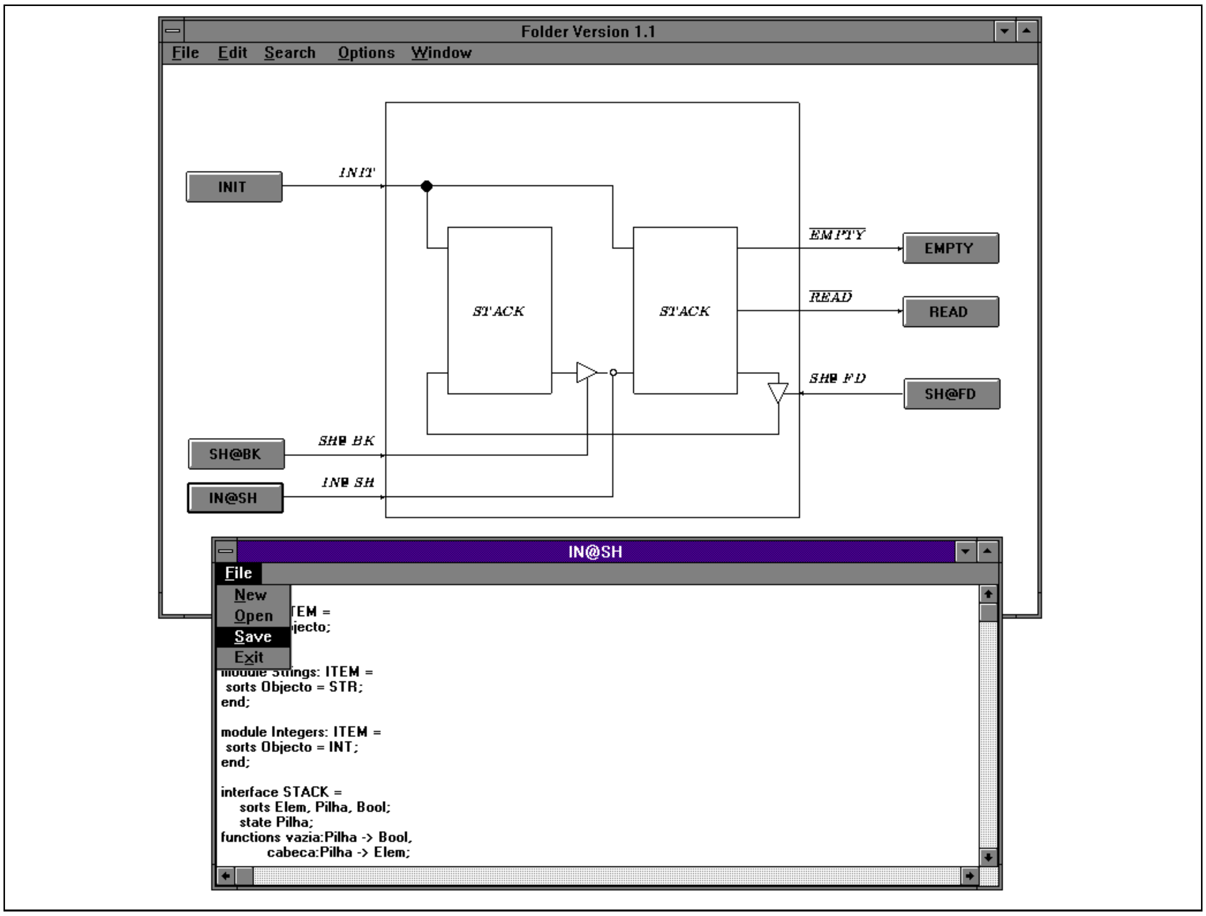
$$\text{f.SETS} = \begin{cases} s = x \times l \\ x = b \times (x+1)^a \\ l = 1 + a \times l \end{cases}$$



Adding the UI. . .

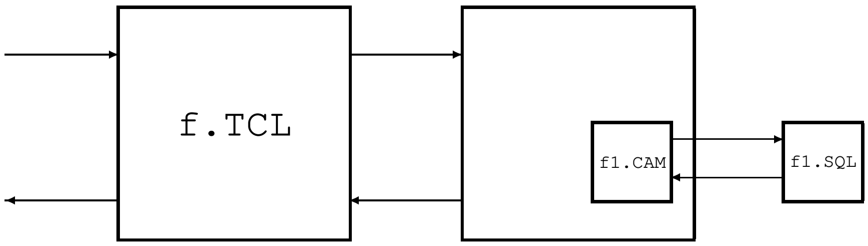


. . . is a game we play with the “end user” which normally proves very useful:

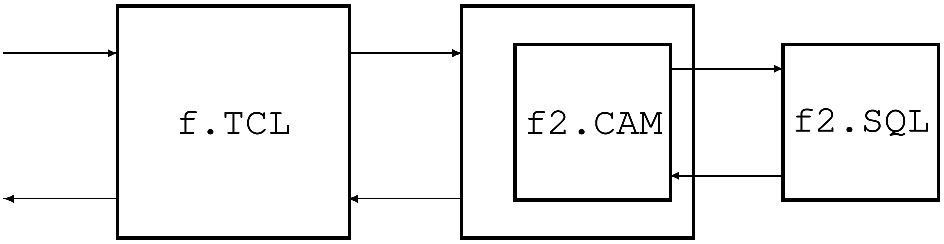


Implementation

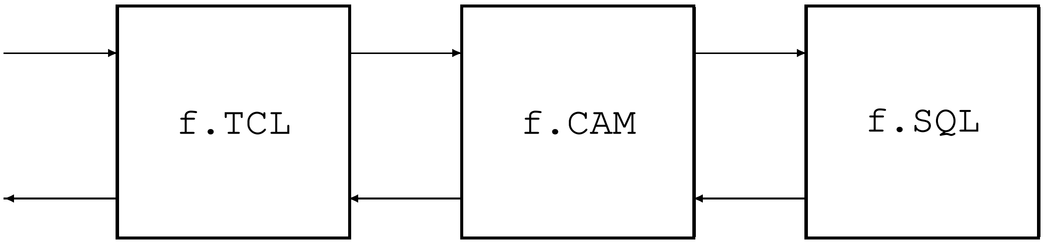
Once the behaviour of the prototype pleases the project contractor, start with implementation calculation. . .



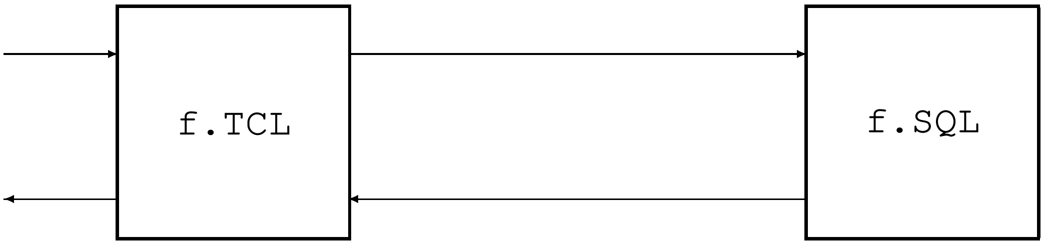
. . . in a stepwise manner. . .



. . . until. . .



Eventually. . .



. . . the prototype dies out and gives room to all final code.

Comments:

(a) “Low energy” development cycle looping around the *formal specification* and far away from final code.

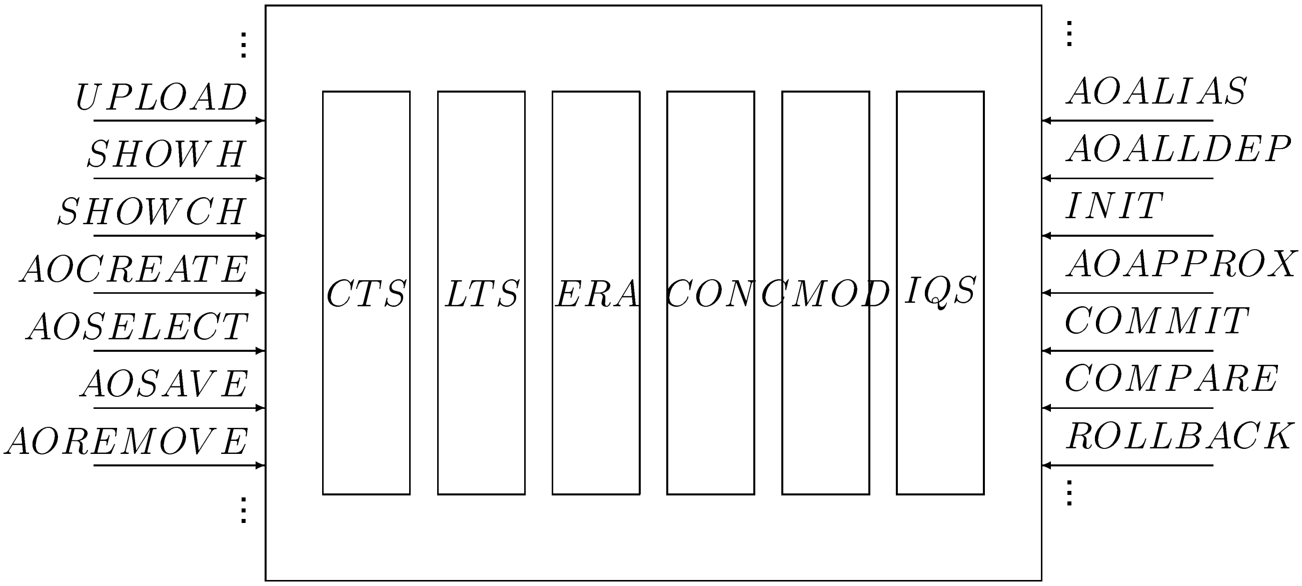
(b) Fully operational rapid prototype (no “stubs”) runs hidden behind an independent presentation layer.

(c) Implementation is formally calculated in a stepwise, “horizontal” flow.

(d) Documentation is “inherently” produced by the method.

Back to the Industrial Case Study

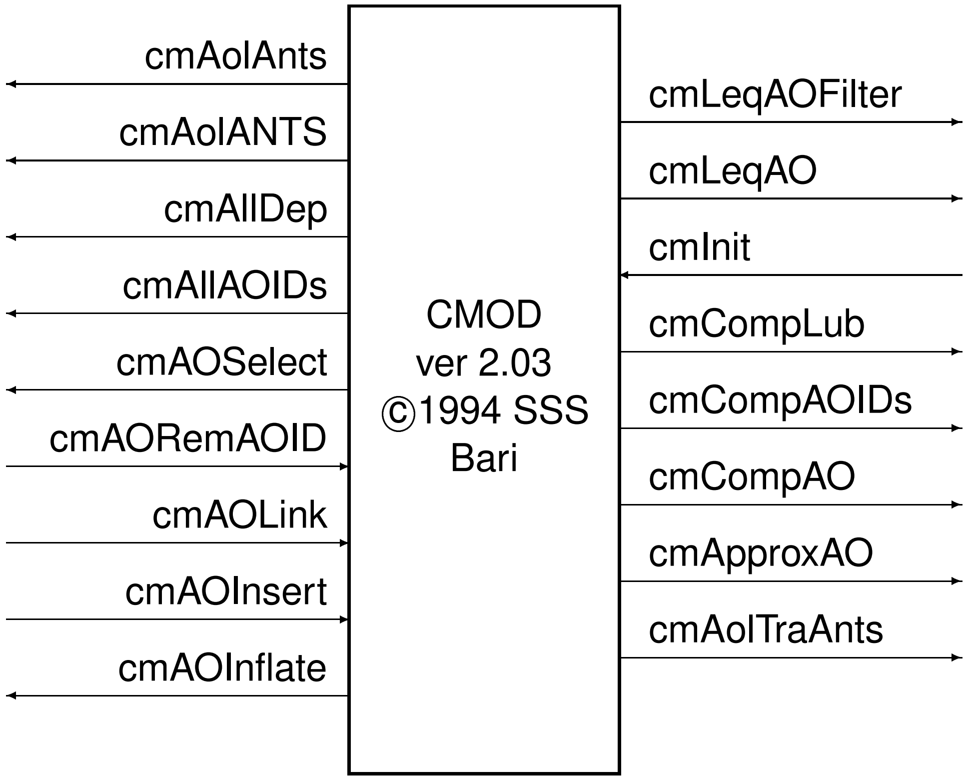
- The overall architecture of SOUR was captured in terms of software system "circuitry", such as (roughly!) sketched below:



- INESC's modules
  - CON (Conceptualizer),
  - $CMOD$ (Comparator and Modifier) and
  - IQS (Intelligent Query System)

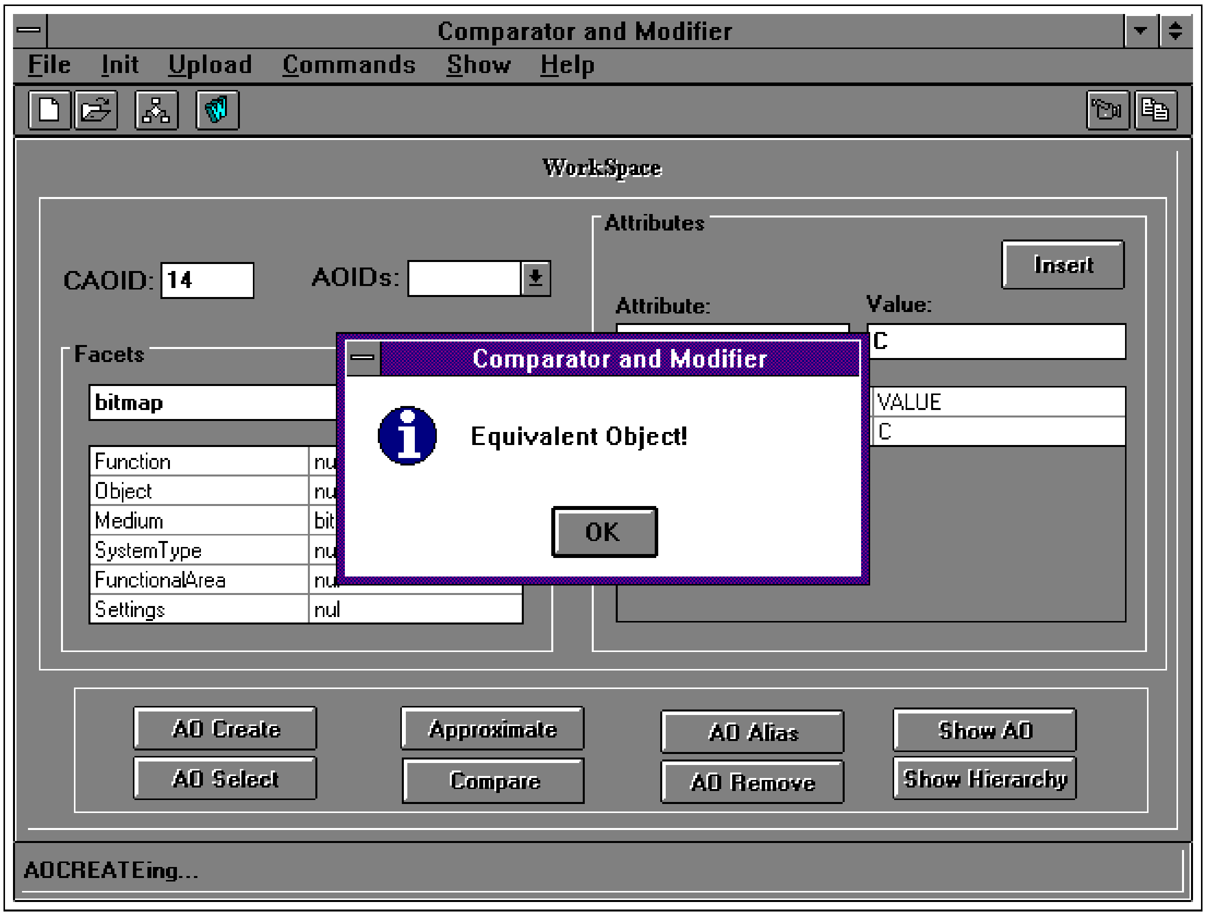
  were formally specified in CAMILA/SETS notation.

  Example of one of the "software ICs" developed at Braga for the project:



- Thanks to the CAMILA rapid prototyper, in the project's first technical meeting these modules were shown "running" behind a tentative user interface (see next slide), when nothing but functional specs were expected.
- The underlying
  - *Era* (Easy Repository Access),
  - *Lts* (Lexicon Thesaurus Subsystem) e
  - *Cts* (Concept Thesaurus Subsystem)

  modules — still under development in Italy at that time — were integrated in the (then rather slow but still very useful!) prototype by a CAMILA animation of the *reverse specification* of their C/C++ APIs, which had already been written down.
- Very early feedback on the hard technical subtleties of the design (as unveiled by the formal specification) could be obtained by INESC's partners inspection of this prototype's behaviour.



- Eventually, all main implementation decisions could be validated by calculation (in SETS). In the system's final prototype, running on Windows 3.1, all CAMILA code gave room to C/C++ code.

Some SOUR figures concerning Braga

(As recorded by October 1994):

- Number of lines of CAMILA code, including reverse specification of the Italian modules: . . . . $\simeq$ 4 200
- Number of lines of C/Visual Basic: . . . . $\simeq$ 31 400

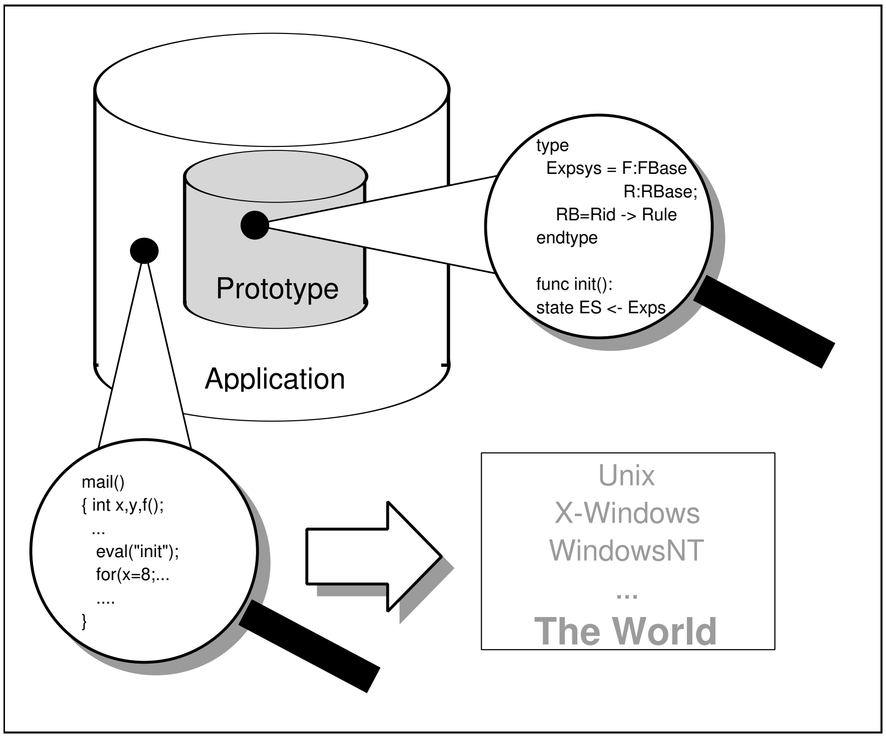
However…

- Because INESC's partners were software companies neither trained nor interested in formal methods, the CAMILA/SETS approach has not been made oficial in the project.

- Final documentation was delivered in plain English by transliterating all mathematics into “natural language”.

More About CAMILA

- CAMILA/SETS animator is based on a fairly elaborated X-LISP extension:



- Braga experience: client-server, information systems
- Bristol experience: network systems
- CAMILA Home Page and Users Group:

  See `http://camila.di.uminho.pt/`.

Final Remarks

- By using formal methods and rapid prototyping, INESC could save about 8 months of programming work (as compared to the initial schedule).
- INESC's team was quite small (1 Ph.D. + 1 M.Sc. + 2 undergrads) but well trained in formal methods and rapid prototyping.
- Taking place when R&D and teaching in formal methods at Minho University was 10 years old (1984–94), SOUR helped us to learn how to survive "formally" in a hostile environment.
- SOUR remains INESC 2361 Group's best experience on collaborating in heterogeneous consortia involving partners unused to formal software design methods.

References

[Almeida & Barbosa 91] Almeida J., Barbosa L.S. *CAMILA by Example.* Lecture Notes, Technical Report 91-11-1, DI/INESC, November 1991.

[Almeida et al 97] Almeida J.J., Barbosa L.S., Neves F.L., Oliveira J.N. *CAMILA: Formal Software Engineering Supported by Functional Programming.* Submitted to DPLE'97: First International Conference on Declarative Programming Languages in Education, Southampton, Sep. 3-5 1997.

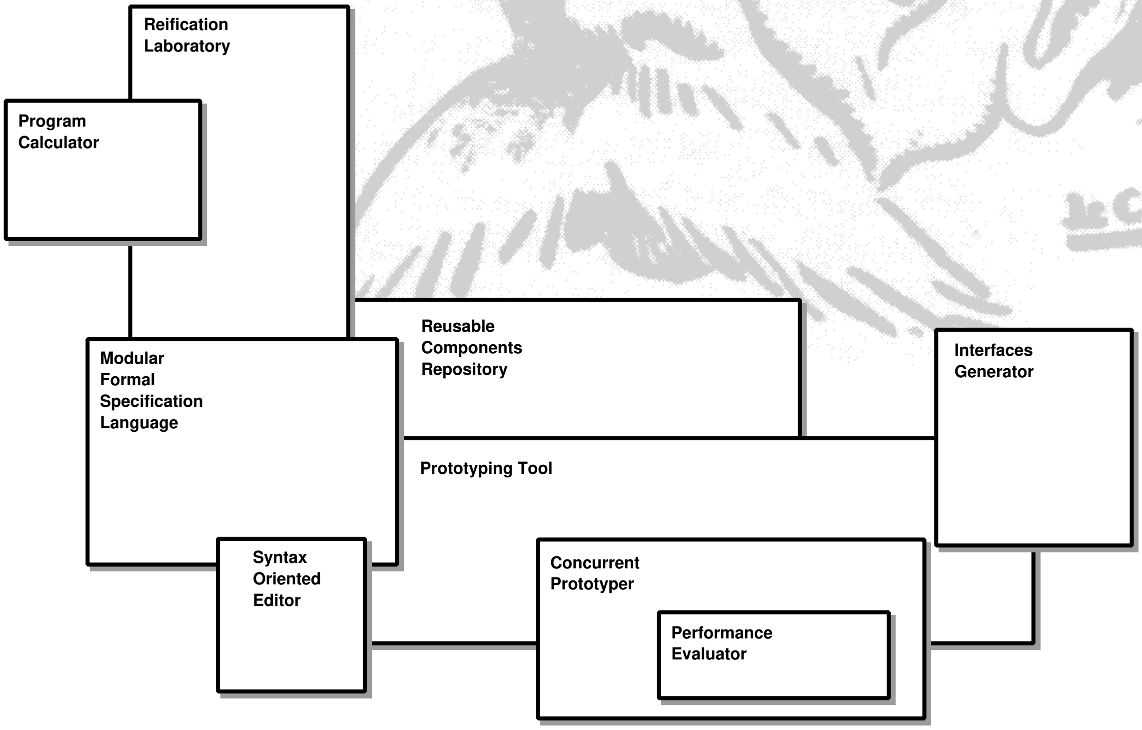
[Oliveira 90] Oliveira J.N. *A Reification Calculus for Model-Oriented Software Specification.* Formal Aspects of Computing, Vol.2, 1-23, 1990, Springer-Verlag.

[Oliveira 92] Oliveira J.N. *Software Reification using the SETS Calculus.* Invited communication. Proc. of the BCS FACS 5th Refinement Workshop, Theory and Practice of Formal Software Development, London, UK, 8–10 January 1992, 140-171, Springer-Verlag.

CAMILA

A System for the Mathematical Development of Software

- Formal Specification
- Rapid Prototyping
- Implementation Calculator
- Modular Development & Component Reuse
- Orientation to Concurrency



CAMILA

Research Domain: ***Programming Foundations***

Project: ***DI - INESC***

Coordinator: ***Prof. José Nuno Oliveira***

Support: ***JNICT - PMCT***

Estou agora a ver se ajunto umas migalhas para comprar um cardenho entre dois carvalhos aí para o coração do Minho, e fechar-me lá, resguardar-me dos lobos e dos homens e das mulheres, com duas canastras de livros.

Camilo Castelo-Branco