Inria
INVENTEURS DU MONDE NUMÉRIQUE

Panel "Experiments in Computer Science: Are Traditional Experimental Principles Enough? »

November 2012

Some questions

- What are the « traditional experimentations principles » ?
- What are the specificities of experimentations in Informatics, if any ?
- Who should do the experimentations ?
- How experimental work can be evaluated ?

What are the « traditional experimentations principles » ?

- Experimentations help to understand the behaviors of the (physical) objects under study

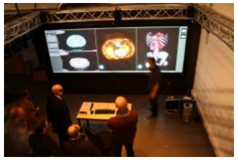
- Experimentations need huge, (very) costly research infrastructures
  - Animal houses, Space telescopes, Boats, Satellites
  - Large Hadron Collider,…

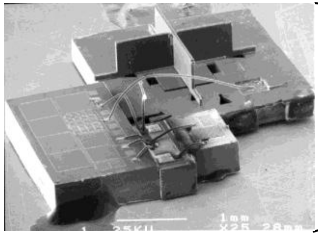
- Experimentations need most often the help of technicians and engineers.

What are the specificities of experimentations in Informatics, if any ?

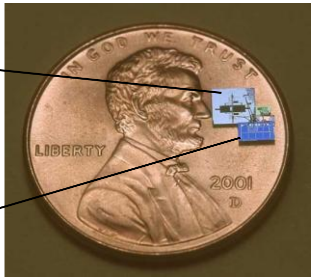
- No, specificity since informatics is a science like the others !

Immersion systems

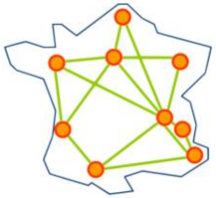




Smart dust



Grid computing



Robots

What are the specificities of experimentations in Informatics, if any ?

- But Software !

- We use software as astronomers use telescopes but to study and “construct” software is also part of our job, whereas astronomers do not study or construct telescopes.

- Our experimentation tools (software, robots,…) are also likely to create value and thus to transfer towards industry.

Who should do the experimentations ?

- No (clear) separation between experimentalists and theoricians (which is not the case for physicists, for instance)

- Do we need specialized engineers / developers for software developments?

  - YES !

    Indeed, we have more 70 of them in Inria

    Essential to maintain huge and perennial soft

  - NO !

    There are almost no such guys elsewhere !

software development
How ~~experimental work~~ can be evaluated ?

- A key point since it has to be evaluated, at least when researchers do it
- No (or almost none) journals or conferences devoted to software
- Difficulties, even though peer review, to take them into account

Proposal of Criteria for Software Self-Assessment

1. Characterize the software

2. Characterize your Own Contribution

3. Additional information

Proposal of Criteria for Software Self-Assessment

1. Characterize the software

    1.1 Audience (A)

    1.2. Software Originality (SO)

    1.3. Software Maturity (SM)

    1.4. Evolution and Maintenance (EM)

    1.5. Software Distribution and Licensing (SDL)

Characterize the software (1/5)

Audience (A)

1. personal or internal team prototype (to experiment an idea);
2. to be used by people in the team or close to the team (including contractual partners);
3. ambitious software, usable by people inside and outside the team but without a clear and strong dissemination and support action plan;
4. large audience software, usable by people inside and outside the field with a clear and strong dissemination, validation, and support action plan;
5. wide-audience software (aims to be usable by a wide public, to become the reference software in its area, etc.).

Characterize the software (2/5)

Software Originality (SO)

Here by ideas we mean algorithms, programming techniques, GUI, interfaces, …

1. none;
2. minor contribution to existing software, reusing known ideas;
3. original software reusing known ideas and introducing a few new ideas;
4. original software implementing a fair number of original ideas.

Characterize the software (3/5)

Software Maturity (SM)

1. demos work, rest not guaranteed, loose documentation, no real software engineering;
2. basic usage should work, terse but usable documentation, some software engineering, basic bug fixes done from time to time;
3. well-developed software, fairly extensive documentation, reasonable software engineering and testing, attention to usability, dissemination, bug fixes, and user feedback;
4. major software project, strong attention to functionality and usability, extensive documentation, strong software engineering, systematic bug chasing, and regression testing;
5. high-assurance software, certified by an evaluation agency (Common Criteria, DO-178, etc.) or formally verified.

Characterize the software (4/5)

Evolution and Maintenance (EM)

1. no real future plans;
2. basic maintenance to keep the software alive;
3. good quality middle-term maintenance, with persistent attention to users;
4. well-defined and implemented plan for future maintenance and evolution, making it possible for users to use the software without risk for important projects, organized users group.

Characterize the software (5/5)

Software Distribution and Licensing (SDL)

1. none;
2. basic source or binary distribution to the team or close community;
3. distribution to an industrial partner in a contractual setting and where the software is actually used;
4. public source or binary distribution on the web, organized by the development team;
5. External packaging and distribution: either as part of a popular open source distribution (e.g. a Linux distribution, an algorithmic or scientific library) or packaged within a commercially distributed product (Matlab, etc.).

Some questions as « conclusion »

- Do we need professional developers for software developers to “help” researchers ?

- Any good idea on how to evaluate software ? `

- Software is not all !
  - Do we have to develop an “ experimental dimension “ to Informatics ?
  - And so, to have also our huge and costly research infrastructures ?