# OUT OF THE TAR PIT

## * Complexity

· building software is hard because of complexity, conformity, changeability and invisibility

↳ complexity is considered the root cause of the vast majority of problems: late delivery, lack of security, unreliability...

## * Approaches to understand

- testing: attempting to understand a system from the outside, from how it behaves in certain situations

- informal reasonning: attempting to understand by examining the code

=> testing doesn't resolve every problem as we settle the inputs. We don't know how it will behave if the inputs don't mactch the one we tested.

## * Causes of complexity

→ state: makes programs hard to understand

↳ test on a system doesn't give any clue a the particular state we can not always force the system into a "good internal state".

↳ if procedure makes use of any procedures that are stateful than it is contaminated. and we can only understand it in the context of the state.

→ control: order in which things happen.

↳ from control intervene concurrency

→ code volume

→ duplicated code, dead code, unnecessary abstraction...

## * Classical approaches to managing complexity

- Object-Orientation programming
  - ↳ the same bit-of-state can be manipulated/access by different procedures
  - ↳ when mutability is not required object identity doesn't make sense. It's then better to use custom access procedures.

- Functional programming
  - ↳ avoid states
  - ↳ more abstract use of control using functionnals rather than explicit looping.
  - ↳ problems arise when the system to be built must maintain state of some kind.

- Logic programming
  - ↳ process the program in the same order as it is read

## * Accidents and essence

essentiel complexity is inherent in, and the essence of, the problem
accidential complexity is all the rest
- ↳ essential means essential to the user's problem
- ↳ what is essential to the team is what the users have to be concerned with.