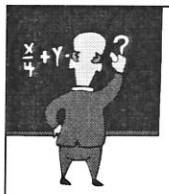
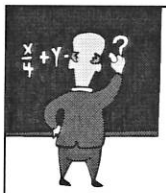
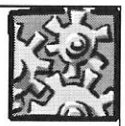


Some thoughts on Theories of Software Testing

Marie-Claude Gaudel
Emeritus Professor
LRI, Univ Paris-Sud & CNRS





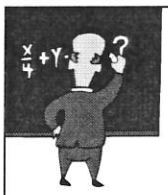
The long quest of a theory of software testing…

A pioneering paper:

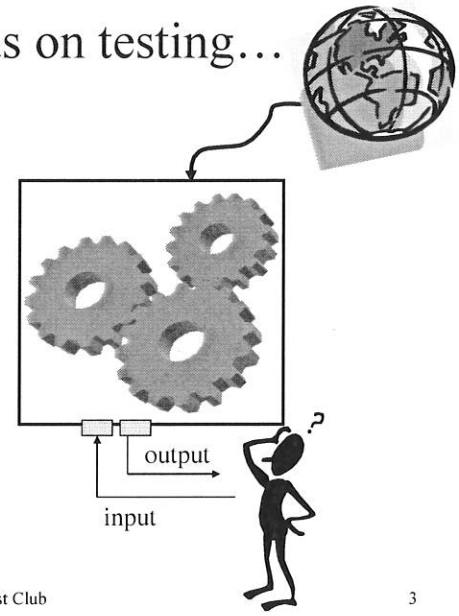
- *« We know less about the theory of testing, which we do often, than about the theory of program proving, which we do seldom »*

Goodenough J. B., Gerhart S.,
IEEE Transactions on Software Engineering, 1975

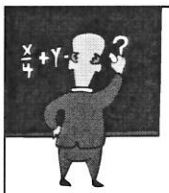
And then many others…



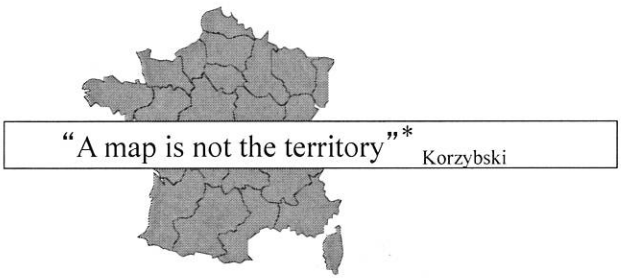
A few words on testing…



- One tests SYSTEMS
- A system is a dynamic entity, *embedded in the physical world*
- It is *observable* via some limited interface/procedure
- It is not always *controllable*
- It is quite different from a *piece of text* (formula, program) or a *diagram*

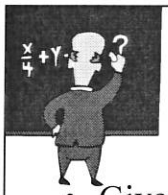


A philosophical interlude



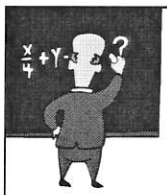
*A variant: "don't eat the menu…" ☺

A program text, or a specification text, or a model, is not the system

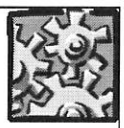


Some keywords

- Given :
  - some specified **requirement** expressed as a text, formula, diagram,
  - an executable **SUT**,
  - some **knowledge** on the SUT and its environment
- The aim is to detect deviations of the SUT w.r.t. the specified requirement via
  - the **selection** of "representative inputs", their **submissions**, and the **verdict** by an "oracle"
- And to assess the confidence obtained after a test campaign

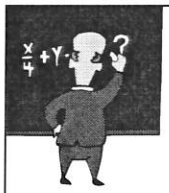


Histoire du domaine



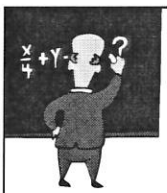
- Un premier article fondateur
  - Goodenough J. B., Gerhart S., *Toward a theory of test data selection*, IEEE Transactions on Software Engineering, vol. SE-1, n° 2, June 1975
- Notion de critères de sélection de tests
  - COMPLETE(T, C) <=> le jeu de test T satisfait le critère C ; D est l'ensemble des données.
  - critères "fiables" et "valides"
    - **RELIABLE**(C) = ($\forall T_1, \forall T_2 \subseteq D$) (COMPLETE($T_1$, C) $\wedge$ COMPLETE(T_2, C)) $\Rightarrow$ (SUCCESSFUL(T_1) $\Leftrightarrow$ SUCCESSFUL(T_2))
    - **VALID**(C) = ($\forall d \in D$)($\neg$OK(d) $\Rightarrow$ ($\exists T \subseteq D$) (COMPLETE(T, C) $\wedge$ $\neg$SUCCESSFUL(T)))

Oracle?



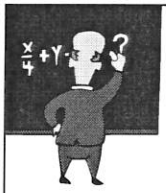
[GG75] inventaire

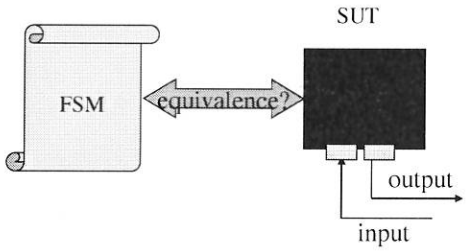
- Théorème : COMPLETE(T, C) ∧ RELIABLE(C) ∧VALID(C) ∧ SUCCESSFUL(T) ⇒ (∀d ∈ D)(OK(d))
- Classification des fautes de programmation
- Notion de test exhaustif et de partition en "classes d'équivalence" du domaine des entrées
- Comparaison entre test et preuve
- Test « boîte noire » basé sur des "condition tables"



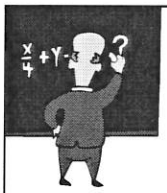
Histoire, suite

- Un deuxième article fondateur
  - Chow T. S., *Testing Software Design Modeled by Finite-State Machines*, IEEE Transactions on Software Engineering, vol. SE-4, n° 3, May 1978
- Test de systèmes dont la structure de contrôle peut être modélisée par une FSM (Finite-State Machine) déterministe
- Un théorème d'exhaustivité

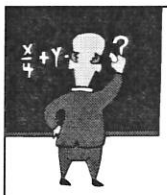




FSM : Finite State Machine
SUT : System Under Test
Exhaustivity under the assumption that the SUT behaves like a FSM with the same number of states.

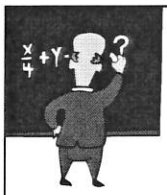


MAIN CONCEPTS

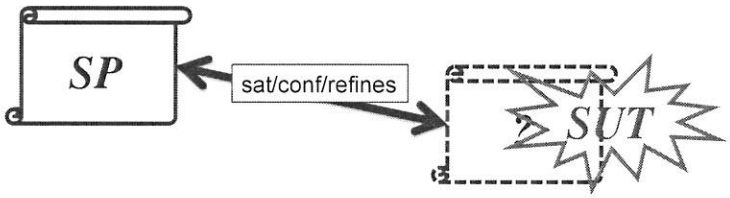


Wanted: a satisfaction/ conformance relation

- Given some "testable" *SUT*, what does it mean that it satisfies *SP*?
- What is the correctness reference? Is there an "exhaustive" (or "complete") set of tests?
- *SP* is some sort of *model or formula*; *SUT* is some sort of *system*; how to define *"SUT sat SP"* or *"SUT conf SP"* in such an heterogeneous context?

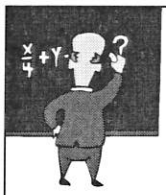


Back to well-established relations



For instance, the *satisfaction/conformance* relation is

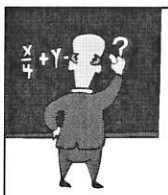
- equivalence for FSM,
- logical satisfaction for formulas,
- refinement for processes,
- *ioco* for LTS…



Specification-based Testing: underlying hypotheses

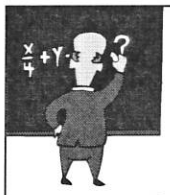
- The internal organisation of the SUT (System Under Test) is not considered, indeed…
- *However,*
  - Implicitly or explicitly, one considers a class of "testable implementations" =>
  - Notion of ***Testability Hypotheses*** on the SUT
  - The SUT behaves like a model of the specification formalism.

Often implicit, but always there!



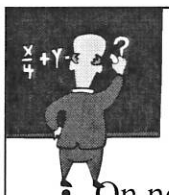
Testability in general?

- If the SUT can be *any demonic system*, there is no sensible way of testing it ☹
- Fortunately, *some basic assumptions are feasible* (example: correct implementation of booleans and bounded integers, determinism, …)
- Some others can be *verified in another way*: static checks on the program, preliminary tests, a priori knowledge of the environment…



Non-déterminisme…

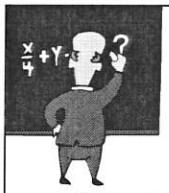
- Que faire en cas de non-déterminisme du SUT?
- "Complete Testing Assumption"
  - Cas particulier : déterminisme
- Sélection au vol
- Instrumentation du SUT ou/et de son environnement (scheduler, etc)



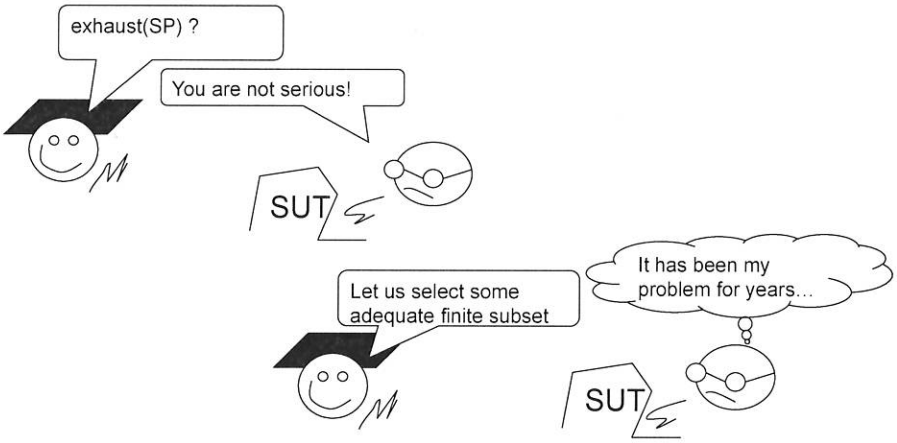
Test et correction

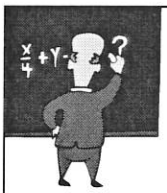
- On note *SUT passes* $Exhaust_{SP}$ le fait que *SUT* passe tous les tests de $Exhaust_{SP}$ avec succès
- On a

  $H_{min} \Rightarrow ([\![SUT]\!] \models SP \Leftrightarrow SUT \text{ passes } Exhaust_{SP})$
- Autrement dit, *sous l'hypothèse de testabilité*, il y a succès du test exhaustif si et seulement si le SUT satisfait *SP*
- Mais comment sélectionner des sous-ensembles finis de $Exhaust_{SP}$?



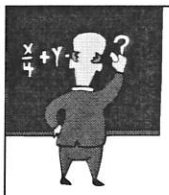
Its nice to have some theorems, but exhaustivity is not practicable…





Sélection et Hypothèses

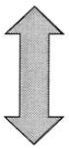
- On fait des hypothèses PLUS FORTES sur *le SUT*
- Exemple : *Hypothèse d'Uniformité*
  - $\Phi(X)$ formule, D sous-domaine
  - $(\forall t_0 \in D)([\![SUT]\!] \models \Phi(t_0) \Rightarrow (\forall t \in D)([\![SUT]\!] \models \Phi(t)))$
- Détermination des sous-domaines ? *guidée par la spécification.*
- Autre exemple : *Hypothèse de Régularité*
  - $((\forall t)(|t| \leq k \Rightarrow [\![SUT]\!] \models \Phi(t))) \Rightarrow (\forall t)([\![SUT]\!] \models \Phi(t))$
  - Détermination de |t|? *cf. spécification*



Le choix des hypothèses?

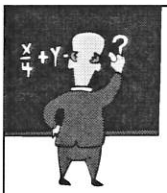
<SUT testable, exhaust(SP)>

<Hyp. Faible, Gros Jeu deTest >



<Hyp. Forte, Petit JT>

<SUT correct, ∅>



En fait…

- Les hypothèses de sélection
- Les critères de test
- Les modèles de fautes

Robert M. Hierons:
Comparing test sets and criteria in the presence of test hypotheses and fault domains.
ACM Trans. Softw. Eng. Methodol. 11(4): 427-448 (2002)