

Propriétés indésirables des entités architecturales

Défauts et anomalies de fonctionnement, correction des logiciels

3.1 DÉFAUTS ET ANOMALIES DE FONCTIONNEMENT DES ENTITÉS ARCHITECTURALES

Dans les sciences de l'ingénieur classiques, les matériaux et les phénomènes que l'ingénieur utilise à son profit présentent tous des défauts, des anomalies temporaires ou permanentes qui ne doivent pas empêcher les machines de fonctionner ou d'accomplir leur service. C'est une contrainte inhérente à la nature du monde physique que l'on a su progressivement théoriser. La résistance des matériaux [RDM] est la science de l'ingénieur qui étudie les conditions de rupture de tous les matériaux que nous utilisons. C'est grâce à la résistance des matériaux que l'on a pu optimiser la quantité de matière à utiliser pour faire un pont qui tient.

En constructions civiles, en mécanique, en électrotechnique, les ingénieurs sont familiers depuis toujours des phénomènes d'usure qui progressivement dégradent les conditions de bon fonctionnement des machines ou des constructions, selon des lois que l'on sait modéliser, du moins pour les matériaux connus depuis un certain temps. Chacun se rappelle les problèmes rencontrés avec les premiers ouvrages en béton précontraint ou les premiers ponts suspendus exposés à des vents violents. Pour surveiller ces différents phénomènes les ingénieurs ont intégré dans leur production toute une mécanique de surveillance qui permet de suivre dans le temps le comportement des matériaux et prévoir l'arrivée des pannes.

Dans les sciences de l'ingénieur qui traitent de l'information, la théorie de l'information, créée par l'ingénieur C. Shannon, née avec les premières communications et le radar, est la première science entièrement fondée sur le traitement de

l'erreur. R. Hamming¹, l'inventeur des codes qui portent son nom, disait : « *Most bodies of knowledge give errors a secondary role, and recognize their existence only in the later stages of design. Both coding and information theory give a central role to errors and are therefore of special interest, since in real-life errors (noise) is everywhere.* » L'ingénieur O. Heaviside inventa une aberration mathématique, le calcul opérationnel (ou calcul symbolique), pour calculer les propagations des signaux électromagnétiques dans les conducteurs que L.Schwartz expliqua rigoureusement cinquante ans plus tard avec la théorie des distributions². En fonction de l'amortissement du signal, cela permettait de calculer le nombre de répéteurs à placer sur la ligne de communication. Qu'en est-il des abstractions informatiques issues du travail des concepteurs, des architectes et des programmeurs ? Comment garantir la sûreté de fonctionnement d'une chaîne de traitements qui n'est jamais qu'une propagation d'information doublée d'un calcul entre un acteur émetteur et un acteur récepteur ?

Toutes les abstractions informatiques, y compris la logique câblée dans les microprocesseurs, sont créées de toutes pièces par l'homme, et de ce fait soumise à l'erreur humaine. Dans les abstractions mises en œuvre pour réaliser l'ordinateur et les programmes qui lui donneront son « intelligence » (système d'exploitation, progiciels, programmes utilisateurs), seul le langage diffère, et la malédiction de Babel³ est déjà au cœur de notre construction : comment assurer la cohérence d'un ensemble hétéro-logique aussi hétéroclite ? Le totalitarisme de la langue informatique unique rend pensable, donc possible, les entreprises les plus arrogantes qui toujours échouent (ALGOL, PL1, Ada, PCTE, l'IA, etc.) ; la sagesse, c'est le principe de subsidiarité qui implique l'interopérabilité et la recherche de conventions sémantiques (qu'est ce que ça fait, à quoi ça sert !) partagées et acceptées par tous.

La qualité des intégrats définis par l'architecte et assemblés dans le processus d'intégration, est directement fonction de la qualité du travail du programmeur pris dans un sens extensif : un ingénieur électronicien qui programme un SOC (*System On Chip*) est un programmeur, un ingénieur qui réalise le paramétrage d'un PGI comme SAP pour le compte d'un client X est également un programmeur, un usager qui personnalise son poste de travail est un programmeur. Un ingénieur en aéronautique qui programme une loi de pilotage pour l'Airbus A380 est évidemment un programmeur. Chacun de ces programmeurs est sujet à l'erreur.

Il est donc fondamental de comprendre la relation complexe que le langage informatique utilisé entretient avec la machine sous-jacente et avec l'utilisateur du langage, et de prendre en compte la présence inévitable de défauts dans les logiciels (s'il n'y en a pas, c'est tant mieux !).

1. Dans, *Coding and information theory*, Prentice Hall, 1980.

2. Voir son livre autobiographique : *Un mathématicien aux prises avec son temps*, chap. VI, L'invention des distributions, chez Odile Jacob, 1997, et l'introduction de *La théorie des distributions*, chez Hermann, 1957.

3. Dans Genèse, 11, 7 : « Mais le seigneur descendit pour voir la ville et la tour [...] Allons, descendons pour mettre la confusion dans leur langage, en sorte qu'ils ne se comprennent plus l'un l'autre. »