Pair Programming

(BINÔMAGE)

Une pratique XP

Qu’est-ce que le pair programming ?

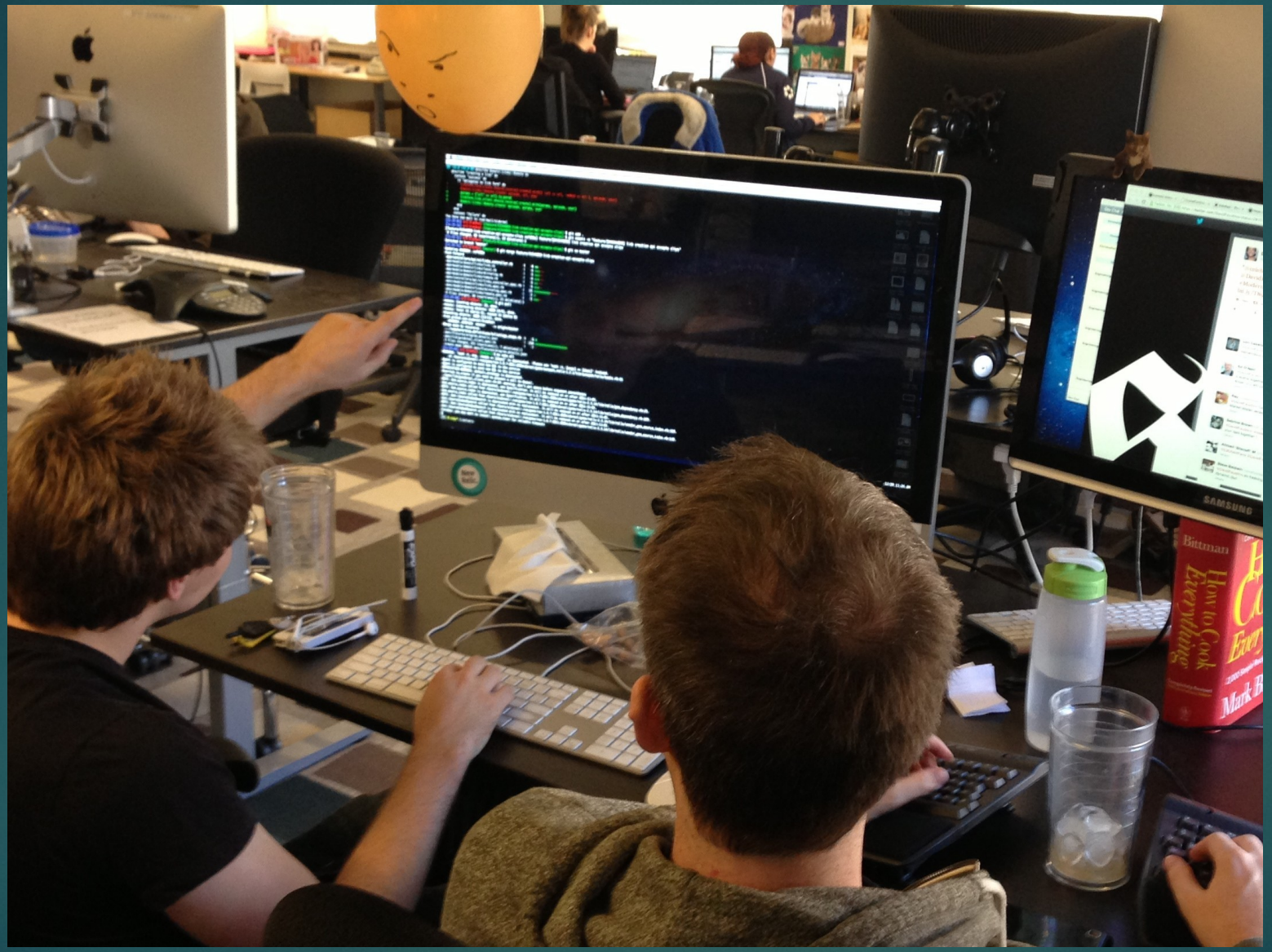
2 Personnes :

- Assises côte à côte !
- Sur la même tâche
- Avec le même objectif
- Pouvant avoir une expertise différente
- Travaillant en équipe

Qu’est-ce que le pair programming ?

2 rôles bien distincts :

- Le « pilote » : réalise la tâche
  - Il a le clavier
  - Il garde le focus sur l’objectif
- Le « co-pilote » :
  - Observe,
  - Garde la vue d’ensemble
  - Empêche la dispersion du pilote (note les points à revoir …)
  - Apporte un regard extérieur au code produit


SAMSUNG
Bittman
How to Cook Everything

Premières réactions ?

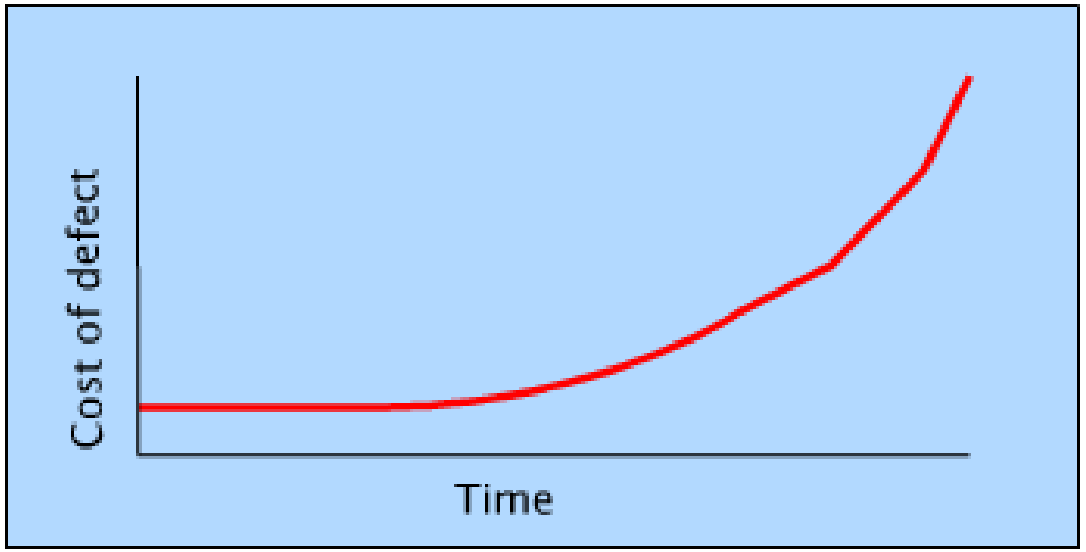
N’est-ce pas du gaspillage ?

- 2 devs qui font le travail d’un seul
- Les juniors vont ralentir les seniors
- « On va mettre 2 fois plus de temps »
- « Ca va coûter 2 fois plus cher »
- « Oui, mais bon, ça je sais faire tout seul »
- « Oui, mais on va pas pairer tout le temps ! »

Avantages ?

- Appropriation collective de la connaissance
- Moins d’anomalies / Résolution plus rapide
- Amélioration de la qualité (design, tests, …)
- - de distractions / + de productivité
- Motivation / satisfaction
- Montée en compétence
- Team building

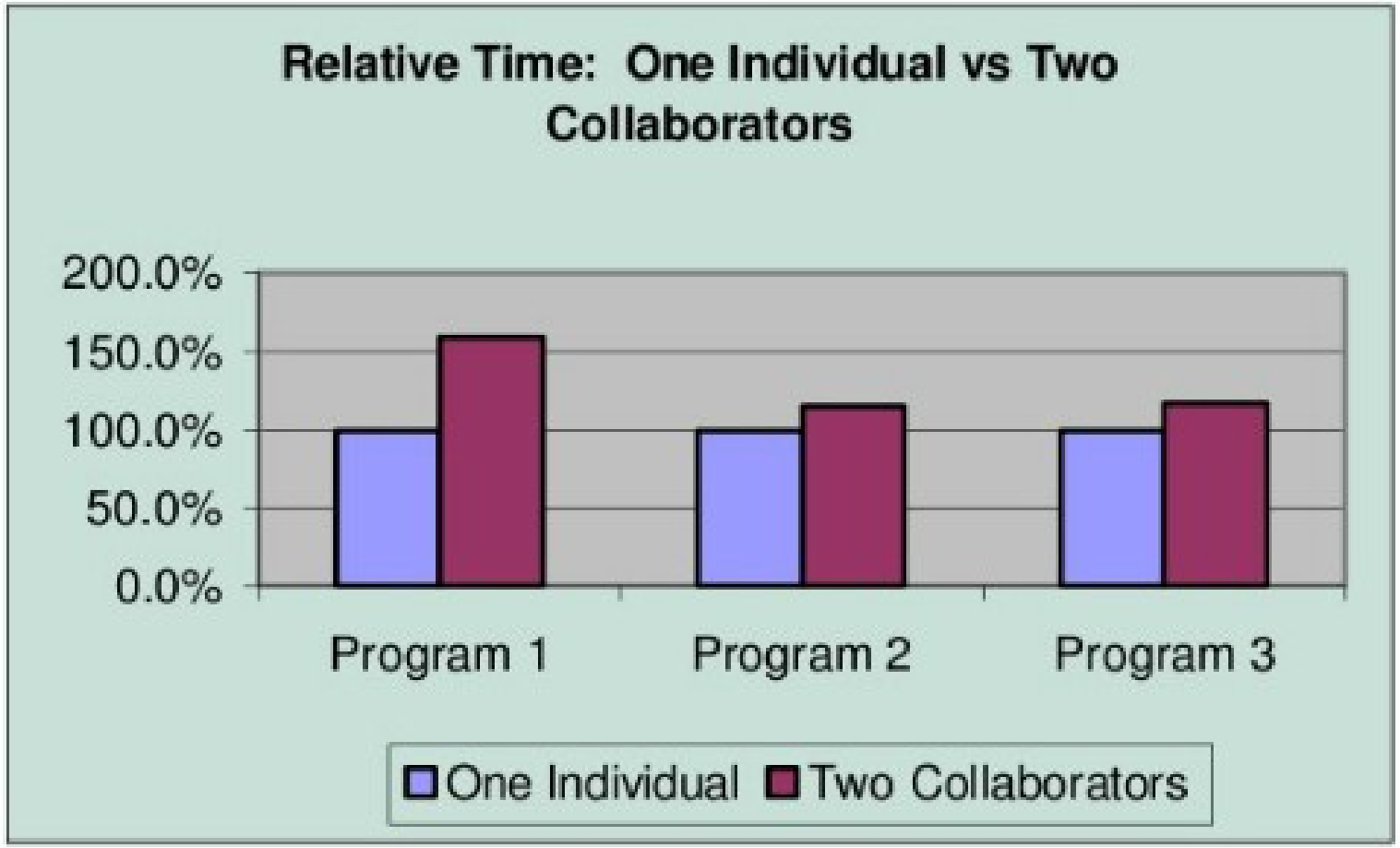
- Rentable … ?

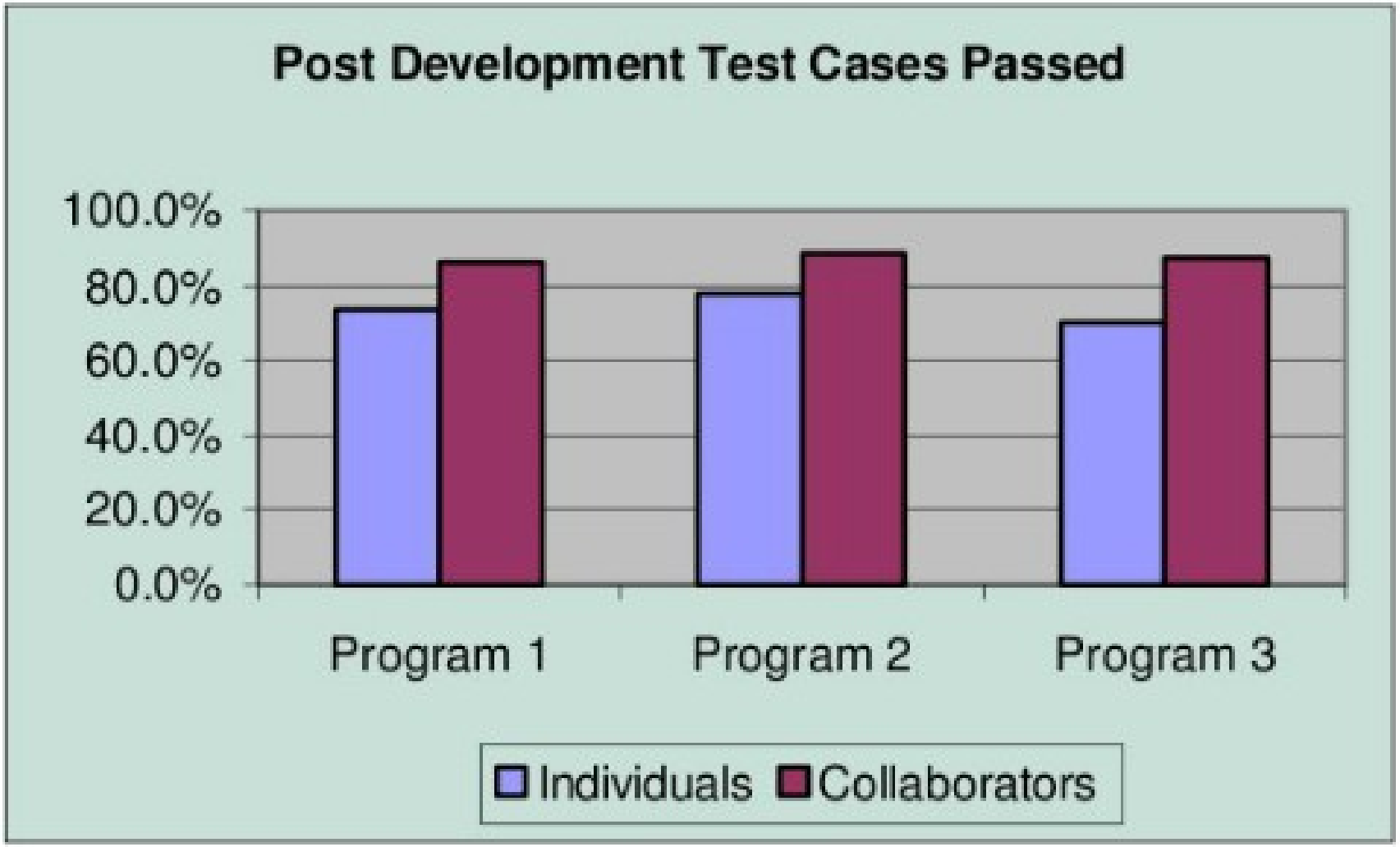


"IBM reported spending about $250 million repairing and reinstalling fixes to 30,000 customer-reported problems. That is over $8,000 for each defect!"

- *"A Discipline for Software Engineering", 1995,* Humphrey, W.S.

Pairing experiment at University of Utah




Post Development Test Cases Passed
100.0%
80.0%
60.0%
40.0%
20.0%
0.0%
Program 1
Program 2
Program 3
Individuals
Collaborators

Alors, on s’y met ?

- Développer demande un peu plus que de taper du code
  - Savoir-faire (craftmanship)
  - Compétence
  - Réflexion
  - Collaboration
  - Abstraction …

- Mettre en place le pair programming dans le process de développement aide :
  - à obtenir un produit de meilleur qualité
  - en moins de temps (moyen terme)
  - Avec des développeurs plus confiants
  - Et plus heureux !

Différentes façons de faire

- Tourner dans la journée :
  - Changer le rôle de pilote/Co-pilote à intervalle régulier
    - 1h30, 1/2journée …
  - Changer de binôme à intervalle régulier
    - 2 jours, 1jour, 1/2jour … ? (à essayer!)
    - Changer en fin d'une tâche : Ne fonctionne pas …
    - Permet de résumer l'histoire de la vieille
      - partage de connaissance
      - Simple à faire = simple à expliquer
    - Un œil externe sur le solution et le code réalisé

Points d’attention :

- Inutile de pairer tout le temps
  - Taches simples et répétitives (mais ca ne devrait presque pas exister !)
  - Besoin de réflexion posée
- Un binôme ne doit pas hésiter à se séparer 1h ou 2h puis revenir ensemble après par exemple …
- Fatigue : un rythme soutenu et soutenable
- Ecran au milieu, bureau adapté …
- Être patient, respectueux, attentif à sa communication

Réticences

- Difficulté à convaincre le management
- Peur d’être jugé, stress, insécurité personnelle
- Code « appartenant » à 1 développeur
- Définition de la qualité non partagée dans une équipe

Smells

- Pas de rotation des binômes
- Pas de rotation des rôles
- Distraction mutuelle
- Constitution des binômes en fonction des délais/criticités…
- Binômes décidés par un manager (ou le scrum master …)

- Le développeur qui n’assume pas son rôle de copilote
  - Ou qui dort :x
  - Ou qui joue tout le temps avec son smartphone …

Alors ?

- Pourquoi ne pas essayer avec timebox pendant 1 sprint ?