J’ai pas mal évoqué les problèmes de qualité de code et remis en cause la façon dont on construisait nos solutions.

Je suis plutôt de l’équipe qui croit qu’un code plus découplé est meilleur, car il limite les impacts, est plus facile à faire évoluer, à maintenir et à tester. Même si le couplage peut avoir de l’intérêt dans des approches de performance ou de simplicité.

Facile à tester, ça veut dire pouvoir mettre en place une politique de test, qui par ricochet va permettre d’aller vers l’intégration continue. Et lorsque l’on se dirige vers une solution SaaS, ça me semble être un bon point de départ.

Je suis aussi plutôt de l’équipe du découpage fin des fonctions, afin de ne pas leur faire porter plusieurs responsabilités. C’est le premier principe SOLID (*Single Responsibility Principle*), qui impacte directement l’application du second principe (*Open/Closed Principle*), et mène ainsi à une meilleure qualité de code.

J’aime bien prendre l’image des Lego. Lorsque l’on construit un château fort, il est plus facile de modifier une petite pièce ayant une responsabilité locale, plutôt que d’être obligé d’intervenir sur les fondations. Sinon, on risque d’abîmer la structure et d’impacter la stabilité de l’édifice à chaque réparation.

Tout ce que je raconte sur la qualité de code, le couplage, les principes de développement, la mise en place de tests automatisés, ce n’est pas une lubie. Tout cela a été documenté par des communautés de développeurs et classé dans la catégorie des bonnes pratiques de développement logiciel et de qualité de code. Bien qu’il s’agisse de recommandations, il peut être intéressant de les avoir en tête pour voir ce qu’elles peuvent apporter selon le contexte.

Tout ça pour en revenir à ce matin, en lisant mes mails, où je découvre que la synchronisation des factures n’a pas pu être réalisée, car…

… les contrats d’entretien n’ont pas été supprimés.

Et pour clarifier, à aucun moment les contrats d’entretien n’ont le moindre lien avec les factures. Jamais ils ne se rencontrent ni ne se croisent dans les solutions que nous mettons à disposition de nos clients…

… à l’exception de ce couplage technique fort, trop fort, qui cause des anomalies sur des objets totalement déconnectés au niveau métier.

La solution proposée par l’expert sollicité était de gérer une clé pour permettre la suppression en cascade des contrats d’entretien. Techniquement, ça tient la route, évidemment.

Mais n’est-il pas plus intéressant d’avoir une architecture propre et découplée, où chaque objet a sa place et ne dépend pas d’autres modules déconnectés des enjeux métier de la facturation ? Ou, pour le dire autrement, est-ce que la facturation ne devrait pas utiliser uniquement des objets propres à son fonctionnement, quitte à introduire un peu de duplication ou à mutualiser à un niveau plus stratégique que ce n’est le cas aujourd’hui ?

Dans notre système, un objet en erreur sans aucun impact direct peut entraîner un dysfonctionnement total de l’application. C’est un peu comme l’effet papillon : une petite anomalie à un endroit peut provoquer un véritable tsunami ailleurs. C’est précisément ce que le découplage cherche à éviter.