Make/Buy/Reuse (MBR)

Dans un projet, tu dois souvent décider **comment** réaliser une fonctionnalité : la fabriquer toi-même, l'acheter, ou la réutiliser. MBR est un modèle simple qui t'aide à faire ce choix de manière logique et défendable.

6TTR 5TTR Découverte

Pourquoi MBR est utile

MBR t'évite de prendre une décision "au feeling". Tu compares des options avec des critères concrets (temps, coût, risques…), puis tu choisis la solution la plus adaptée au projet.

Les trois options

Make (faire)

Tu développes la solution toi-même.

Tu choisis **Make** quand :

- tu as un besoin très spécifique (valeur ajoutée du projet),
- tu veux un contrôle maximal (fonctionnement, données, sécurité),
- tu as les compétences et le temps.

Attention : **Make** augmente souvent le risque de retard et de bugs, et demande plus de maintenance.

Buy (acheter)

Tu utilises un produit ou un service existant (licence, SaaS, API payante).

Tu choisis **Buy** quand :

- c'est un besoin standard (authentification, paiement, emailing…),
- tu veux gagner du temps,
- tu acceptes les contraintes d'un fournisseur (prix, limites, contrat, dépendance).

Reuse (réutiliser)

Tu réutilises un composant existant : code interne déjà validé, bibliothèque open-source, template, module du framework.

Tu choisis **Reuse** quand :

- c'est compatible avec ton besoin,
- c'est déjà testé et documenté,
- la licence et la sécurité sont acceptables.

Comment décider rapidement

Pour comparer Make / Buy / Reuse, utilise cette grille (simple mais efficace) :

- **Délai** : est-ce que ça te permet de livrer à temps ?
- **Coût total** : développement + maintenance + abonnements (pas seulement "gratuit/payant").
- **Risque technique** : est-ce facile à intégrer et à faire fonctionner ?
- **Maintenance** : qui corrige et met à jour dans 6 mois ?
- **Conformité** : licence, RGPD, données hébergées où et par qui ?

Règle pratique : si c'est **standard**, tu vas souvent vers **Buy** ou **Reuse**. Tu gardes **Make** pour ce qui rend ton projet unique.

Exemple concret (application web)

Besoin : "Les utilisateurs doivent se connecter et récupérer leur mot de passe."

- **Make** : tu codes tout (hash, tokens, reset, protections…).
- **Buy** : tu relies un service d'identité externe.
- **Reuse** : tu utilises un module d'authentification éprouvé (framework/bibliothèque).

Décision fréquente : **Reuse**, car tu avances vite tout en restant sur une solution robuste.

Synthèse

Ce qu'il faut retenir :

- **Make** = contrôle maximal, mais plus long et plus risqué.
- **Buy** = plus rapide, mais dépendance et contraintes.
- **Reuse** = souvent le meilleur compromis si c'est compatible.

- Tu décides avec des critères simples : **délai, coût total, risque, maintenance, conformité**.
- Une bonne décision MBR est **justifiée** et **écrite** (trace de décision).

Études de cas — Décisions MBR en situation réelle

Les études de cas suivantes montrent comment **Make / Buy / Reuse** est utilisé dans de vrais projets informatiques. Chaque cas met en évidence le **contexte**, le **choix effectué** et la **justification**, comme on l'attend dans une gestion de projet réaliste.

Cas réel — Application de billetterie pour un événement local

Contexte

Une petite équipe développe une application web pour gérer des événements : création d'événements, vente de tickets et contrôle à l'entrée. Le délai est court (quelques semaines) et le budget limité.

Besoin analysé

Paiement en ligne des billets.

Choix MBR

Buy

Justification

- Le paiement est un besoin **standard** et critique.
- Les exigences de sécurité (cartes bancaires, données sensibles) sont élevées.
- Développer un système fiable prendrait trop de temps et serait risqué.

Décision

Utilisation d'un service de paiement externe via API. L'équipe se concentre sur la logique métier (événements, billets, scan).

À retenir

Quand un besoin est sensible, normé et complexe, **Buy** réduit fortement les risques.

Cas réel — Site web d'une école avec espace élèves

Contexte

Une école souhaite un site web avec un espace sécurisé pour les élèves : consultation de documents, messages internes, accès par identifiant.

Besoin analysé

Authentification et gestion des comptes utilisateurs.

Choix MBR

Reuse

Justification

- Le framework utilisé propose déjà un module d'authentification éprouvé.
- La solution est documentée et maintenue.
- La licence est compatible avec le projet.

Décision

Réutilisation du module existant, avec une configuration adaptée aux besoins de l'école.

À retenir

Quand une solution fiable existe déjà et correspond au besoin, **Reuse** est souvent le meilleur compromis.

Cas réel — Logiciel interne de gestion de matériel informatique

Contexte

Un service informatique développe un outil interne pour suivre le matériel (PC, écrans, imprimantes) et les prêts aux utilisateurs.

Besoin analysé

Gestion spécifique des règles internes (types de matériel, procédures, rapports personnalisés).

Choix MBR

Make

Justification

- Les outils existants ne correspondent pas aux règles internes.
- Le logiciel n'est pas destiné à être vendu ou généralisé.
- L'équipe possède les compétences nécessaires.

Décision

Développement sur mesure, parfaitement aligné avec les procédures internes.

À retenir

Quand le besoin est très spécifique et stratégique, **Make** permet un contrôle total.

Synthèse transversale

Ces trois cas montrent que :

- **Buy** est pertinent pour les fonctionnalités standards et critiques.
- **Reuse** est idéal quand une solution existante est compatible et fiable.
- **Make** est réservé aux besoins spécifiques qui font la valeur du projet.

En gestion de projet, il n'y a pas de "bon choix universel". Une bonne décision MBR est toujours **contextuelle**, **argumentée** et **assumée**.