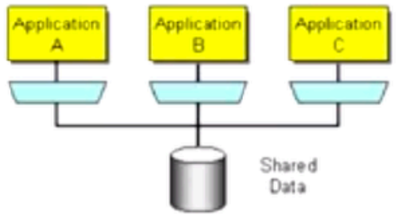
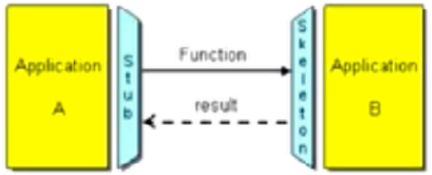
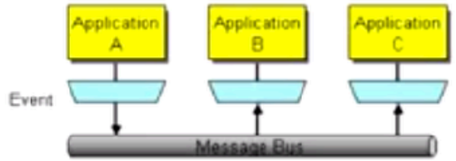


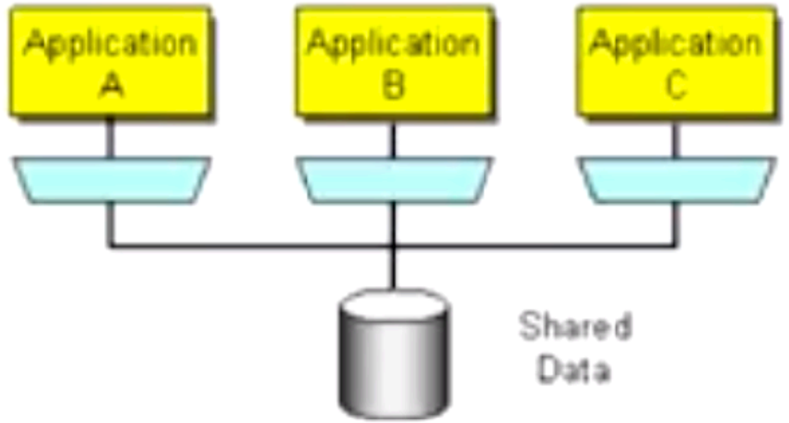


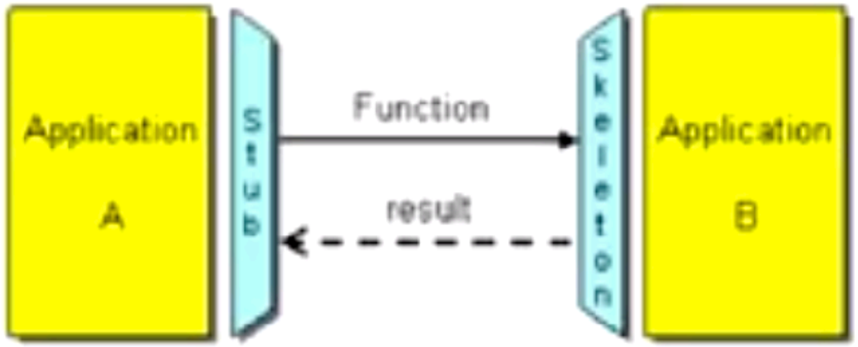


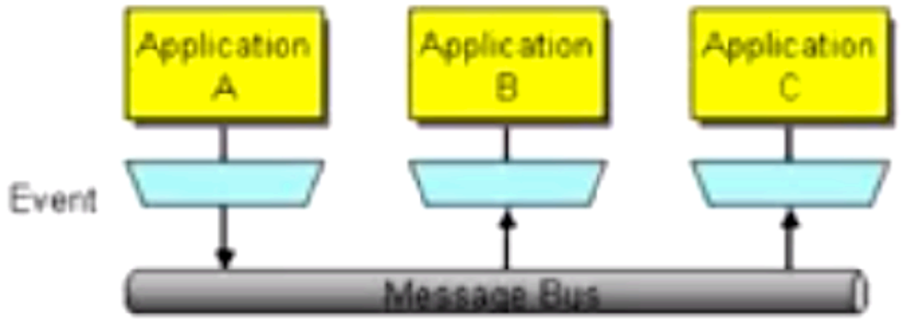
STYLES D'INTÉGRATION

ARCHITECTURE LOGICIELLE

ENTREPRISE INTEGRATION PATTERNS





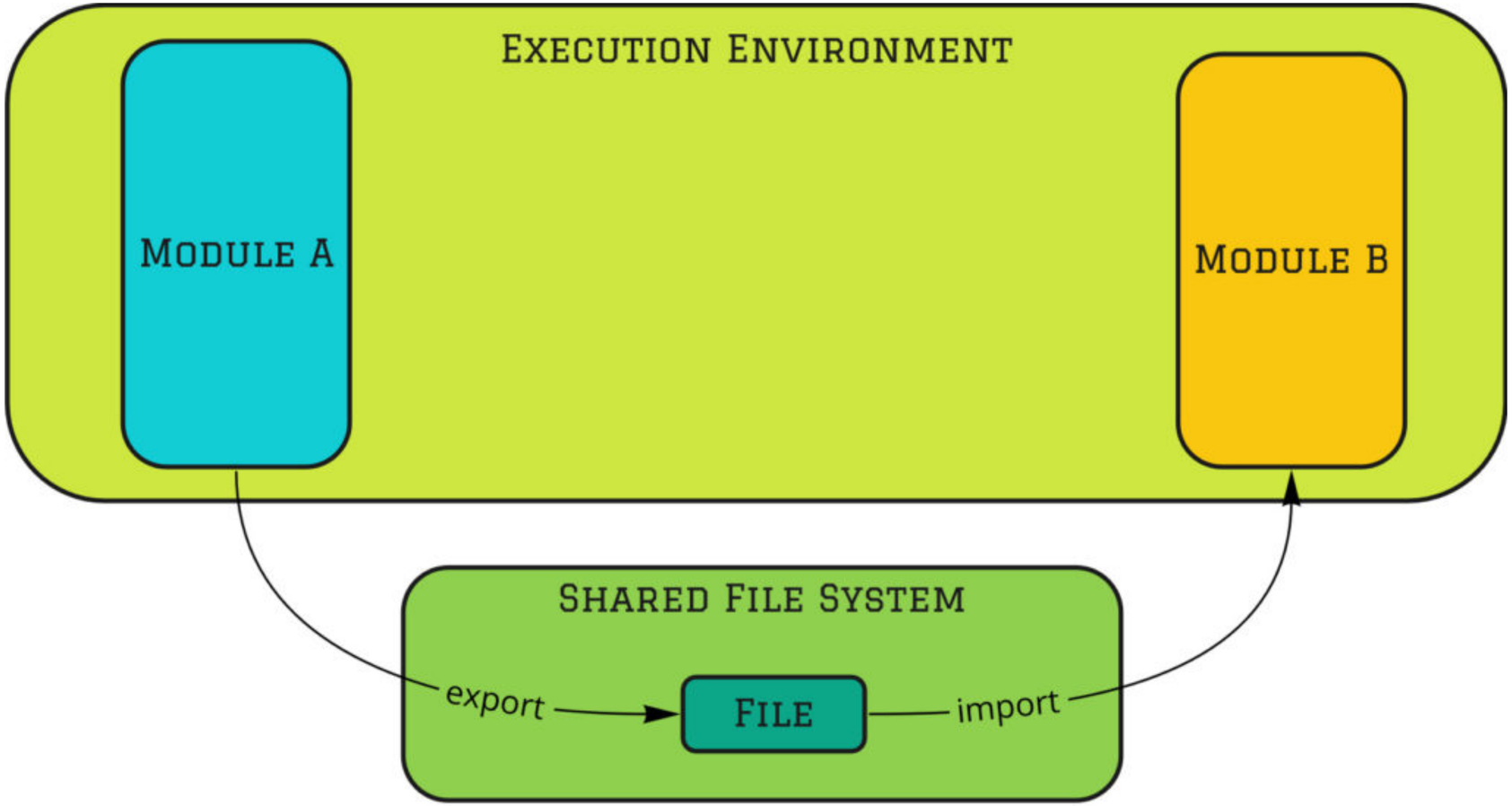


Gregor Hohpe

CRITÈRES D'INTÉGRATION

- Coupling
  - Mesure le degré de dépendances entre deux modules
- Complexity
  - De la simplicité à la complexité d'un module qui demande de l'engagement, de la connaissance et de la discipline
- Data Timeless
  - Le temps entre la fourniture d'un donnée d'un module et l'exploitation de cette même donnée par un autre module

FILE TRANSFER



https://www.kamilgrzybek.com/design/modular-monolith-integration-styles/

FILE MANAGER

- Usage : principalement du batch
- Fonctionnement
    - Asynchrone: pas d’attente de traitement des données
- Avantages
    - Simple à mettre en oeuvre
    - Couplage inexistant entre le module A et le module B :
        - grande évolutivité : chacun peut avoir son propre cycle de vie
        - Pas de dépendances entre les modules au runtime
            - Ex: le module B peut être inactif lors de la production de données du module A

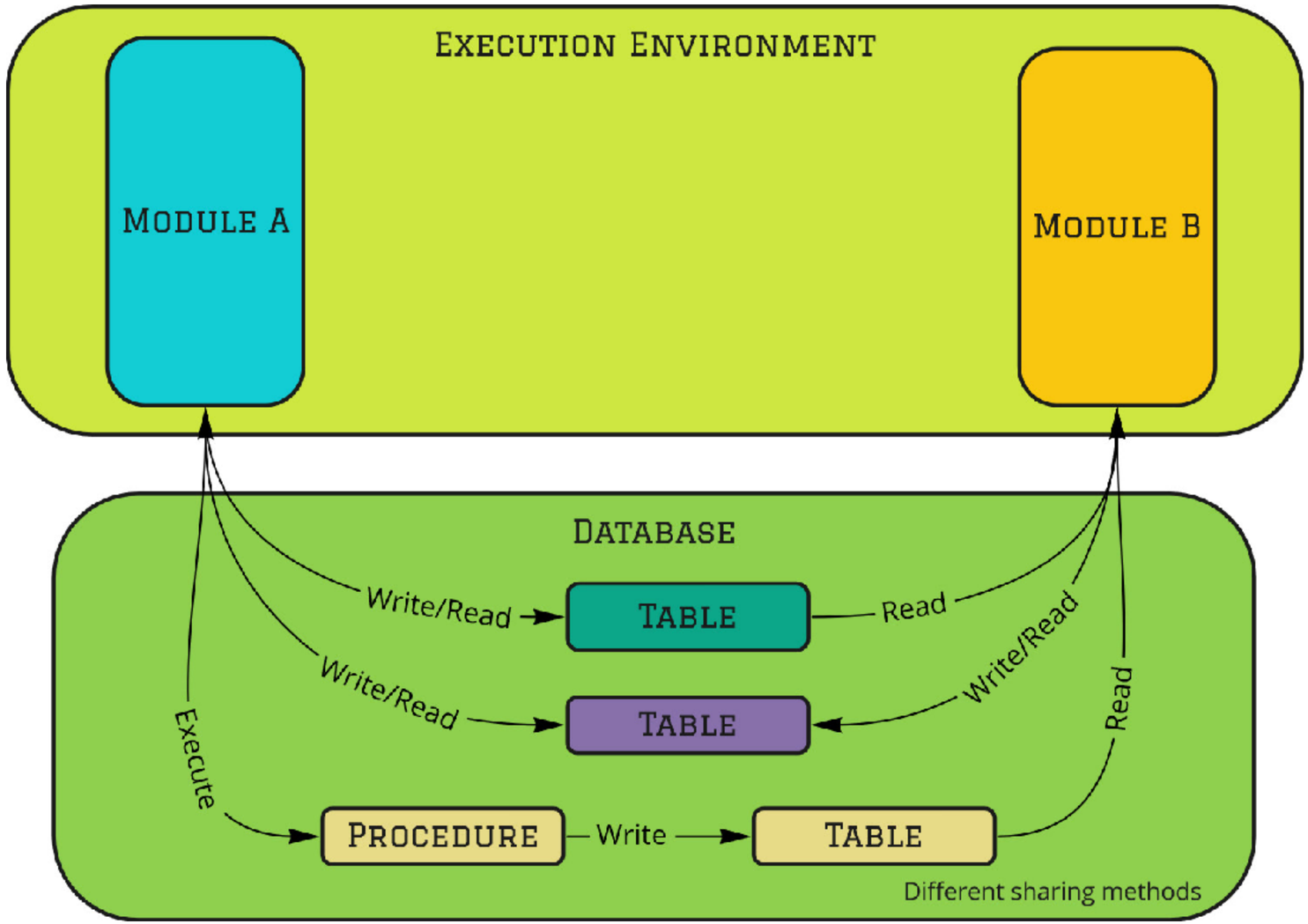
FILE MANAGER

- Inconvénients
  - Latence des données
  - Nécessite de mise en place d'une monenclature du noms des fichiers
  - Ressource partagée (lecture/écriture)
    - Exemple de solution: l'usage de fichier témoin (.TOP) pour déclencher le traitement de fichiers de collecte de données
  - Contrat d'interface est portée au niveau de la structuration de la donnée et partagée entre les deux modules
  - Sécurité des données
    - Exemples de solutions : chiffrement des fichiers avec des clés, mise en place d'une solution tiers pour accéder aux données (ObjectRepository), signature des fichiers pour garantir l'intégrité
  - Gestion de la purge
  - Gestion des I/O
  - Création de fichiers dédiés à l'acquittement (.AR)

FILE MANAGER

- Exemple de fonctionnement avec le fichier .TOP
  - Le module A écrit en continue un fichier de donnée
    - 333.DATA
  - Quand le module A a finit d'écrire ses données, il crée un fichier .TOP nommé du même nom que le fichier de données
    - Ex: 333.TOP
  - Le module B scrute la présence de fichier .TOP et récupère le fichier de données associé (.DATA) - qui est par construction terminé
  - Prérequis : Une fois que le fichier .TOP a été crée, il n'y a plus d'écritures possibles dans le fichier de données (.DATA)

SHARED DATA



https://www.kamilgrzybek.com/design/modular-monolith-integration-styles/

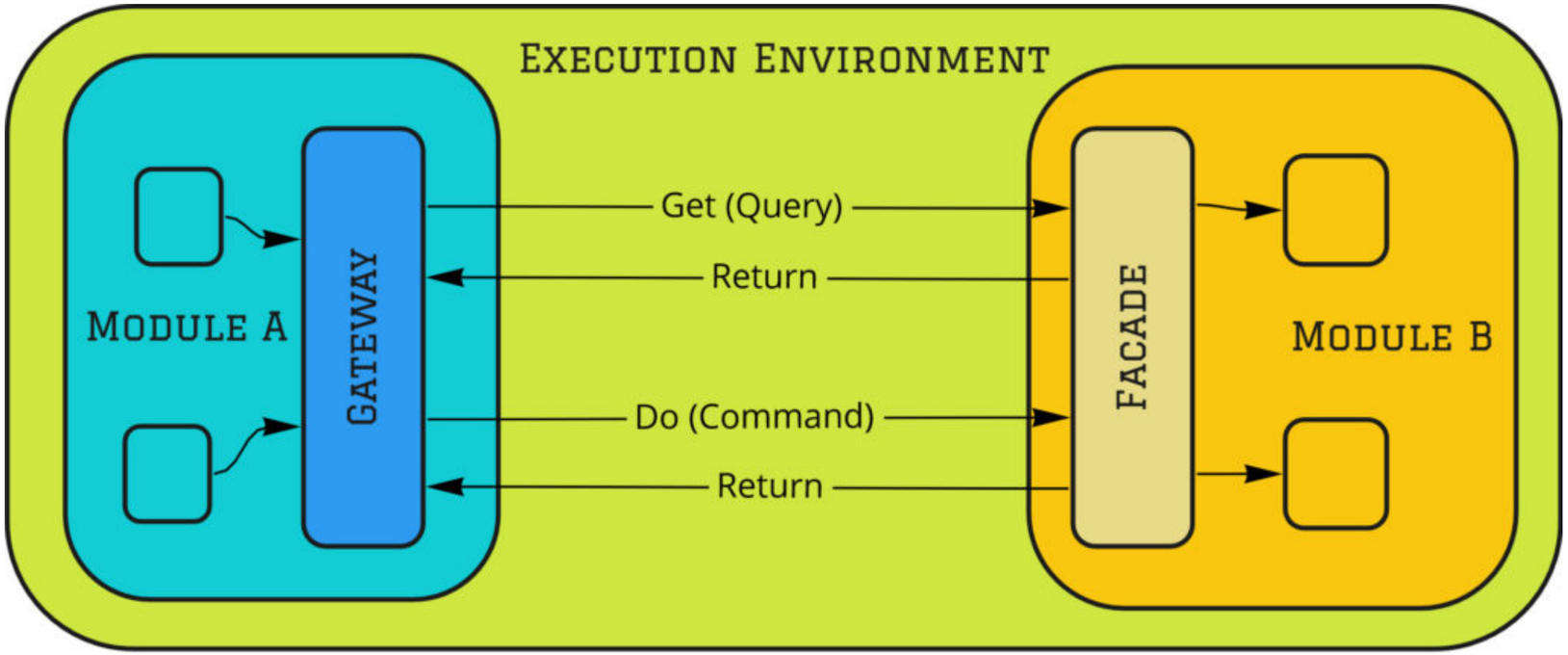
SHARED DATABASE

- Avantages
  - Une source de vérité transactionnelle (ACID)
    - Moins de transfert de données
    - Une donnée accessible et à jour automatiquement par l'ensemble des applications consommatrices
  - Possibilité de la mise en place de trigger (fonction/bout de code au niveau du moteur de base de données qui est déclenché automatiquement par la mise à jour d'une table de données) pour information l'ensemble des applications en cas de mise à jour
  - Utilisation de la sécurité associée à la solution de base de données (authentification & autorisation)

SHARED DATABASE

- Inconvénients
  - Besoin d'avoir l'ensemble des composants applicatifs (module A & module B) et les composants techniques (service de base de données) disponibles
  - Le schéma de la base de données est partagée à travers toutes les applications
    - Ajout/Suppression/Modification dépend de chaque application
    - Cycle de vie dépendant du schéma
  - Nécessité d'avoir la base de données en haute-disponibilité
  - Utilisation d'un même schéma avec les bases de données relationnelles (RDBMS)
    - Limité avec les bases de données NoSQL (schemaless & query driven design)

REMOTING / DIRECT CALL



https://www.kamilgrzybek.com/design/modular-monolith-integration-styles/

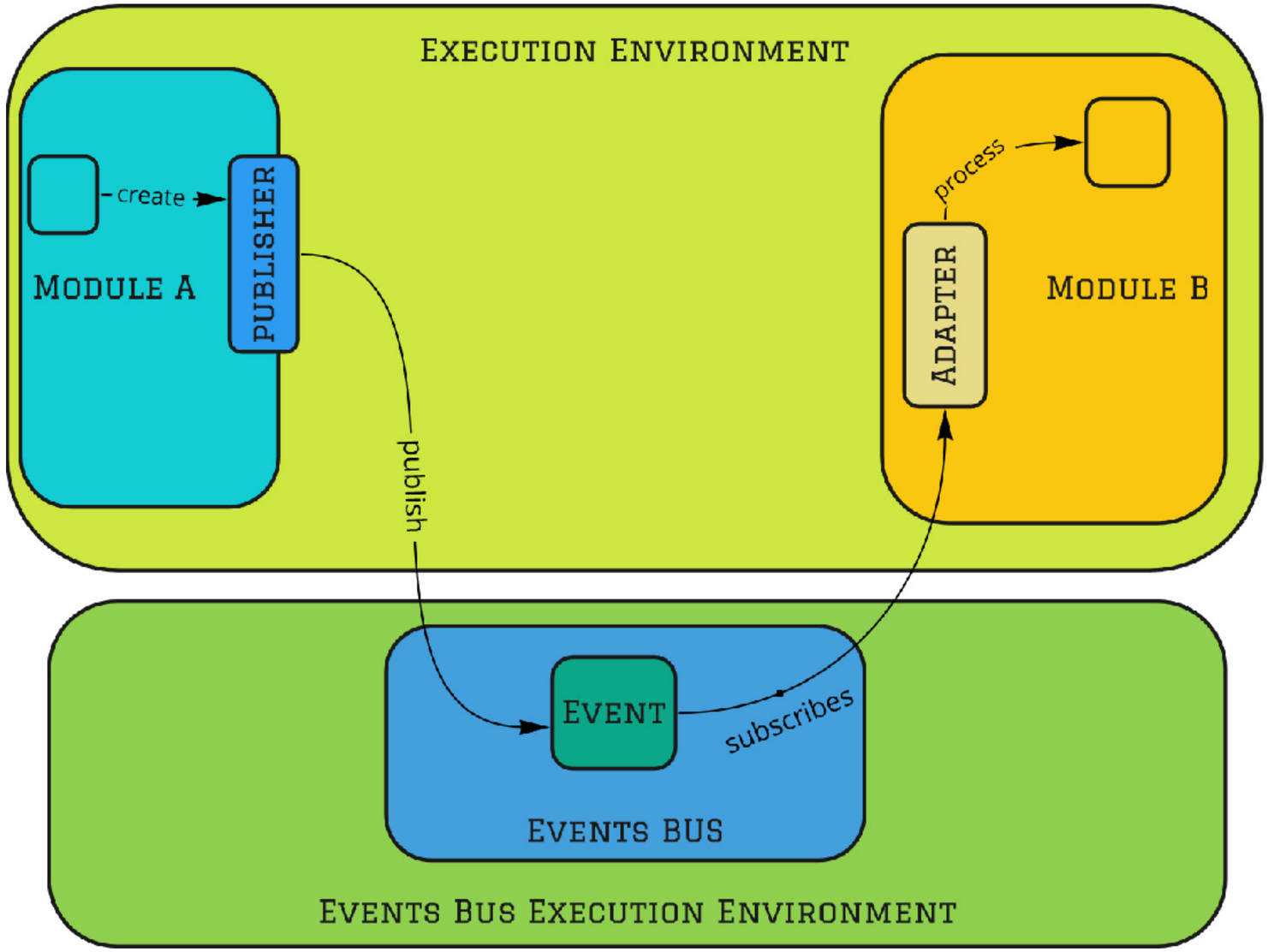
REMOTING

- Exemples d'utilisation
  - Web services SOAP
- Avantages
  - Simple à mettre en place
    - Une transposition 1-1 entre la couche de services et la couche d'exposition
    - Très présent dans l'écosystème Java : Java Remoting
      - RMI//IIOP (EJB), Spring Remoting, etc

REMOTING

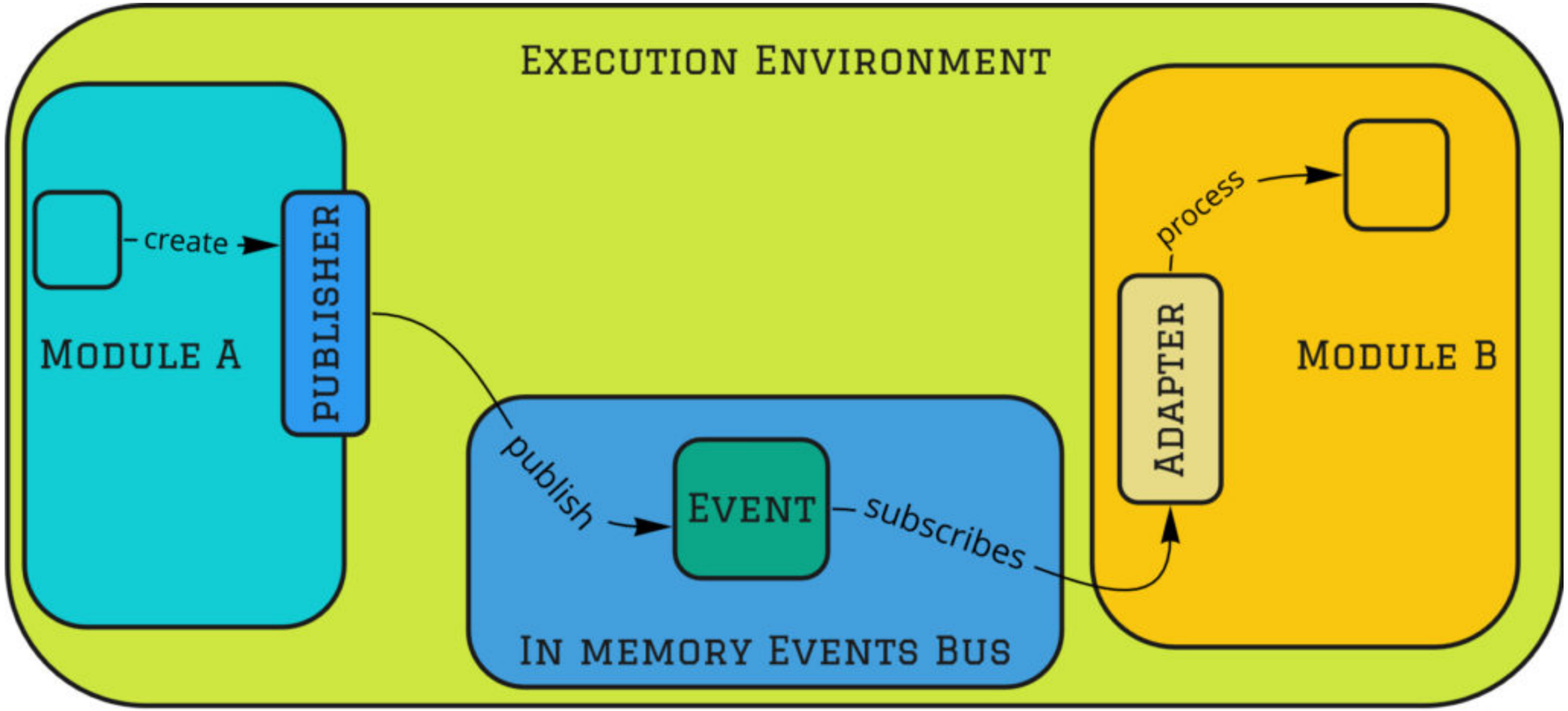
- Inconvénients
  - Les deux modules doivent être actif lors de la communication
  - Les deux modules sont fortement couplés
    - Spatial
      - Doivent actif tous les deux
    - Temporel
      - Synchrone
    - Applicatif
      - Le module A doit connaître le nom des services(Méthodes), le nom et type de paramètres, le type de résultat
  - Gestion des versions

MESSAGING (1/2)



https://www.kamilgrzybek.com/design/modular-monolith-integration-styles/

MESSAGING (2/2)



https://www.kamilgrzybek.com/design/modular-monolith-integration-styles/

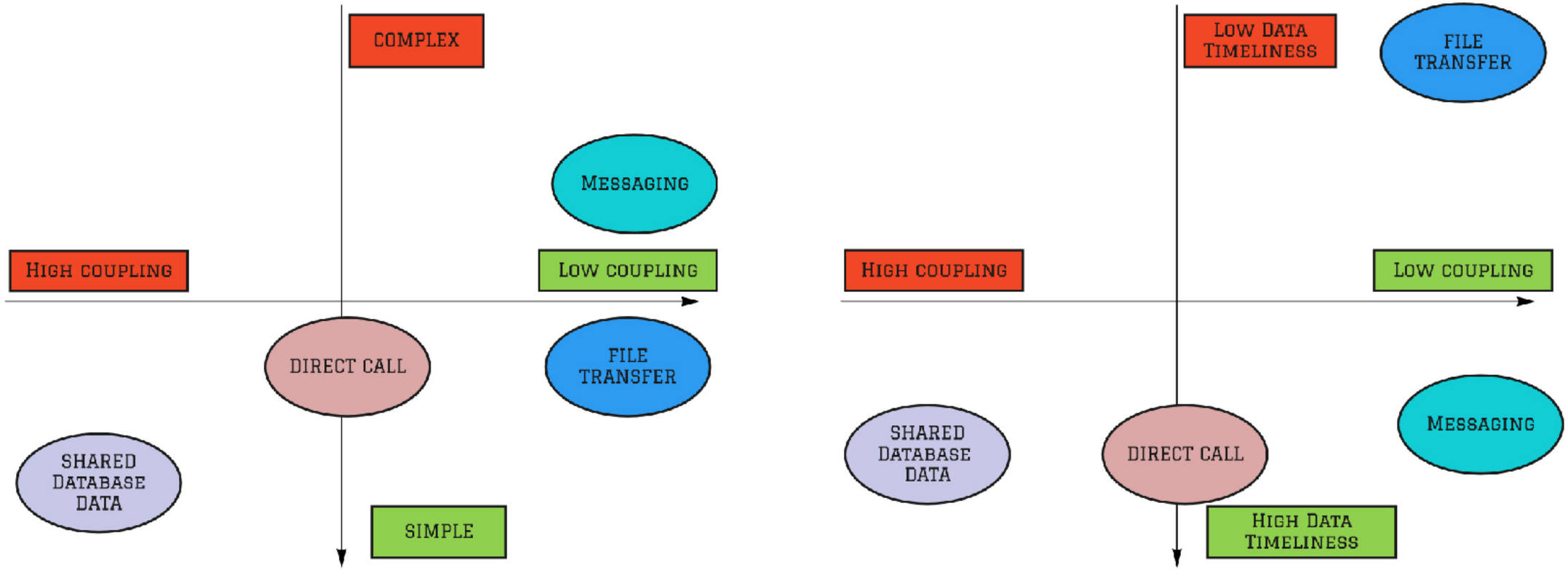
ASYNC MESSAGING

- Fonctionnement
  - Le message envoyé contient toutes les informations pour la mise à jour des applications (comme le File Transfer)
  - Au fil de l'eau, les messages sont envoyés
  - Technologies: bus de messages comme ActiveMQ/Arthemis, RabbitMQ, Apache Kafka
- Avantages
  - Les applications sont découplées entre elles
  - Elles sont pas besoin d'être actives en même temps
    - Bufferisation des messages dans le bus
  - Réduction de la latence de transfer des données

ASYNC MESSAGING

- Inconvénients
  - Complexité de mise en oeuvre
    - Asynchronisme, choregraphie, gestion du contrat de chaque message à définir

COMPARAISON



https://www.kamilgrzybek.com/design/modular-monolith-integration-styles/