

# Le processus Unifié

Le **Processus Unifié** est apparu en **1996**.

C'est un **guide méthodologique** qui regroupe les **bonnes pratiques** pour le développement de logiciels.

Le mot « *unifié* » vient de la **fusion des travaux de Jacobson, Booch et Rumbaugh**, qui ont aussi participé à la création du langage UML.

## Les bonnes pratiques de l'UP

1. **Développement itératif** : le projet est découpé en petits cycles pour mieux suivre l'avancement.
2. **Gestion des exigences** : les besoins du client sont bien suivis et documentés.
3. **Architecture par composants** : le logiciel est organisé en parties réutilisables et modulaires.
4. **Modélisation visuelle (UML)** : on utilise des diagrammes pour représenter le système.
5. **Contrôle de la qualité** : le produit est vérifié à chaque étape.
6. **Gestion des changements** : on gère facilement les modifications du projet.

## Dans quels projets utiliser UP ?

- Des projets **de taille moyenne à grande** (entre 6 mois et 2 ans),
- Des **architectures complexes** avec plusieurs intégrations,
- Des **équipes de 10 à 50 développeurs**,
- Des projets où les **besoins évoluent** mais restent structurés,
- Des projets avec des **risques techniques ou fonctionnels modérés à élevés**.

## Les quatre grands principes du Processus Unifié

### 1. Piloté par les cas d'utilisation

Les **cas d'utilisation** (use cases) représentent les besoins fonctionnels vus du point de vue de l'utilisateur.

Le processus de développement est donc **centré sur l'utilisateur**.

À partir de ces cas, on crée les **modèles UML** qui décrivent les fonctions du système.

### 2. Centré sur l'architecture

L'**architecture** du logiciel regroupe les différentes **vues du système** : fonctionnelle, logique, matérielle, etc.

On commence par une version simple, puis on l'enrichit progressivement à mesure que de nouveaux cas d'utilisation sont ajoutés.

Ainsi, l'architecture évolue en même temps que le projet.

### 3. Itératif et incrémental

Le projet est découpé en **itérations courtes** (de 2 semaines à 2 mois).

Chaque itération apporte une **nouvelle version** du logiciel, plus complète que la précédente.

Cela permet de suivre facilement la progression et de **corriger les problèmes au fur et à mesure**.

C'est ce qu'on appelle le développement **incrémental**.

### 4. Piloté par la diminution des risques

Les **risques** (techniques, financiers, humains, etc.) sont identifiés dès le début.  
Les itérations sont planifiées de façon à traiter **les risques les plus importants en premier**.  
Ainsi, plus le projet avance, plus les risques diminuent.

### Les phases du Processus Unifié

Le cycle de vie du Processus Unifié comporte **quatre grandes phases**. Chaque phase contient plusieurs **itérations** et **activités**.

#### 1. Pré-étude (Inception)

Objectif : définir le **périmètre du projet** et vérifier sa faisabilité.

Tâches principales :

- Définir les **besoins fonctionnels et non fonctionnels**.
- Identifier les **acteurs** et les **cas d'utilisation principaux**.
- Repérer les **risques importants**.
- Évaluer si le projet est **réalisable**.

Résultat : un document de **vision** du projet.

#### 2. Élaboration

Objectif : concevoir l'**architecture générale** du système et planifier le projet.

Tâches principales :

- Définir l'**architecture logicielle**.
- Identifier les **composants essentiels**.
- Planifier les ressources, le **budget** et les **délais**.
- Confirmer la **faisabilité technique**.

Résultat : le **document d'architecture logicielle**.

#### 3. Construction

Objectif : réaliser le **développement complet** du logiciel.

Tâches principales :

- Finaliser l'**analyse**, la **conception** et l'**implémentation**.
- Effectuer les **tests unitaires et d'intégration**.
- Réduire les **risques critiques** restants.

Résultat : une **version stable et fonctionnelle** du produit.

#### 4. Transition

Objectif : **livrer le produit** au client et le préparer à son utilisation.

Tâches principales :

- Adapter le logiciel à l'environnement du client.
- Corriger les anomalies restantes.
- Rédiger les **manuels** et **former les utilisateurs**.

Résultat : **logiciel final déployé** chez le client.

## Les activités principales du Processus Unifié

1. **Modélisation métier** → comprendre l'organisation et son fonctionnement.
2. **Exigences** → exprimer les besoins et les cas d'utilisation.
3. **Analyse et conception** → créer les modèles UML.
4. **Implémentation** → coder le système selon les modèles.
5. **Test** → vérifier et corriger les erreurs.
6. **Déploiement** → installer et gérer les versions du logiciel.

## Les extensions du Processus Unifié

- **RUP (Rational Unified Process)**
- **EUP (Enterprise Unified Process)**
- **2TUP (2tracks Unified Process)**
- **BUP (Basic Unified Process)**
- **OUM (Oracle Unified Method)**

### ◆ 1. RUP (Rational Unified Process)

Développé en 1998 par **IBM**. C'est une version **commerciale et complète** du Processus Unifié.

Il fournit :

- Des **modèles de documents** prêts à l'emploi.
- Des **guides détaillés** pour chaque activité.
- Des **outils intégrés** (Rational Suite).
- Un **cadre de certification** pour les professionnels.

### Quand utiliser RUP ?

- Grands projets d'entreprise (banques, santé, énergie...).
- Projets très réglementés.
- Équipes réparties sur plusieurs sites.

### ◆ 2. EUP (Enterprise Unified Process)

Le **EUP** est une **extension stratégique** du RUP.

Il ajoute une vision plus large du **cycle de vie du système** et de la **gestion d'entreprise**.

Ses **4 piliers** :

1. Gestion du portefeuille de projets.
2. Gestion des capacités et ressources.
3. Gouvernance informatique.
4. Gestion du changement organisationnel.

EUP ajoute **deux nouvelles phases** :

- **Production** : exploitation et maintenance du système.
- **Retrait** : fin de vie et remplacement du logiciel.

### Quand utiliser EUP ?

- Grands programmes de transformation digitale.
- Systèmes complexes et de longue durée (plus de 10 ans).
- Projets multi-équipes et interdépendants.

### **Conclusion**

Le **Processus Unifié** représente une évolution importante dans la façon de concevoir les logiciels. Son choix dépend de la **taille du projet**, du **niveau de risque** et de la **maturité de l'équipe** :

- **UP** pour les projets moyens,
- **RUP** pour les grands projets,
- **EUP** pour les transformations d'entreprise.