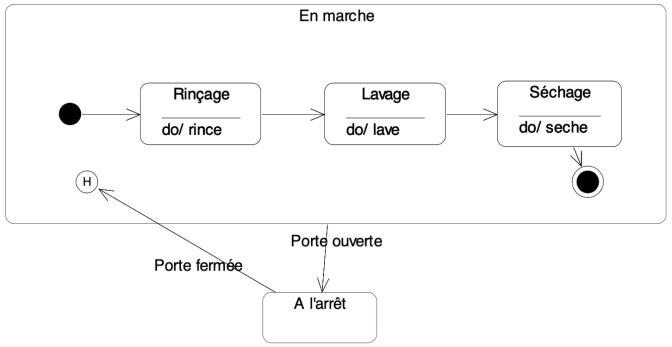
Processus de Développement Logiciel - PDL

Leonard Cseres | June 12, 2025

Diagramme d'états

- **Éléments:**
  - **États:** Conditions d'un objet.
  - **États initiaux et finaux**
  - **Transitions:** Déplacement entre états, déclenchées par événements.
  - **Actions:** Activités lors de l'entrée, sortie ou pendant un état.
  - **États imbriqués et états d'historique**
- H : pour revenir à l'état d'où on est sortie
- H* : historique profond, revient à l'état de sortie des états imbriqués



Méthodes Agiles

Manifeste Agile - Valeurs

- **Individus et interactions** > processus et outils
- **Logiciels opérationnels** > documentation exhaustive
- **Collaboration avec clients** > négociation contractuelle
- **Adaptation au changement** > suivi d'un plan

Manifeste Agile - 12 Principes

1. **Satisfaire client:** Livraison rapide et régulière de fonctionnalités à valeur ajoutée
2. **Changements positifs:** Accueillir les changements, même tard dans le projet
3. **Livraison fréquente:** Cycles de quelques semaines à quelques mois (préférence courts)
4. **Collaboration quotidienne:** Utilisateurs/représentants et développeurs ensemble
5. **Personnes motivées:** Environnement, soutien et confiance
6. **Communication face-à-face:** Méthode la plus efficace de transmission d'information
7. **Logiciel opérationnel:** Principale mesure d'avancement
8. **Rythme soutenable:** Maintenir indéfiniment un rythme constant
9. **Excellence technique:** Attention continue à la conception et technique
10. **Simplicité:** Minimiser le travail inutile
11. **Équipes auto-organisées:** Meilleures architectures et conceptions émergent
12. **Amélioration continue:** Réflexion régulière pour devenir plus efficace

XP - Valeurs

- **Simplicité:** Faire le nécessaire, pas plus
- **Communication:** Face-à-face quotidien, travail ensemble
- **Feedback:** Logiciel fonctionnel, démonstrations précoces
- **Courage:** Dire la vérité, s'adapter aux changements
- **Respect:** Valeur de chaque membre, expertise mutuelle

XP - Principes (Sélection)

- **Humanité:** Besoins développeurs (sécurité, accomplissement, appartenance)
- **Économie:** Priorisation selon valeur créée, coût temporel
- **Bénéfice mutuel:** Pratiques profitables aujourd'hui ET demain
- **Amélioration:** "Perfect" est un verbe, pas un adjectif
- **Qualité:** Accélère le développement, ne le ralentit pas
- **Petits pas:** Plus petit changement possible dans bonne direction

XP - Pratiques Clés

- **Client sur site:** Représentant utilisateur disponible
- **Planification incrémentale:** User stories, estimations, itérations courtes
- **Programmation en binôme:** 2 développeurs, 1 ordinateur
- **Développement orienté tests:** Tests avant fonctionnalité
- **Intégration continue:** Intégration plusieurs fois/jour
- **Refactoring:** Amélioration continue du code
- **Propriété collective:** Tous peuvent modifier tout

SCRUM - Rôles

- **Product Owner:** Gestion backlog, priorités business
- **Scrum Master:** Coach équipe, facilitateur
- **Développeurs:** Auto-organisation, livraison

SCRUM - Événements

- **Sprint Planning:** Objectif + sélection backlog items
- **Daily Scrum:** 15min, progrès + obstacles
- **Sprint Review:** Présentation aux parties prenantes
- **Sprint Retrospective:** Amélioration continue

SCRUM - Artefacts

- **Product Backlog:** Liste ordonnée des besoins
- **Sprint Backlog:** Objectif + items + plan d'action
- **Incrément:** Version utilisable du produit

Test-Driven Development (TDD)

Cycle TDD

1. **Red:** Écrire test qui échoue
2. **Green:** Implémenter fonctionnalité minimale
3. **Refactor:** Nettoyer le code

Pyramide des Tests

```
    Tests E2E
  Tests Intégration
Tests Unitaires (base)
```

Pattern AAA

- **Arrange:** Préparer système et dépendances
- **Act:** Exécuter méthodes à tester
- **Assert:** Vérifier résultats

Refactoring

Code Smells

- **Bloaters:** Long Method, Large Class, Long Parameter List
- **OO Abusers:** Mauvaise application POO
- **Change Preventers:** Modifications en cascade
- **Dispensables:** Code mort, commentaires excessifs

Techniques Refactoring

- **Extract Method:** Diviser méthodes longues
- **Rename Variable:** Noms explicites
- **Replace Magic Number:** Constantes symboliques
- **Introduce Parameter Object:** Remplacer listes paramètres

Personas

- **Personnages fictifs** représentant utilisateurs/intervenants
- **Partagés et visibles** dans l'équipe
- **Utilisés dans user stories** pour contextualiser besoins

User Stories

Format Standard

```
En tant que <type utilisateur>,
Je veux <objectif>
Pour que <raison/bénéfice>
```

Hiérarchie

- **Epic:** Histoire générale de haut niveau
  - Exemple: "En tant qu'utilisateur, je peux backuper mon disque dur pour ne pas perdre de données"
- **User Stories:** Décomposition d'épiques en éléments plus petits
  - Exemple: "En tant que power user, je peux spécifier les fichiers selon taille/date"
- **Tâches:** Division technique par développeurs (2-3 jours typique)

Caractéristiques INVEST

- **I**ndépendantes
- **N**égociables
- **V**alorisées (valeur business)
- **E**stimables
- **S**mall (petites, réalisables en sprint)
- **T**estables

Avantages

- Plus faciles à comprendre qu'une spécification
- Repriorisables/modifiables selon besoins changeants
- Développement au fur et à mesure

Limitations

- Difficile d'évaluer couverture complète des besoins
- Besoins non-fonctionnels peu définis (sécurité, performance)
- Spécification incomplète sur papier (complétée par discussions)

Outils Gestion Projet

Kanban

- **Colonnes:** Backlog → Doing → Review → Done
- **Objectif:** Flux continu, limitation WIP

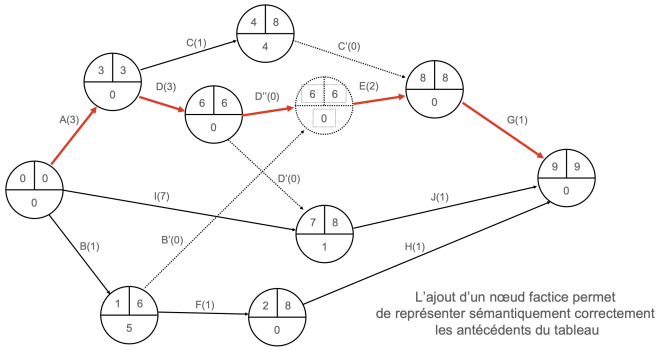
Burndown Chart

- **X:** Temps disponible
- **Y:** Travail restant
- **Ligne idéale:** Progression théorique

PERT/CPM

CPM - Étapes

1. **Liste tâches:** Antécédents + durées
2. **Dates au plus tôt:** Calcul de gauche à droite
3. **Dates au plus tard:** Calcul de droite à gauche
4. **Marges:** Date plus tard - Date plus tôt - Durée
5. **Chemin critique:** Tâches avec marge = 0



CPM - Création Chemin Critique (Étapes Détaillées)

1. **Identification des tâches:**
   - Lister toutes les activités du projet
   - Définir les dépendances entre tâches
   - Estimer la durée de chaque tâche
2. **Construction du réseau:**
   - Créer un diagramme de réseau
   - Relier les tâches selon leurs dépendances
   - Identifier les nœuds de début et fin
3. **Calcul des dates au plus tôt (ES - Earliest Start):**
   - Commencer par la tâche initiale (ES = 0)
   - Pour chaque tâche: ES = max(EF des prédécesseurs)
   - EF (Earliest Finish) = ES + Durée
4. **Calcul des dates au plus tard (LS - Latest Start):**
   - Commencer par la tâche finale (LF = EF)
   - Remonter: LS = LF - Durée
   - Pour chaque tâche: LF = min(LS des successeurs)
5. **Calcul des marges:**
   - Marge totale = LS - ES = LF - EF
   - Marge libre = ES(successeur) - EF(tâche)
6. **Identification du chemin critique:**
   - Sélectionner toutes les tâches avec marge = 0
   - Tracer le chemin continu de début à fin
   - Ce chemin détermine la durée minimale du projet

PERT - Estimations 3 Points

- **o:** Optimiste, **a:** Probable, **p:** Pessimiste

Formules par Tâche:

- $E(tâche) = \frac{o+4a+p}{6}$
- $SD(tâche) = \frac{p-o}{6}$
- $Var(tâche) = SD^2 = \left(\frac{p-o}{6}\right)^2$

Formules Projet:

- $E(projet) = \sum E(tâches)$
- $Var(projet) = \sum SD^2(tâches)$
- $SD(projet) = \sqrt{Var(projet)}$

Intervalles de Confiance:

- $IC = E(projet) \pm z \cdot SD(projet)$
- z = 1 → 68.26%, z = 2 → 95.44%, z = 3 → 99.72%

Estimation Coûts:

- $Coût = E(projet) \times Taux\_horaire$
- $Coût_{max} = (Borne\_sup\_IC) \times Taux\_horaire$

PERT - Étapes de Calcul Détaillées

1. **Collecte des estimations 3 points:**
   - Pour chaque tâche, obtenir:
     - **o** (optimiste): durée minimum possible
     - **a** (probable): durée la plus réaliste
     - **p** (pessimiste): durée maximum possible
2. **Calcul temps espéré et écart-type par tâche:**
   - Temps espéré: $TE = \frac{o+4a+p}{6}$
   - Écart-type: $\sigma = \frac{p-o}{6}$
   - Variance: $V = \sigma^2 = \left(\frac{p-o}{6}\right)^2$
3. **Construction réseau et identification chemin critique:**
   - Utiliser les temps espérés (TE) comme durées
   - Appliquer méthode CPM pour trouver chemin critique
   - Identifier toutes les tâches critiques
4. **Calculs projet global:**
   - Durée projet: $TE_{projet} = \sum TE_{tâches\ critiques}$
   - Variance projet: $V_{projet} = \sum V_{tâches\ critiques}$
   - Écart-type projet: $\sigma_{projet} = \sqrt{V_{projet}}$
5. **Analyses probabilistes:**
   - Distribution normale: $N(TE_{projet}, \sigma_{projet})$
   - Probabilité terminer avant date D: $Z = \frac{D-TE_{projet}}{\sigma_{projet}}$
   - Consulter table loi normale pour P(Z)

Design Patterns

Catégories

- **Creational:** Création objets (Singleton, Factory, Builder)
- **Structural:** Composition (Adapter, Composite, Proxy)
- **Behavioral:** Algorithmes/responsabilités (Observer, Strategy, Visitor)

Pattern Visitor

- **Problème:** Ajouter opérations sans modifier classes
- **Solution:** Encapsuler opérations dans visiteurs
- **Avantage:** Séparation domaine/comportement