Ingénierie logicielle

Résolution de problèmes réels avec solutions logicielles; méthodes pour développer **systèmes complexes en équipe**; discipline pour spécifier, construire et maintenir des logiciels avec **qualité** (fiabilité, adaptabilité), coûts contrôlés et délais garantis.

Qualités du logiciel

Internes: Maintenabilité (facilité à modifier), **évolutivité** (adaptation aux changements), **réparabilité** (correction d'erreurs), **réutilisabilité** (emploi dans d'autres contextes), **interopérabilité** (interaction avec autres systèmes).

Externes: Correction (conformité aux spécifications), **fiabilité** (continuité du service), **robustesse** (fonctionnement en conditions d'erreur), **performance** (efficacité d'exécution), **ergonomie** (facilité d'utilisation).

Loi de Brooks

"Adding manpower to a late software project makes it later." Adaptation des nouveaux membres prend du temps; communication exponentielle; productivité globale diminuée.

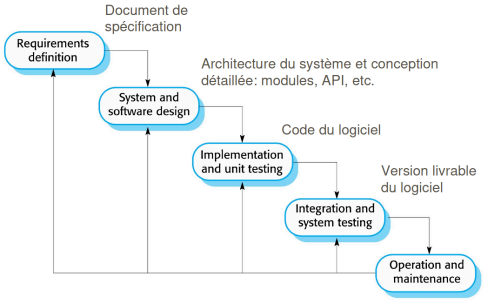
Complexité

Gérer par simplification ou encapsulation. **Échecs** dus à complexité croissante et non-respect des bonnes pratiques d'ingénierie.

Processus de développement logiciel

Activités principales

Structure de production: **Spécification** (définir besoins), **conception** (établir architecture), **implémentation** (coder), **vérification/validation** (tester), **évolution** (adapter).



Spécification

Élicitation (identifier attentes), **spécification** (décrire besoins), **validation** (vérifier cohérence).

Conception

Architectural design (organiser composants), **database design** (structurer données), **interface design** (définir interactions), **component selection** (choisir éléments).

Implémentation

Programmation collective non individualiste; approches variées avec solutions multiples; travail d'équipe essentiel pour code maintenable.

Validation

Vérification: conformité aux spécifications, bugs, performance. **Validation:** réponse aux besoins utilisateur. **Tests:** unitaires, intégration, système, acceptation.

Évolution

Projet (fin délimitée) vs logiciel (évolution continue); software modifiable vs hardware fixe; fusion maintenance-développement.

Programmation

Tactique: solutions rapides, court terme. **Stratégique:** qualité, maintenabilité, long terme.

Types de processus

Cascade: phases séquentielles (Waterfall, V, Spirale); adapté projets stables, gestion facile; rigide, corrections coûteuses.

Incrémentaux: phases entrelacées (RAD, RUP, XP); adaptable, livraison rapide; architecture dégradable, gestion complexe.

Intégration: réutilisation de composants; rapidité, coûts réduits; compromis fonctionnels.

Prototype

Version initiale prouvant faisabilité et anticipant changements.

Maturité des processus

1-**Initial** (processus décrits), 2-**Managed** (appliqués), 3-**Defined** (données collectées), 4-**Quantitatively managed** (performance mesurée), 5-**Optimizing** (processus améliorés). **Amélioration:** Mesure → Analyse → Changement.

Ingénierie des exigences

Définition

Établit **services exigés** (fonctionnel) et **contraintes** (non-fonctionnel).

Types d'exigences

Par audience: Business (décideurs), **User** (utilisateurs), **System** (détails techniques).

Par nature: Fonctionnels (services), **Non-fonctionnels** (contraintes), **Domaine** (aspects légaux).

Élicitation

Processus complexe avec attentes floues; histoires et scénarios facilitateurs; méthodes: interviews, ethnographie, scénarios concrets.

Spécification

Langages: naturel, structuré, UML, mathématique. **Propriétés: valide** (besoins), **cohérente** (sans contradictions), **complète** (couvrant tout), **vérifiable** (testable), **réalisable** (techniquement possible), **compréhensible** (claire), **traçable** (origine identifiable).

Modélisation du système

Définition

Modèles abstraits multi-perspectives avec UML facilitant communication.

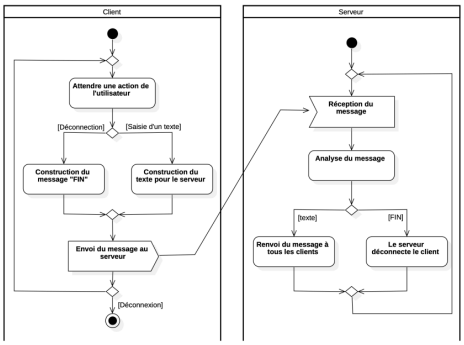
Diagrammes UML

Cas d'utilisation

Interactions système-environnement; acteurs primaires/secondaires; relations: Include (dépendance), Extend (ajout conditionnel).

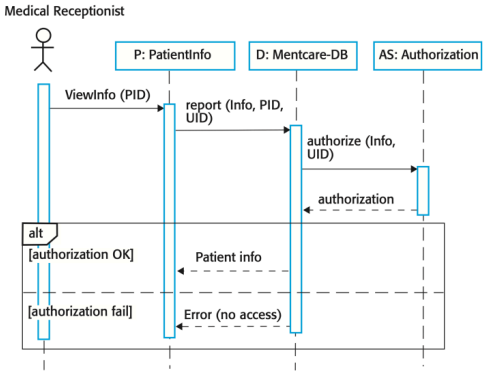
Activité

Processus et flux pour méthodes, algorithmes, workflows.



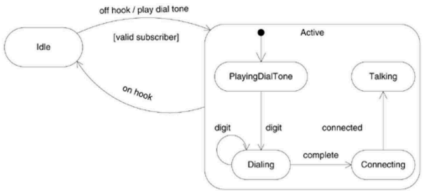
Séquence

Interactions chronologiques entre acteurs et système.



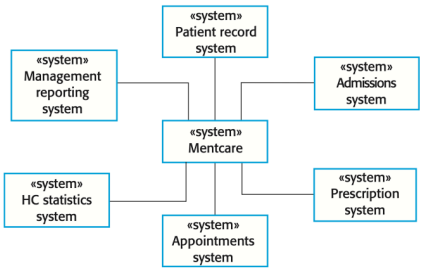
Machine à état

Cycle de vie d'un objet avec états et transitions.

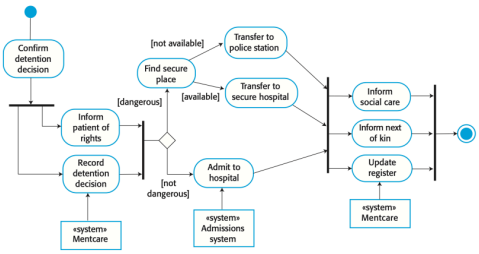


Types de modèles

Contexte: limites système et interactions externes.



Processus: vision métier par diagrammes d'activités.



Interactions: échanges utilisateurs-système. **Structurels:** organisation composants-relations. **Comportement:** pilotés par données (systèmes métiers) ou événements (temps réel).

MDE

Modèle central vs code; génération automatique; meilleure abstraction, pas universelle.

Modélisation du domaine

Objectifs

Identifier classes conceptuelles sans méthodes: classes domaine, associations, attributs.

Techniques

Analyse syntaxique (noms clés); liste catégories (personnes, objets); réutilisation patterns; ignorer détails techniques.

Éléments clés

Associations

Liens entre classes

- **Composition:** cycle de vie commun (●)
- **Agrégation:** cycle indépendant (◯)

Attributs

Simples: nombre, texte, date. **Complexes:** deviennent classes.

Héritage

Sous-concepts partageant attributs; utilisations différenciées.