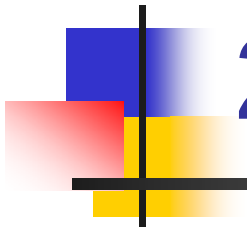
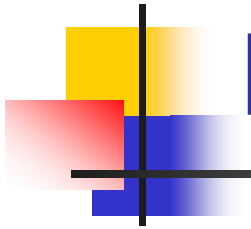


GL - 2
2.3 Ingénierie des Exigences

Lydie du Bousquet
Lydie.du-bousquet@imag.fr

En collaboration avec J.-M. Favre, I. Parissis, Ph. Lalanda, Y. Ledru

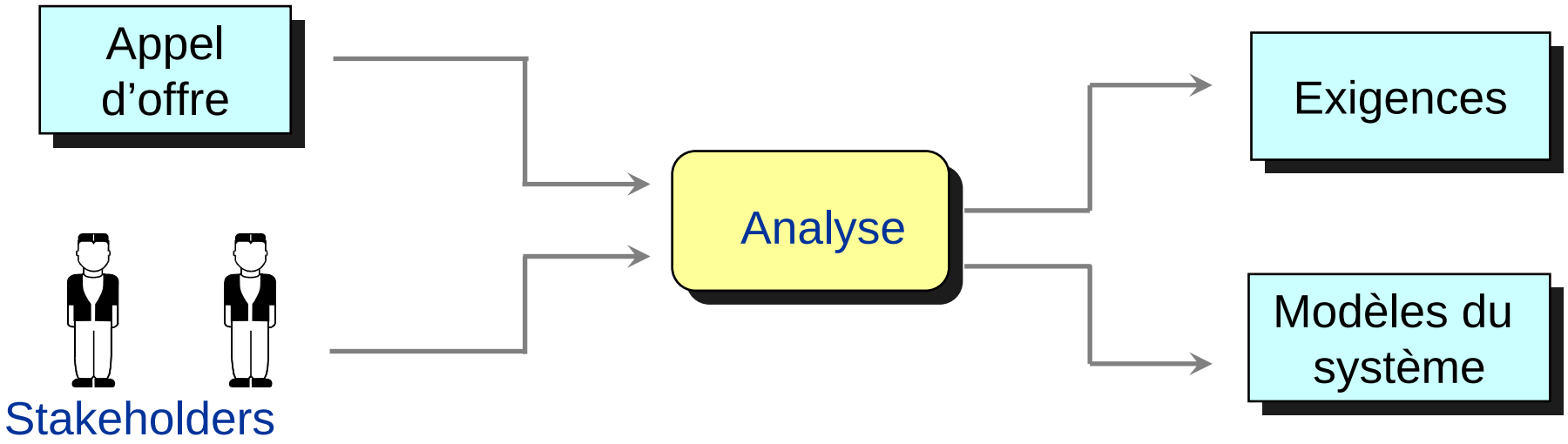


Plan

- **Introduction**
- Exigences
  - Fonctionnelles
  - Non-fonctionnelles
- Traçabilité des exigences
- Ingénierie des exigences
- Phase d'analyse et techniques
- Spécification d'exigences
- Conclusion

Exigences

- Objectifs
  - Établir ce que souhaite le client
  - Spécifier ces exigences



Différentes exigences

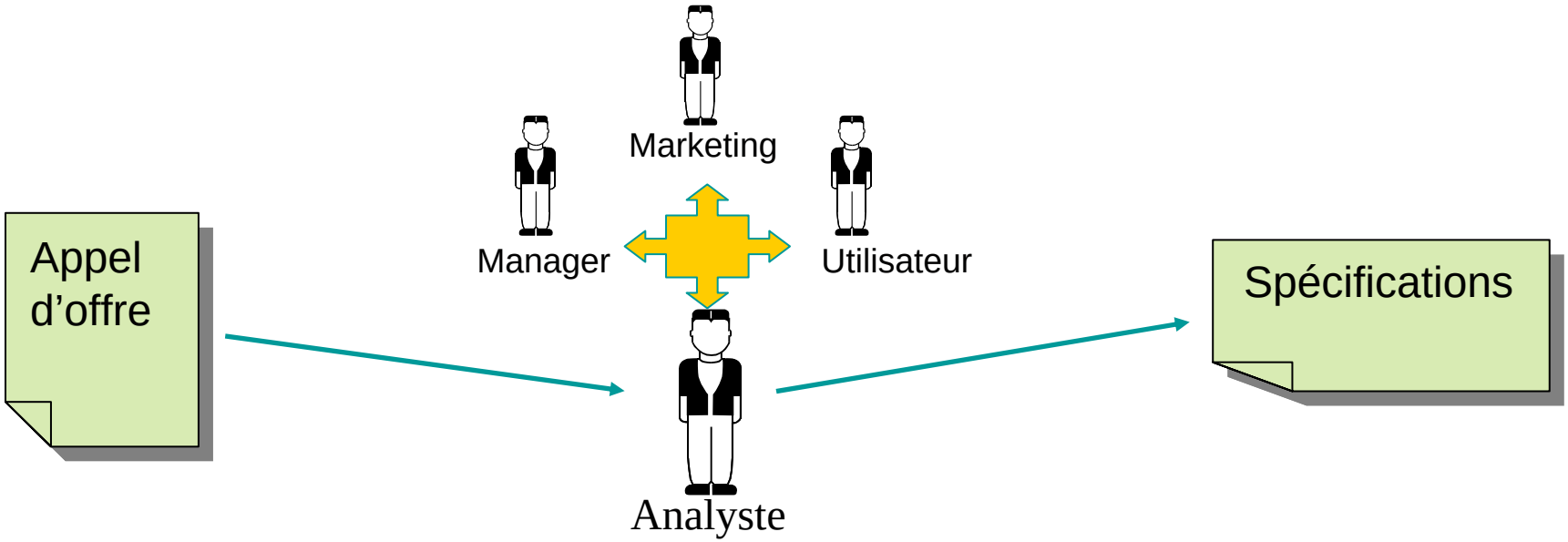
- Les exigences ont une double fonction
  - Pour une réponse à un appel d'offre – doit être ouvert à une interprétation
  - Pour le contrat lui-même – doivent être définies en détail
  - Appel d'offre et cahier des charges : exigences

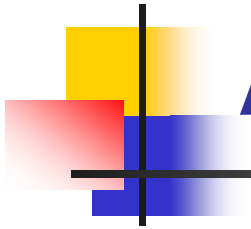
Appel d'offre

Cahier des charges

L’origine des exigences

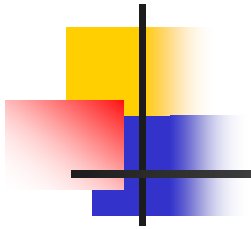
- Les exigences proviennent du côté client
  - Utilisateur, experts du domaine, managers, marketing,...
- Reformulées par la maîtrise d’œuvre
  - Analyste





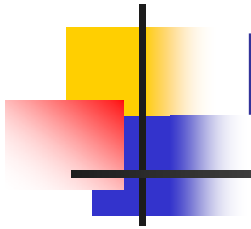
Acteurs

- Plusieurs parties, différents besoins
  - managers et preneurs de décision
  - experts du domaine
  - clients et utilisateurs
  - analystes
  - architecte et développeurs
- Points de vue divergents



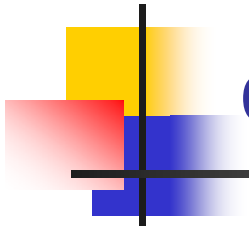
Impact des exigences

- Impact légal
  - Base pour le contrat entre client et maîtrise d'œuvre
- Impact économique
  - Coût de correction dues des exigences incorrectes
- Impact social
  - Des exigences incorrectes peuvent conduire à des désastres
- Impact du point vue de l'usage
  - Acceptation ou rejet d'un logiciel



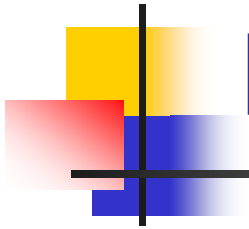
De l’importance des exigences

- Les exigences influent toutes les phases de développement
  - Architecture
  - Conception (design)
  - Définition des tests
  - Acceptation (recette), ...
- Les use cases (cas d’utilisation UML) peuvent être utilisés pour dériver des tests



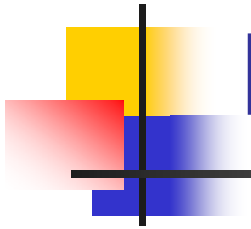
Très difficile de formuler un ensemble complet et consistent d'exigences

- Les clients ne savent pas toujours ce qu'ils veulent
- Les souhaits des clients évoluent
- Il existe des conflits internes
- Certains problèmes ne peuvent être complètement capturés/compris. La compréhension évolue avec le développement
- Clients et maîtrise d'œuvre parlent différents langages
- Il faut gérer une grande masse d'information



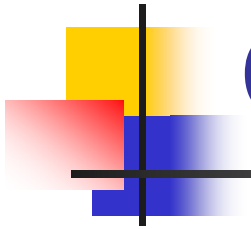
Problèmes possibles

- incomplétude
- inconsistance
- inadéquation
- ambiguïté
- non intelligibilité
- faible structure
- sur spécification



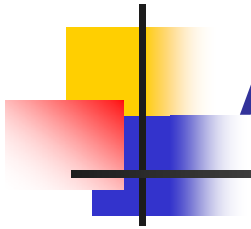
Plan

- Introduction
- **Exigences**
  - Fonctionnelles
  - Non-fonctionnelles
- Traçabilité des exigences
- Ingénierie des exigences
- Phase d'analyse et techniques
- Spécification d'exigences
- Conclusion



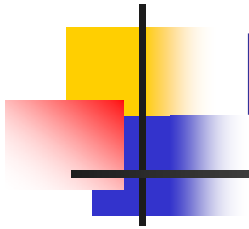
Qu’est-ce qu’une exigence ?

- Description d’une **capacité** ou d’une **contrainte**
  - un service offert par le logiciel
  - une contrainte sous laquelle il doit opérer ou doit être développé
- Exprime
  - Tout ce que le client veut
  - Tout ce qui est nécessaire pour le développement du logiciel



Attention

- Ne pas confondre le « **quoi** » et le « **comment** »



Exigences fonctionnelles

- Services offerts
  - Description d'une fonction ou son comportement
  - Une propriété générale du système
  - IHM attendue, ...
- Exemples
  - Le logiciel doit gérer le système de prêt de la bibliothèque
  - Un abonné doit payer 20 euros par ans
  - Le logiciel doit afficher la liste des emprunts pour un abonné, ...

Contraintes (exigences non fonctionnelles)

- Une contrainte sous laquelle un logiciel doit fonctionner ou être développé

Qualité

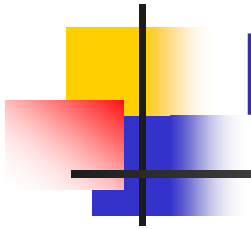
- Performance
- Fiabilité
- Utilisabilité
- Maintenabilité

Développement

- COTS (OS, middleware, ...)
- Méthodes
- Outils
- Standards

Domaine

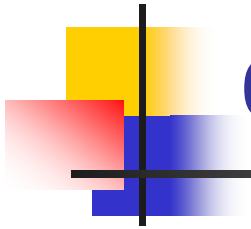
- Usage
- Lois



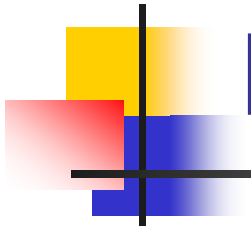
Plusieurs niveaux d'abstraction

- Exprimables sous la forme
  - d'expression de haut niveau
  - à l'expression sous la forme de spécification formelle détaillée
- Trois niveaux abstraction
  - Exigences relatives au domaine (besoins)
  - Exigences relatives à l'utilisateur
  - Exigences relatives au système

Exigences domaine/utilisateur/système



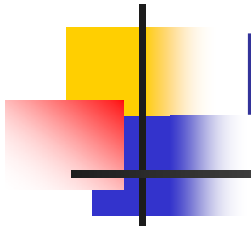
- Les besoins (exigences relatives au domaine)
  - Les propriétés d'un domaine influencent une large gamme de logiciels
  - Écrits **par** les clients
- Exigences utilisateur
  - Services offerts et contraintes opérationnelles
  - Décrites en langue naturelle et avec des diagrammes
  - Écrites **pour** les clients
- Exigence système
  - Document structuré détaillant les descriptions des services systèmes offerts
  - Contrat entre les parties



Exemples

- Besoins (Exigences relatives au domaine)
  - 1. Le système doit gérer les accords de "copyright"

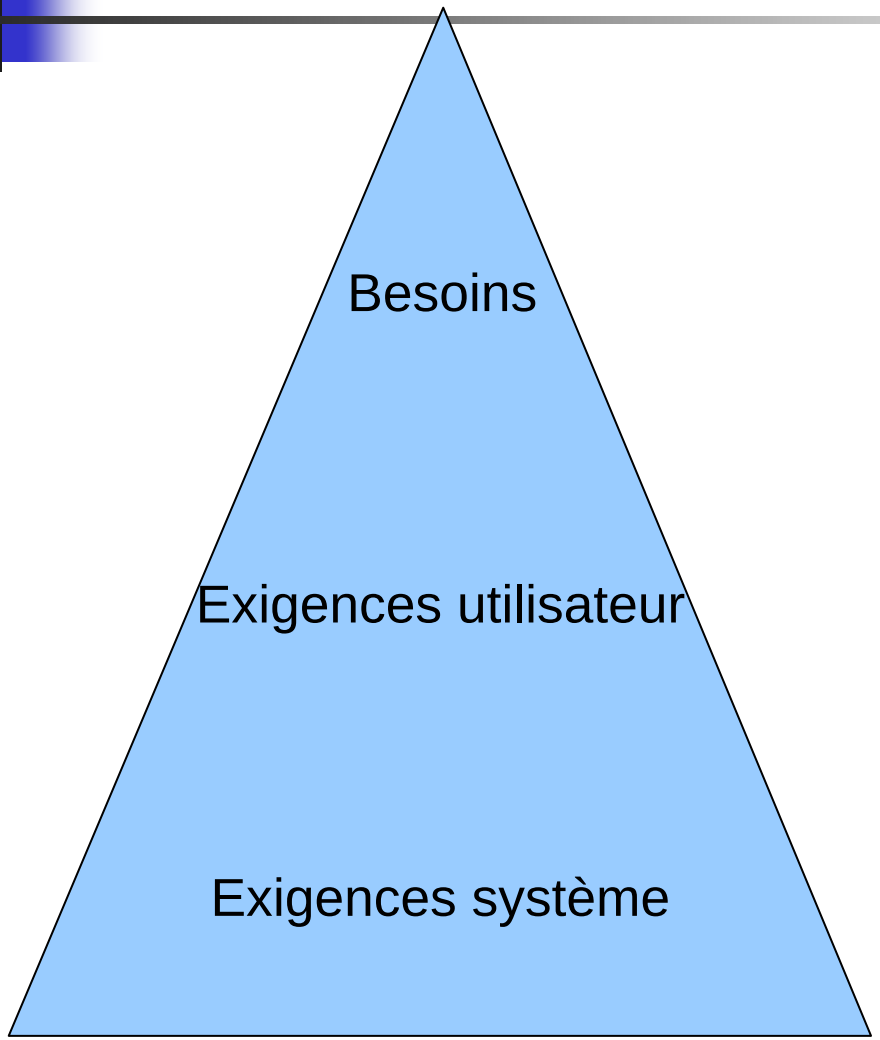
- Exigences relatives à l'utilisateur
  - 1. Le système devra garder trace de toutes les données nécessaires à la gestion du "copyright"



Exemples

- Exigences système
  - 1.1. pour toute requête, l'utilisateur doit remplir un formulaire avec son nom et sa demande
  - 1.2. les formulaires doivent être sauvegardés 5 ans
  - 1.3. Les formulaires doivent être indexés par utilisateur, par document demandé et par fournisseur
  - 1.4. Le système doit maintenir un log de toutes les requêtes
  - 1.5. Pour les documents avec droit d'auteur, un détail des sommes dues sera envoyé chaque mois

Niveaux d’abstraction



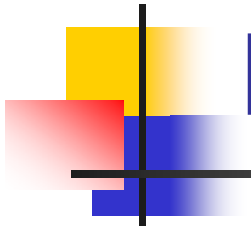

Manager


Utilisateur


Développeurs

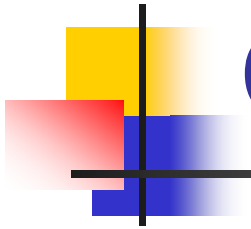
Propriétés des exigences : Une exigence doit être

- Concise
- Non ambiguë
- Compréhensible par l'utilisateur et la maîtrise d'œuvre
- Structurée
- Vérifiable
  - Les exigences doivent être écrites de telle sorte qu'elles doivent être compatibles avec les méthodes de vérification qui peuvent être utilisées



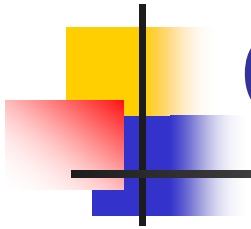
Plan

- Introduction
- Exigences
  - **Fonctionnelles**
  - Non-fonctionnelles
- Traçabilité des exigences
- Ingénierie des exigences
- Phase d'analyse et techniques
- Spécification d'exigences
- Conclusion



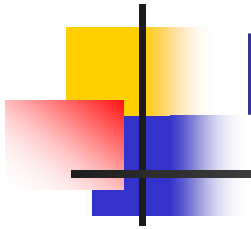
Objet, fonction et état – Davis, 93

- Une exigence fonctionnelle concerne ...
  - un objet
    - Un abonné est identifié par son nom et sa date de naissance
  - ou une fonction liée au logiciel
    - Un abonné peut emprunter 5 livres
  - ou un état
    - Un livre est disponible, emprunté ou perdu
  - ou plusieurs de ces éléments en même temps



Objets – Davis, 93

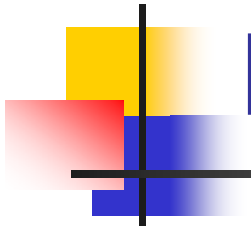
- Un objet est une entité clairement définie dans le domaine considéré
  - Correspond à un concept du métier, pas à un concept d'implantation
  - Seulement les concepts ayant un sens pour le logiciel ≠ d'une analyse du domaine
- Les exigences spécifient des objets et limitent leurs portées
  - « Le système doit afficher le nom de l'abonné »
  - « Un abonné peut emprunter 5 livres et 2 revues »
  - « les livres ont un titre et un ou plusieurs auteurs »



Fonctions – Davis, 93

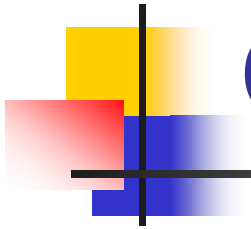
- Activités clairement définies dans le domaine
  - Tâches, services, processus
  - On se limite aux fonctions réalisées par le logiciel (suite à une action utilisateur ou à un événement)

- Les exigences spécifient des fonctions et les détaillent
  - « Le système peut afficher le nom des abonnés »
  - « Le système effectue les emprunts des abonnés ayant droit »
  - « Le système autorise les emprunts pour 5 semaines »



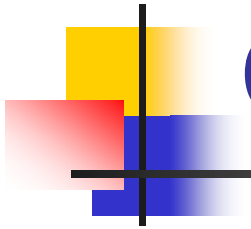
États – Davis, 93

- Caractérise une situation d'une entité
- Peut s'exprimer sous la forme d'un prédicat
  - Qui influence le comportement de cette entité
  - Aspect temporel important
    - L'état peut être transitoire
    - Il peut capturer un certain historique
- Certaines exigences spécifient des états et les détaillent
  - Un livre peut être disponible, emprunté ou perdu
  - Un client est caractérisé par le nombre de livres empruntés
  - Un client est caractérisé par sa situation vis à vis du paiement de l'abonnement



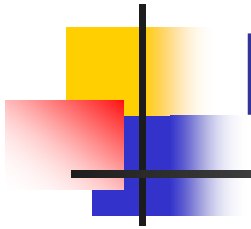
Objets, fonctions, états

- Certaines exigences établissent des relations entre les objets, les fonctions et les états
  - Un client peut emprunter un livre quand il a payé son abonnement et s'il a moins de 5 prêts en cours
- Les méthodes d'analyse se concentrent sur un aspect
  - Objet
  - Fonction
  - État



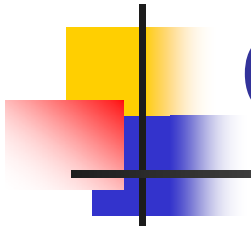
Questions à poser – Pfleeger

- Fonctionnalités
  - Qu'est-ce que le système va faire ?
  - Quand le système devra-t-il le faire ?
  - Existe-t-il plusieurs modes de fonctionnement ?
  - Quelles sont les bonnes réactions aux stimuli possibles ?
- Données
  - Quel doit être le format des données ?
  - Combien de temps les données doivent-elles être conservées ?



Plan

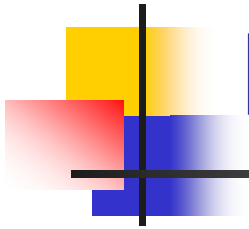
- Introduction
- Exigences
  - Fonctionnelles
  - **Non-fonctionnelles**
- Traçabilité des exigences
- Ingénierie des exigences
- Phase d'analyse et techniques
- Spécification d'exigences
- Conclusion



Quantification

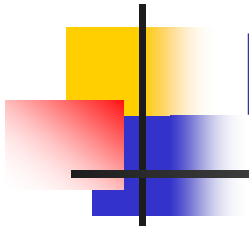
- Exigences NF sont liées aux propriétés globales qui sont difficiles à vérifier (dans leur première formulation)
  - Non précises, plusieurs interprétations possibles
    - “Le système doit être facile à utiliser”
  - Source de conflits
    - “Le système doit être robuste et performant”

- Il est nécessaire de quantifier ces exigences
  - Avec des métriques qui peuvent être mesurées, testées



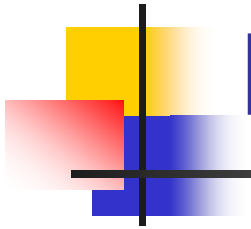
Exemple

- Première formulation
  - Le système doit être facile à utiliser pour un contrôleur expérimenté et doit être organisé pour limiter le nombre d'erreur.
- Reformulation
  - Un contrôleur avec plus de 5 ans d'expérience doit être capable d'utiliser le système après une formation de 2 heures. A l'issue de cette formation, le nombre moyen d'erreurs ne doit pas dépasser deux par jour.



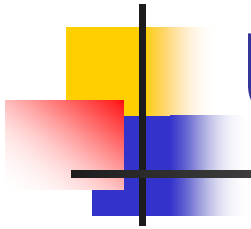
Performance

- Questions à poser
  - Quelles sont les contraintes sur la vitesse d'exécution, le temps de réponse, ou le temps de rafraîchissement ?
  - Quelles mesures d'efficacité vont être appliquées pour évaluer l'usage des ressources ou le temps de réponse ?
  - Combien de données vont transiter à travers le système ?
  - Avec quelle fréquence les données vont être reçues ou émises ?



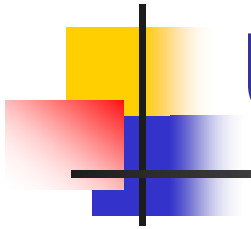
Performance

- Exemples d'exigences
  - Nombre de transactions par seconde
  - Temps de rafraîchissement
  - Temps de réponse pour un « schéma » d'occurrence d'événements
- Pour être plus précis
  - Pour chaque entrée, le domaine d'entrée (range)
  - Fréquence d'arrivée des événements
  - Que faire quand trop d'événements arrivent ?
    - erreur, ignorance, services dégradés



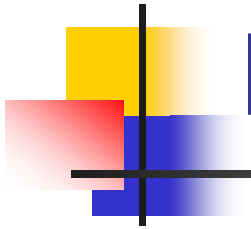
Utilisabilité

- Questions à poser
  - Quel type de formation sera nécessaire pour chaque type d'utilisateur ?
  - A quel point il sera facile pour un utilisateur de comprendre et d'utiliser le système ?
  - A quel point il sera difficile pour un utilisateur d'utiliser incorrectement le système ?



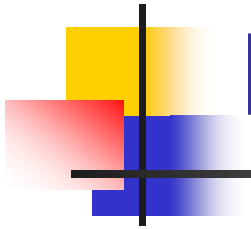
Utilisabilité

- Exemple d'exigences
  - Interfaces
  - Messages d'erreur
  - Techniques nécessaires pour aider les utilisateur et améliorer leur usage
- Il existe aujourd'hui des outils pour prototyper rapidement des interfaces



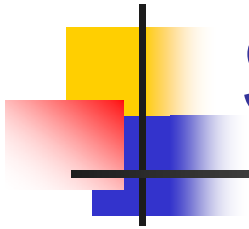
Fiabilité et disponibilité

- Le système doit –il détecter et isoler les fautes ?
- Quel est le « Meantime Between failures » attendu ?
- Quel est le délai maximum de temps pour redémarrer le système après une panne ?
- A quelle fréquence le système doit être « backuppé » ?
- Les backups doivent ils être stockés dans un endroit différent ?
- Quelles précautions en cas d'incendie ?



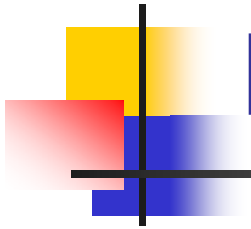
Fiabilité et disponibilité

- Exemples d'exigences
  - Nb max de fautes par SLOC pendant l'intégration
  - Temps moyen sans défaillance



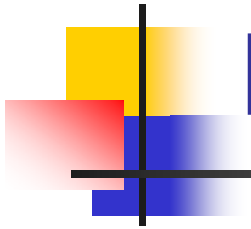
Sécurité

- L’accès au système ou aux informations doit-il être contrôlé ?
- Les données de chaque utilisateur doivent-elles être isolées ?
- Les programmes de chaque utilisateur doivent-ils être isolés les uns des autres / de l’OS ?
- Faut-il prendre des précautions contre le vol ou le vandalisme ?



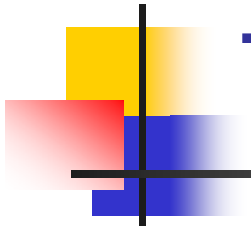
Maintenabilité

- La maintenance ne concernera-t-elle que la correction de bug ou aussi l'amélioration du système ?
- Quand et comment pourra-t-on faire évoluer le système ?
- A quel point il sera facile d'ajouter des fonctionnalités au système ?
- A quel point il sera facile de porter le système d'une plateforme à une autre (machine, OS) ?



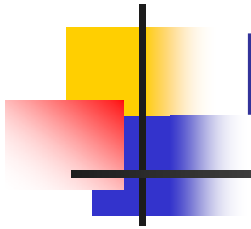
Plan

- Introduction
- Exigences
  - Fonctionnelles
  - Non-fonctionnelles
- **Traçabilité des exigences**
- Ingénierie des exigences
- Phase d'analyse et techniques
- Spécification d'exigences
- Conclusion



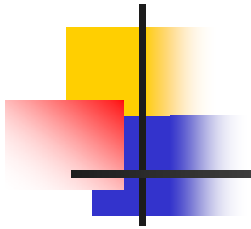
Traçabilité des exigences

- Des liens doivent être maintenus entre
  - Les exigences et les besoins originaux
  - Les exigences abstraites et les exigences dérivées
  - Les exigences et l'implantation
  - Les exigences dépendantes (à un niveau d'abstraction donné)
  - Les exigences et les tests
- Note: on ne peut éviter l'évolution des exigences



Exemple

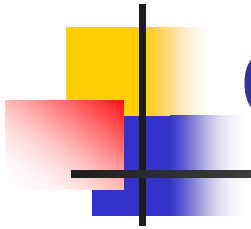
- Un jour un constructeur automobile décida de réduire les coûts sur l'un de ses modèles phares.
- Une équipe se pencha sur les spécifications du modèle et chercha des axes de réduction des coûts.
- Quelqu'un s'avisa que le modèle était conçu pour résister à un vent arrière, avec de la pluie, de 200km/h (exigence produit) ce qui entrainait des coûts de fabrication importants.
- On décida donc de changer cela en allégeant la fermeture du coffre à bagage situé à l'arrière (exigence composant).
- Ce n'est qu'à l'automne, chez les concessionnaires, qui trouvaient de l'eau dans les coffres, que l'on s'avisa que les voitures étaient acheminées par train Express (exigence partie prenante).



Importance de la traçabilité

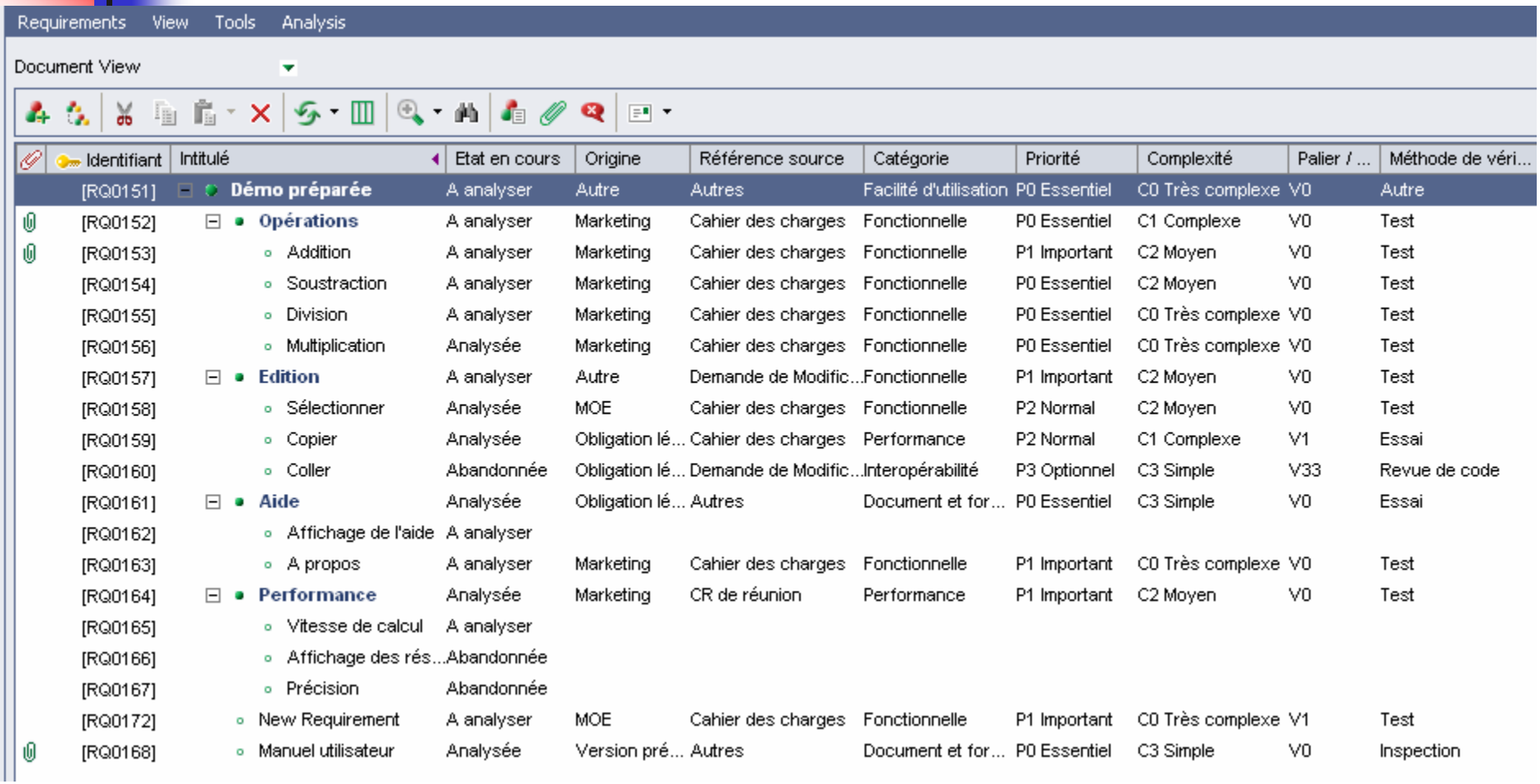
- La traçabilité permet
  - De revenir aux besoins initiaux
  - D'évaluer le coût d'une évolution
  - D'évaluer la couverture
    - Pertinence des exigences
    - Progrès du projet
- Il faut des outils

Outils pour l'ingénierie des exigences



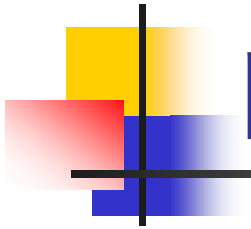
- Création d'exigences
  - Définition d'exigences et des propriétés associées
- Traçabilité des exigences
  - Lien vers les documents externes
- Lien avec des outils basés sur les cas d'utilisation (UML par ex.)
- Lien avec des outils de test
- Génération de documents
- Vérification de couverture (vers UML ou les tests)

Exemple d'outil

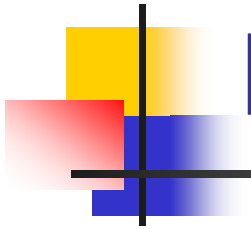


Identifiant	Intitulé	Etat en cours	Origine	Référence source	Catégorie	Priorité	Complexité	Palier / ...	Méthode de véri...
[RQ0151]	**Démo préparée**	A analyser	Autre	Autres	Facilité d'utilisation	P0 Essentiel	C0 Très complexe	V0	Autre
[RQ0152]	**Opérations**	A analyser	Marketing	Cahier des charges	Fonctionnelle	P0 Essentiel	C1 Complexe	V0	Test
[RQ0153]	Addition	A analyser	Marketing	Cahier des charges	Fonctionnelle	P1 Important	C2 Moyen	V0	Test
[RQ0154]	Soustraction	A analyser	Marketing	Cahier des charges	Fonctionnelle	P0 Essentiel	C2 Moyen	V0	Test
[RQ0155]	Division	A analyser	Marketing	Cahier des charges	Fonctionnelle	P0 Essentiel	C0 Très complexe	V0	Test
[RQ0156]	Multiplication	Analysée	Marketing	Cahier des charges	Fonctionnelle	P0 Essentiel	C0 Très complexe	V0	Test
[RQ0157]	**Edition**	A analyser	Autre	Demande de Modific...	Fonctionnelle	P1 Important	C2 Moyen	V0	Test
[RQ0158]	Sélectionner	Analysée	MOE	Cahier des charges	Fonctionnelle	P2 Normal	C2 Moyen	V0	Test
[RQ0159]	Copier	Analysée	Obligation lé...	Cahier des charges	Performance	P2 Normal	C1 Complexe	V1	Essai
[RQ0160]	Coller	Abandonnée	Obligation lé...	Demande de Modific...	Interopérabilité	P3 Optionnel	C3 Simple	V33	Revue de code
[RQ0161]	**Aide**	Analysée	Obligation lé...	Autres	Document et for...	P0 Essentiel	C3 Simple	V0	Essai
[RQ0162]	Affichage de l'aide	A analyser							
[RQ0163]	A propos	A analyser	Marketing	Cahier des charges	Fonctionnelle	P1 Important	C0 Très complexe	V0	Test
[RQ0164]	**Performance**	Analysée	Marketing	CR de réunion	Performance	P1 Important	C2 Moyen	V0	Test
[RQ0165]	Vitesse de calcul	A analyser							
[RQ0166]	Affichage des rés...	Abandonnée							
[RQ0167]	Précision	Abandonnée							
[RQ0172]	New Requirement	A analyser	MOE	Cahier des charges	Fonctionnelle	P1 Important	C0 Très complexe	V1	Test
[RQ0168]	Manuel utilisateur	Analysée	Version pré...	Autres	Document et for...	P0 Essentiel	C3 Simple	V0	Inspection

Les outils actuels parmi les plus utilisés

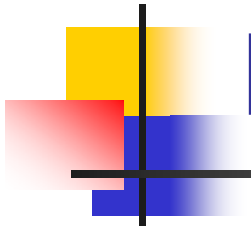


- DOORS (Telelogic)
- RequisitePro (IBM/Rationale)
- Analyst Pro (Goda Software)



Plan

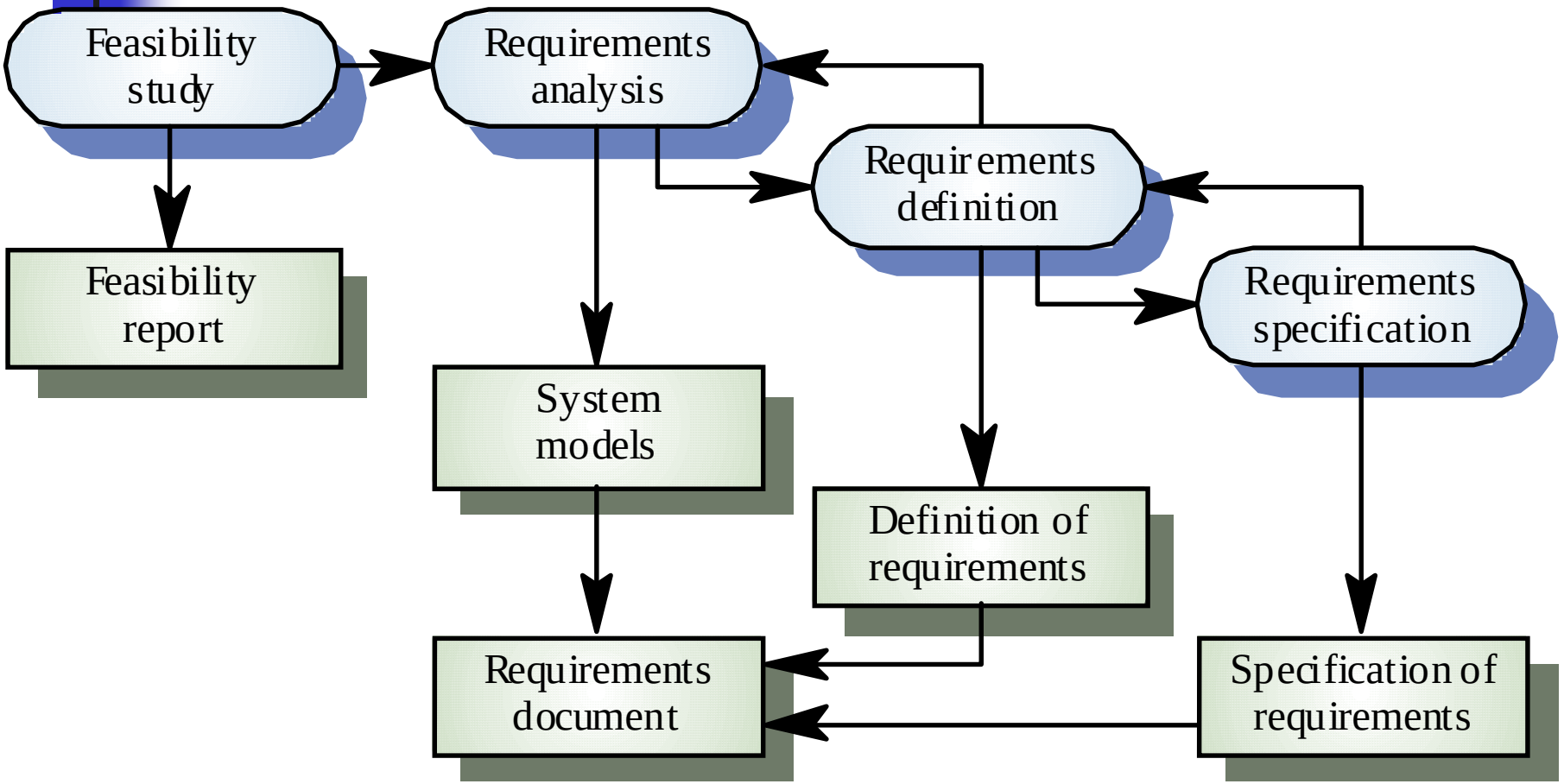
- Introduction
- Exigences
  - Fonctionnelles
  - Non-fonctionnelles
- Traçabilité des exigences
- **Ingénierie des exigences**
- Phase d'analyse et techniques
- Spécification d'exigences
- Conclusion

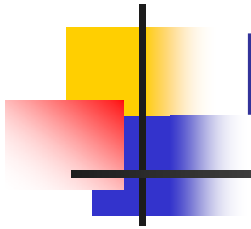


Des processus mêlés

- Étude de faisabilité
- Analyse
- Définition
- Spécification
- Vérification, validation,
- Gestion du changement

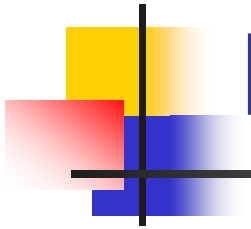
Un processus itératif –Sommerville





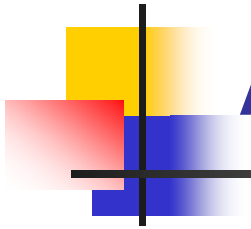
Étude de faisabilité

- Étude courte et focalisée
- Pour répondre aux questions :
  - Le système répond-il aux objectifs business ?
  - Le système peut-il être développé avec les technos actuelles ?
  - Le nouveau système pourra-t-il être intégré aux systèmes existants ?
  - Peut-on utiliser les outils disponibles ?
- Documents
  - En entrée: appel d'offre
  - En sortie: rapport (OK/KO) avec recommandations



Étude de faisabilité : exemple

- CISCO s'intéressent au marché des passerelles domestiques
- Côté marketing
  - Qui sont les acteurs principaux ?
  - Qui sont les nouveaux acteurs ?
  - Quels sont les profits (marges) ?
  - Évaluation des gains et risques ?
- Côté technique
  - Le hardware existant peut-il être réutilisé ?
  - Les processus de développement peuvent-ils être réutilisés ?
  - Faut-il une nouvelle BD embarquée ?
  - Évaluation des coûts et des problèmes potentiels ?

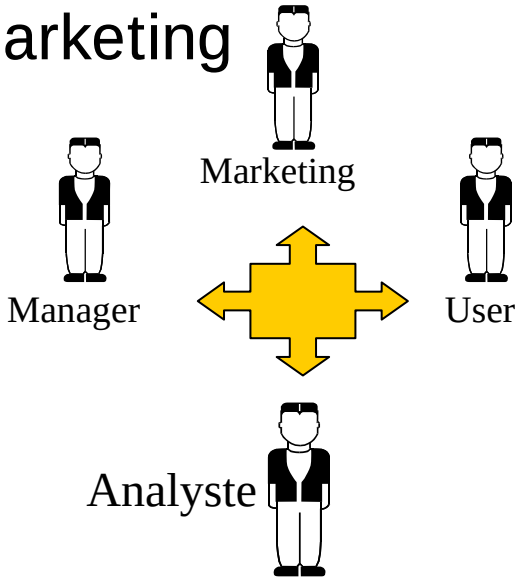


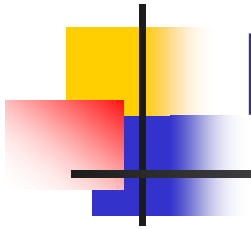
Analyse

- Pour collecter des données sur le logiciel
  - Comprendre le domaine et l'environnement
  - Identifier les objectifs et les conflits
- L'analyse est une phase ouverte
  - Impliquer autant de stakeholders que possible
  - Éviter les idées préconçues
  - Attention à l'autocensure
  - Attention à l'apparente simplicité des objectifs et des besoins

Analyse: exemple

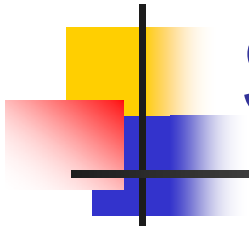
- Après l'étude de faisabilité, le projet CISCO est lancé. Le but de l'analyse est d'écrire les exigences
- Compréhension du problème
  - Discussion avec le service de marketing
  - Réunion avec FT
  - Réunion avec des utilisateurs
  - Réunion avec les dev





Définition des exigences

- But: confronter les stakeholders avec les exigences possibles et établir une liste d'exigences valides
  - Comparaison d'options alternatives
  - Résolution de conflits
  - Négociation des meilleurs compromis
  - Obtenir un agrément partagé
- C'est une phase de « fermeture »
  - Regroupement des stakeholders
  - Réduction des exigences à un cœur stable

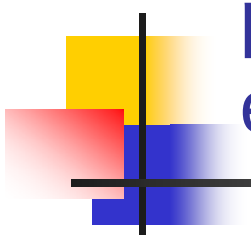


Spécification des exigences

- But: définir clairement le logiciel à produire
  - Documentation compréhensible par toutes les parties -> base contractuelle
- Phase de synthèse
  - Écriture des exigences
  - Structuration des exigences (type, niveau d'abstraction)
  - Vérification de la consistance, complétude, ...
  - Assignation de priorités possible
  - Validation

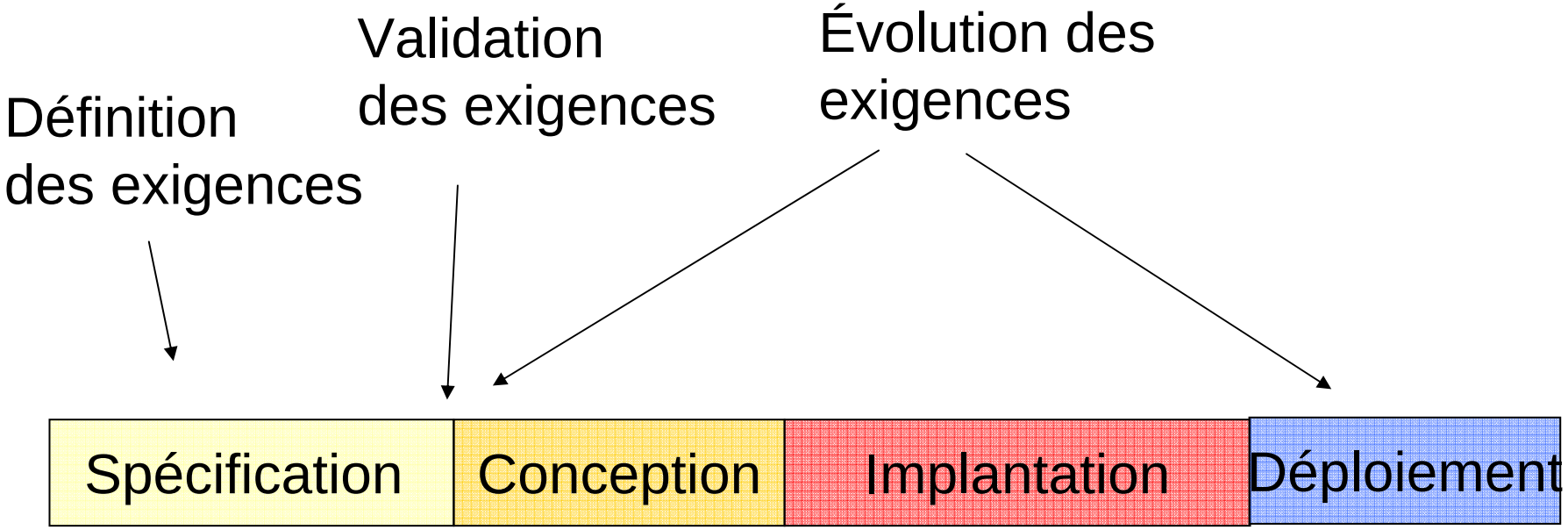
Validation des exigences :

pour vérifier que touts les exigences écrites expriment des besoins clients

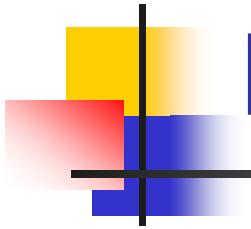


- Complétude
- Consistance
- Adéquation
- Précision
- Pertinence
- Compréhensibilité
- Bonne structuration
- Modifiabilité
- Traçabilité
- Mesurabilité

Exigence et cycle de vie



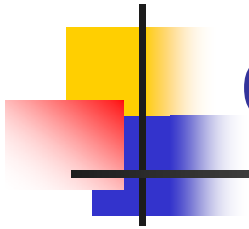
Ingénierie des exigences



Evolution des exigences

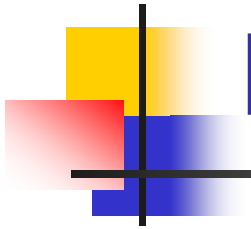
- On ne peut pas empêcher l'évolution des exigences
- Possible si le travail initial de collecte, d'analyse et de validation a été fait de façon rigoureuse
- Évaluer les impacts
  - Mécanismes de traçabilité nécessaires
- Outil nécessaire
  - Au moins pour automatiser la gestion des liens de traçabilité

Problèmes liés à l'évolution des exigences



- Évolution sans analyse d'impact
- Modification par la hiérarchie
- Nouveaux stakeholders (très fréquent)
- Pas de formalisme, de processus pour l'évolution
- Faible traçabilité

- Un bon processus initial ne suffit pas: besoin d'un processus de gestion d'évolution

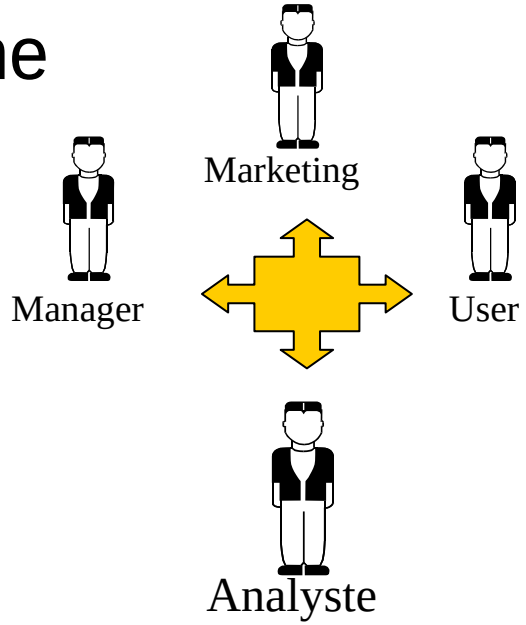


Plan

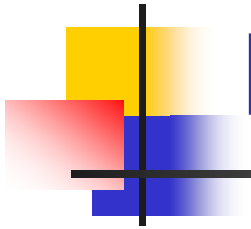
- Introduction
- Exigences
  - Fonctionnelles
  - Non-fonctionnelles
- Traçabilité des exigences
- Ingénierie des exigences
- **Phase d'analyse et techniques**
- Spécification d'exigences
- Conclusion

Analyse

- Identifier les exigences est une activité complexe
- Combinaison de compétences
  - Communication
  - Politique
  - Synthèse
  - Compréhension rapide

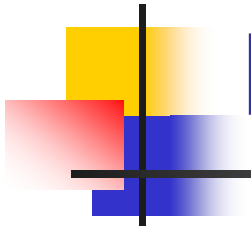


- Des techniques d'analyse peuvent être utilisées



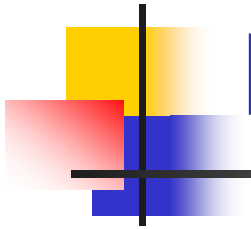
Défis relatifs à l'analyse

- Gestion des différents acteurs (et conflits)
  - Comment communiquer avec eux ?
  - Comment identifier et résoudre les conflits ?
- Gestion de la quantité d'information
  - Structuration
  - Différentes vues ?
- Obtenir toutes les exigences
- Ne pas commencer la conception



Différents acteurs

- Exploiter toutes les sources disponibles
  - Stakeholders
  - Systèmes existants (legacy system)
  - Standards
  - Autorité de régulation, de certification ...
  - Domaine



Différents acteurs - exemple

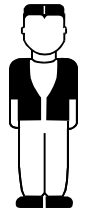
- Cas d'une bibliothèque
  - Les employés de la bibliothèque (utilisateurs directs)
  - Clients (utilisateur indirect)
  - Directeur de la bibliothèque (utilisateur indirect)
  - Administrateur du système
  - Editeurs de livres
  - Législation (loi sur l'emprunt des livres)

Différents acteurs – différent intérêts

Utilise

Paye

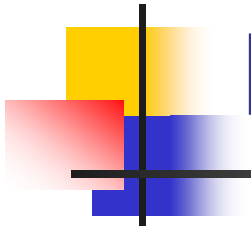
Développe



<u>Utilisateur</u>
- fonctions
- utilisabilité
- temps de réponse
- fiabilité

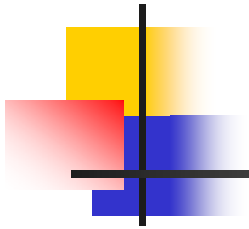
<u>Client</u>
- temps de dév
- coûts
- Fonctions
- maintenabilité
- évolutivité
- fiabilité

<u>Partie développement</u>
- profits
- réutilisation de l'existant
- capitalisation
- maintenabilité : selon le contrat.
- produit récurrent ou « one shot »



Différents acteurs – politique

- Joue un rôle important
  - De nombreuses décisions sur les exigences sont prises pour satisfaire des intérêts personnels
  - Les personnes peuvent être contre un développement et adopter une attitude agressive ou passive
  - Comportement humain (attitudes protectrice, en général)
- Gestion du conflit
  - Intérêt personnels et les oppositions sont difficiles à percevoir...
  - et à résoudre



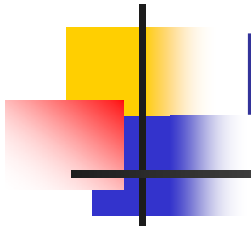
Index Group Cambridge, 90

- Étude sur 100 projets

- Conclusion
  - « Systems developers are operating amid turf battles, historical bickering, low credibility and the difficulty in pinning down ever-changing systems requirement»

Importance de la communication

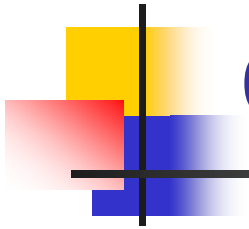
- Obtenir les besoins du client/utilisateur est difficile
  - Ils n'ont pas encore conscience de ce qu'ils vont avoir
  - Il y a confusion entre leur besoins et ce qu'ils ont déjà avec les systèmes existants
  - Il y a une différence entre ce qui est voulu et ce dont ils ont besoins
  - Ils ne souhaitent pas s'investir
    - « montrez moi ce que vous pouvez faire ... »
    - « Vous êtes les spécialistes »



Exemple de communication

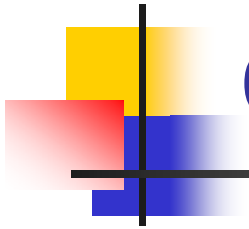
- Le client/l'utilisateur demande à l'analyste de changer un algorithme qui ne fonctionne pas correctement sur le système existant
  - Analyste: "Avec quelle fréquence cet algo est utilisé"?
  - Utilisateur (client): "jamais"!
  - Alors l'analyste ignore la demande
- Bien sûr, la raison pour laquelle l'algo n'est jamais utilisé est... que l'algo est incorrect !
  - Attention aux questions biaisées
  - Attention à ne pas décider pour les autres

Importance de la communication

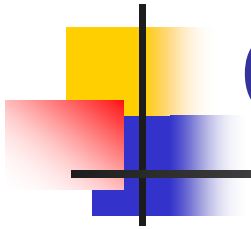


- Attention aux “slang”
  - Les spécialistes domaine utilisent un langage subtile sans le savoir
- Omission d’information volontaire
  - Les spécialistes domaine ne parlent pas des concepts « évidents »
- Omission d’information involontaire
  - Exigences relatives à l’organisation

Différents types de connaissances

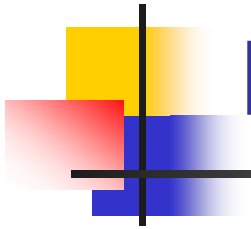


- Connaissances tacites
  - Connaissances dont vous n'avez pas conscience
- Connaissances semi-tacites
  - Connaissances dont vous avez vaguement conscience (peut être exprimé si questions)
- Connaissances non-tacites
  - Connaissances exprimées directement



Quantité d'information

- Techniques d'analyse pour
  - Structurer l'information
  - Améliorer/systématiser la recherche d'info
  - Vérifier la cohérence, complétude, ...
- Techniques connues
  - Orientées Objet
  - Orientées Fonction
  - déclaratives



Recherche d’information (data mining)

- Utilisation des systèmes existants pour déduire les exigences
  - Nécessite d’avoir accès aux documents, au code, ...
- Avantages
  - Autonomie
  - Réutilisation (d’analyse(s) précédente(s))
- Limites
  - Réutilisations des parties bonnes et moins bonnes
  - Contraintes non identifiables
  - Une re-conception propre n’est pas possible

Techniques d'analyse

Ethnographie

- Immersion dans l'environnement de l'utilisateur
  - Nécessite l'accès à l'environnement de travail
- Avantages
  - Découvertes des exigences implicites et des contraintes
  - Utilisation réussie pour le contrôle du trafic aérien, du métro, travail de bureau (≠ entre le travail présumé et le travail réel, Suchman 87)
- Limites
  - Collecte de beaucoup d'informations inutiles
  - Difficile de trouver des informations pertinentes
  - Absence d'exigences organisationnelles et de domaine

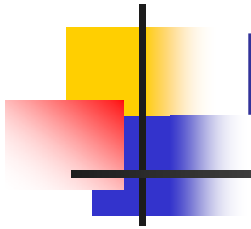
Techniques d’analyse

Analyse de tâche (Protocol analysis)

- Un utilisateur effectue une tâche en expliquant constamment ce qu’il fait
  - Nécessite l’identification des tâches importantes
- Avantages
  - Informations précises sur les activités
  - Étude des personnes au travail
- Limites
  - Collecte de beaucoup d’informations inutiles
  - Certaines actions sont difficiles à comprendre
  - Approche longue et méticuleuse

Techniques d’analyse

Brainstorming

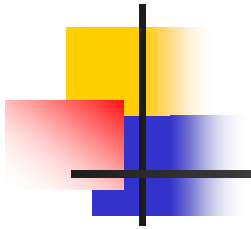


- Groupe de personnes pour générer autant d’idées que possible
  - Créativité, pas d’évaluation
  - Nécessite de former un groupe homogène
- Avantages
  - Attraper des aspects non obtenus par des approches traditionnelles
- Limites
  - Dépend beaucoup du groupe
  - Non systématique
  - Peut sortir complètement du scope et donc être inutile

Techniques d'analyse

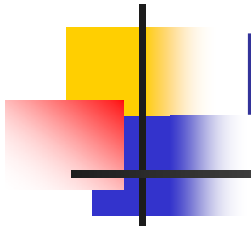
Interviews non structurées

- Interview des stakeholders majeurs sans questionnaire prédéfini
- Avantages
  - Pas ou peu de préparation
  - Les aspects Importants sont directement identifiés par les stakeholders
- Limites
  - Beaucoup de temps peut être passé sur des détails mineurs
  - L'information obtenue peut être difficile à analyser / structurer
  - Nécessite des compétences et de l'expérience
  - Les conflits ne sont en général pas révélés (ni résolus)



Interviews structurées

- Liste de questions pré-définies
  - Les questions sont préparées avant les interviews
- Avantages
  - Systématique (même questions pour tous)
  - Les interviews sont sous contrôle
- Limites
  - Le canevas prédéfinis peut oublier certains points très importants
  - Les conflits ne sont en général pas révélés (ni résolu)

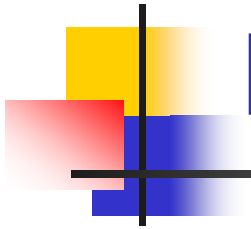


Exemples

- Qui a demandé ce logiciel ?
- Qui va l’utiliser ? Qui va le payer ?
- Quels sont les intérêts économiques?

- Qu’est-ce qu’une bonne solution ?
- Quel est l’environnement de travail ?
- Quelles sont les principales contraintes ?

- Êtes vous la bonne personne pour répondre ?
- Est-ce que mes questions ont un sens ?
- Ai-je oublié quelque chose ?

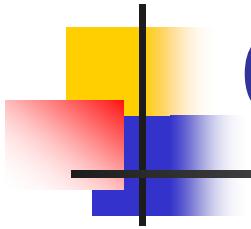


Mener un interview

- Compétences nécessaire
  - Ouverture d'esprit
  - Pas d'idée préconçue
  - Capacité d'écoute
  - Capacité à générer une discussion, à proposer des idées, ...
    - Poser une question pour lancer le débat
    - Parler d'un prototype à faire ensemble
    - Donner un contexte restreint de discussion
    - Utiliser le tableau !

Techniques d’analyse

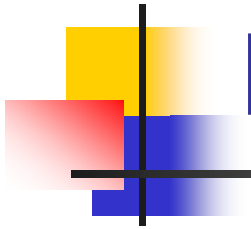
Cas d’utilisation (use cases)



- L’analyse peut être partiellement orientée pour l’obtention des cas d’utilisation
- Les cas d’utilisation présentent les interactions entre le système et les utilisateurs
  - Construits avec les stakeholders, en particulier les utilisateurs
- Avantages
  - Facilite la discussion (très concrète)
  - Investissement fort de l’utilisateur
- Limites
  - Toutes les exigences ne peuvent pas s’exprimer par des cas d’utilisations
  - Des conflits et contraintes peuvent ne pas être révélées

Techniques d’analyse

RAD workshop (Rapid Application Dev)



- Entre 8 et 20 personnes
  - sous la direction d’un facilitateur indépendant
  - Constitution de sous groupes pour poursuivre le travail
  - Gros travail de préparation
- Avantages
  - Rassembler les acteurs principaux (stakeholders)
  - Révélation des conflits
  - Qualité et efficacité
- Limites
  - Dépend du facilitateur et du groupe
  - Personnes importantes a rassembler sur plusieurs jours
  - Ne convient qu’à des projets importants et pas trop gros

Techniques d'analyse

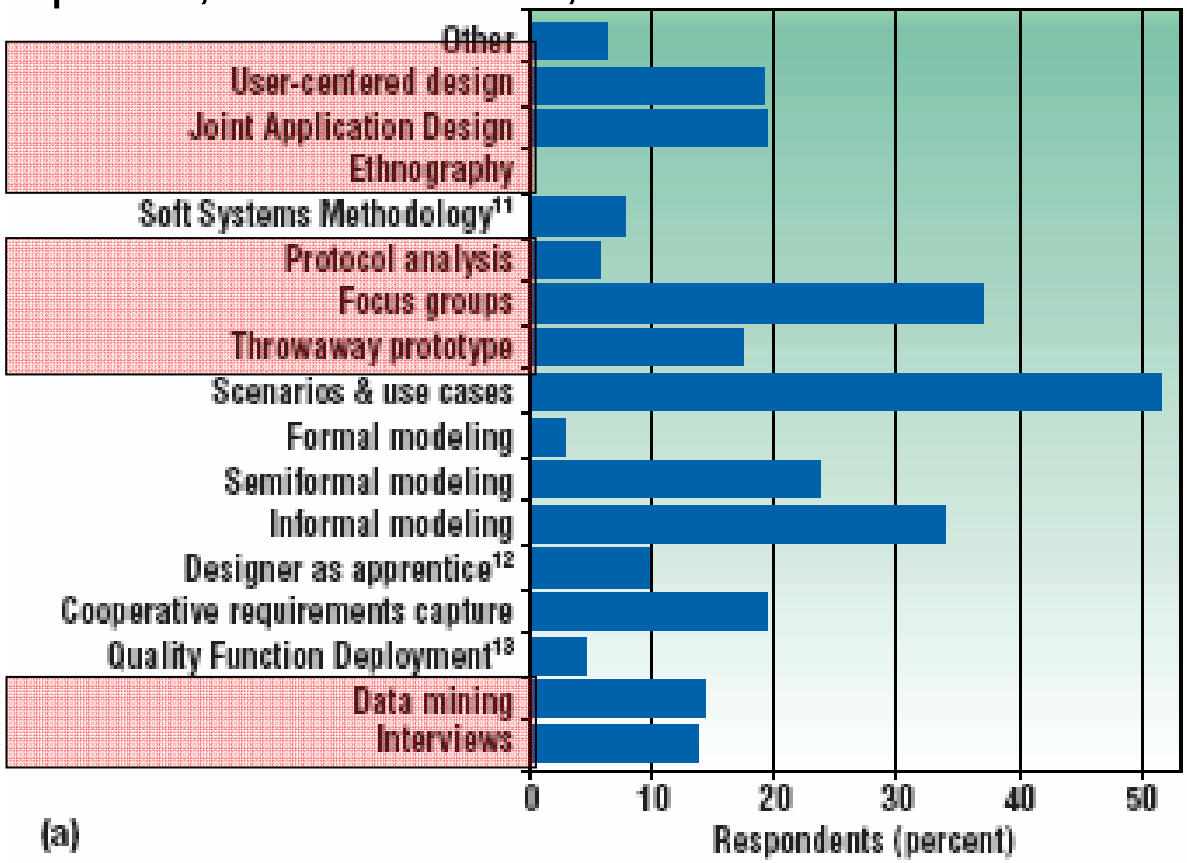
Prototypage

- Construire un prototype et le présenter aux acteurs importants
- Avantages
  - Très utile pour les logiciels avec de lourdes interactions avec l'utilisateur, des interfaces très dynamiques, des algorithmes à faire évoluer
  - Facilite les discussions par son aspect concret
  - Forte implication des utilisateurs
- Limites
  - Coût mal compris par les managers
  - Tentation de construire la suite à partir du prototype
  - Choix des fonctions à prototyper : une première phase d'identification des exigences est nécessaire (cf. méthodes précédentes)

Techniques d’analyse

Techniques utilisées

- Requirements engineering: The state of the practice
  Neill et Laplante, IEEE Software, 2003



Comparaison: Acquisition des données vs fonctions

	Data mining	Observation	Analyse tâche	Interview ns	Interview	Brainstorming	RAD	Prototype
Fonctions	+	+ +	+ +	+	+	+	+ +	+
Données	+	-	-	+	+	+	+ +	+

From Maiden and Rugg, 96

Techniques d'analyse

Comparaison : acquisition des connaissances

	Data mining	Observation	Analyse tâche	Interview ns	Interview	Brainstorming	RAD	Prototype
Non tacite	-	-	+ +	+ +	+ +	+ +	+ +	-
Tacite	+ +	+ +	-	-	-	-	-	+ +

From Maiden and Rugg, 96

Comparaison : préparation et temps d'acquisition

	Data mining	Observation	Analyse tâche	Interview ns	Interview	Brainstorming	RAD	Prototype
Temps de préparation	-	-	+ +	+	+ +	-	+ +	+ +
Temps d'acquisition	+ +	+	-	-	-	+ +	-	-

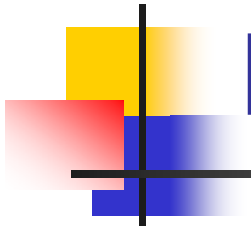
From Maiden and Rugg, 96

Techniques d'analyse

Comparaison: intérêt des acteurs

	Data mining	Observation	Analyse tâche	Interview ns	Interview	Brainstorming	RAD	Prototype
Intérêts des acteurs	+ +	-	-	+ +	+	+ +	+	+

From Maiden and Rugg, 96

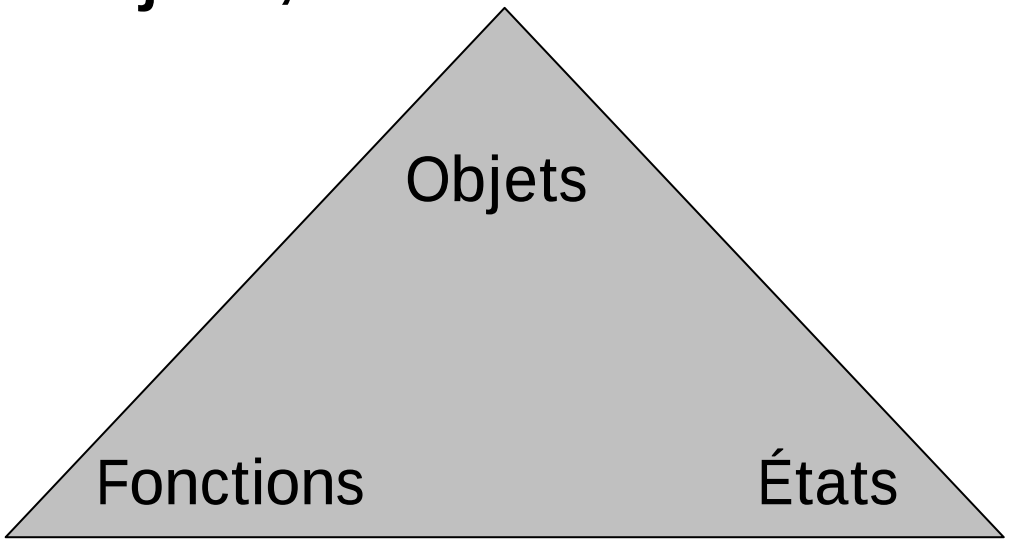


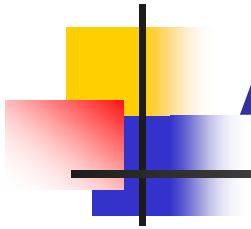
Plan

- Introduction
- Exigences
  - Fonctionnelles
  - Non-fonctionnelles
- Traçabilité des exigences
- Ingénierie des exigences
- Phase d'analyse et techniques
- **Spécification d'exigences**
- Conclusion

Méthodes d’analyse

- Méthodologies proposées : dirigées par les objets, fonctions ou états



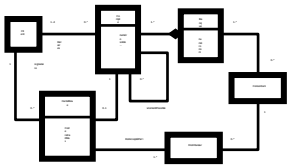
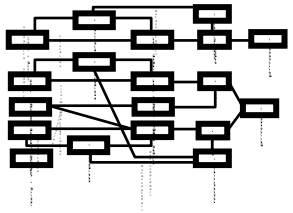
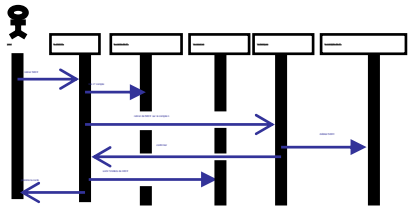
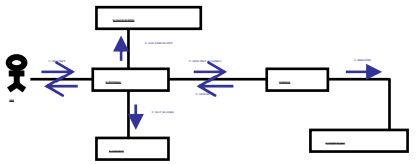
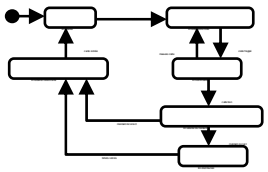


Analyse orientée objet

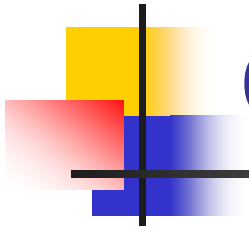
- Guidée par l'identification et l'étude des objets du système
- But
  - Identifier et définir les objets
    - Nom, attributs, méthodes
  - Identifier et définir les relations entre objets
    - Nom, attributs
  - Définir les séquences pour réaliser une fonction
  - Définir les états des objets

Analyse OO – diagrammes

- Beaucoup de diagrammes
  - De contexte
  - Cas d'utilisation
  - De classe
    - D'objet
    - De séquence
    - De collaboration
  - D'état

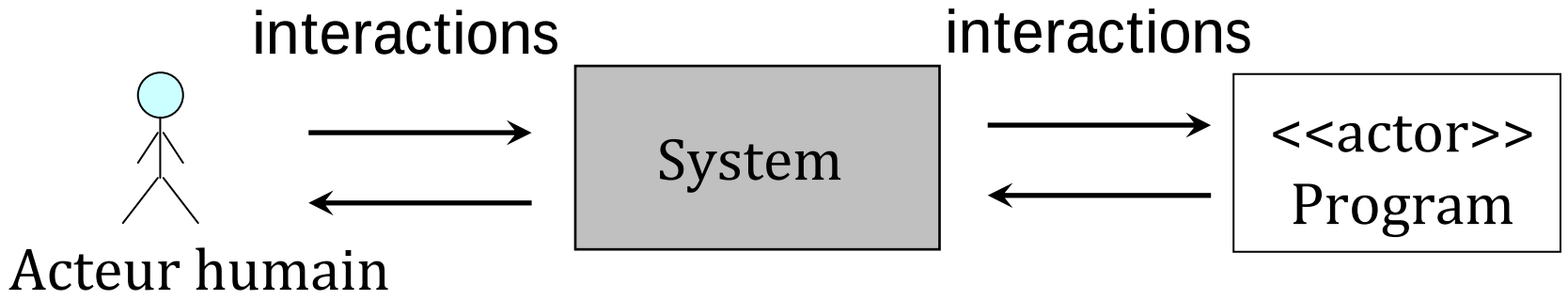
Analyse OO – diagramme de contexte



- Pour identifier les acteurs du système
  - Acteurs peuvent être humain ou logiciels
  - Style d'interactions et données échangée (à un haut niveau d'abstraction)

- Pour définir la frontière du système à développer
  - Environnement, limites, ...

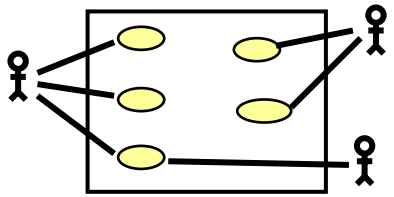
Analyse OO – diagramme de contexte



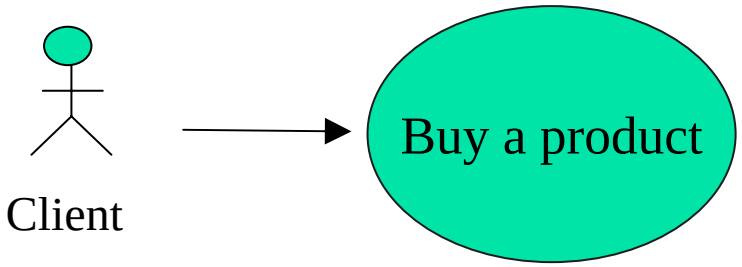
- Acteurs de premier niveau

Analyse OO – diagramme de cas d'utilisation

- Décrire les interactions entre les acteurs et le système
  - Contient plusieurs échanges
  - Correspond à une fonction visible de l'acteur
  - Permet à un acteur d'atteindre un but
  - Doit être utile
  - Regroupe une ensemble de scénarios



Analyse OO – diagramme de cas d'utilisation



Scenario : buy a product

Actors

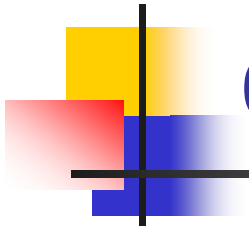
- Client

Description

The scenario begins with ...

Exceptions

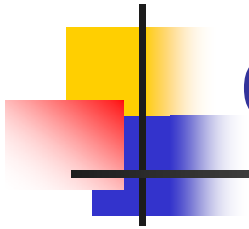
- Credit card is invalid
- Etc.



Analyse OO – diagramme de classe

- Identifier les objets et leurs relations
  - Identifier les classes possibles (pas de détails d'implantation)
  - Éliminer les classes non pertinentes (redondances, peu d'intérêt, vague, ...)
  - Identifier les relations
  - Éliminer les relations non pertinentes (redondances, niveau implantation, ...)
  - Identifier les attributs ...

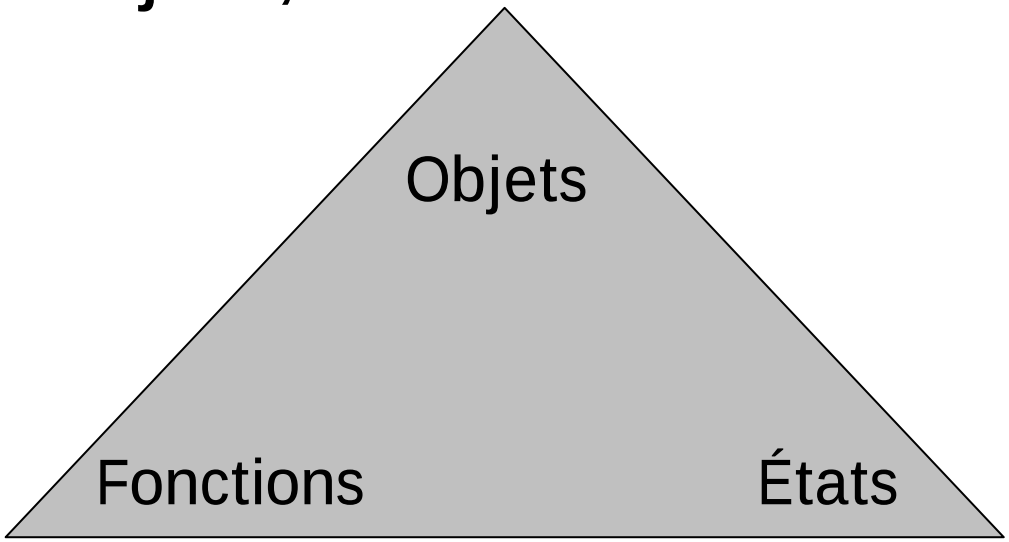
Analyse OO – conclusion



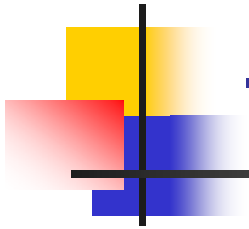
- Ne pas être naïf
  - Les entités relatives au domaine n'apparaîtront pas par magie
  - Niveau exigences et non étude du domaine
- Fonctions difficiles à découvrir
- Passage à la conception n'est pas "sans couture"
- Très populaire de nos jours

Méthodes d’analyse

- Méthodologies proposées : dirigées par les objets, fonctions ou états



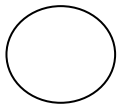
Diagrammes à flot de données - DFDs



- Ils spécifient
  - Les fonctions du système
  - Les flots de données
- Données :
  - Dans des dépôts (repository)
  - Entre les fonctions (flow)
  - En provenance de l'extérieur (sources de données)
  - En direction de l'extérieur (puits de données)
- Utilisation d'un langage graphique simple
  - Explication de leur succès

Diagrammes à flot de données - DFDs

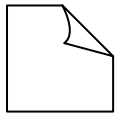
- Éléments de base
  - Fonctions: cercles
  - Flots de données: flèches
  - Données: deux lignes horizontales



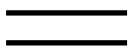
Symbole des fonctions

Symbole des entrées

Symbole des flots de données

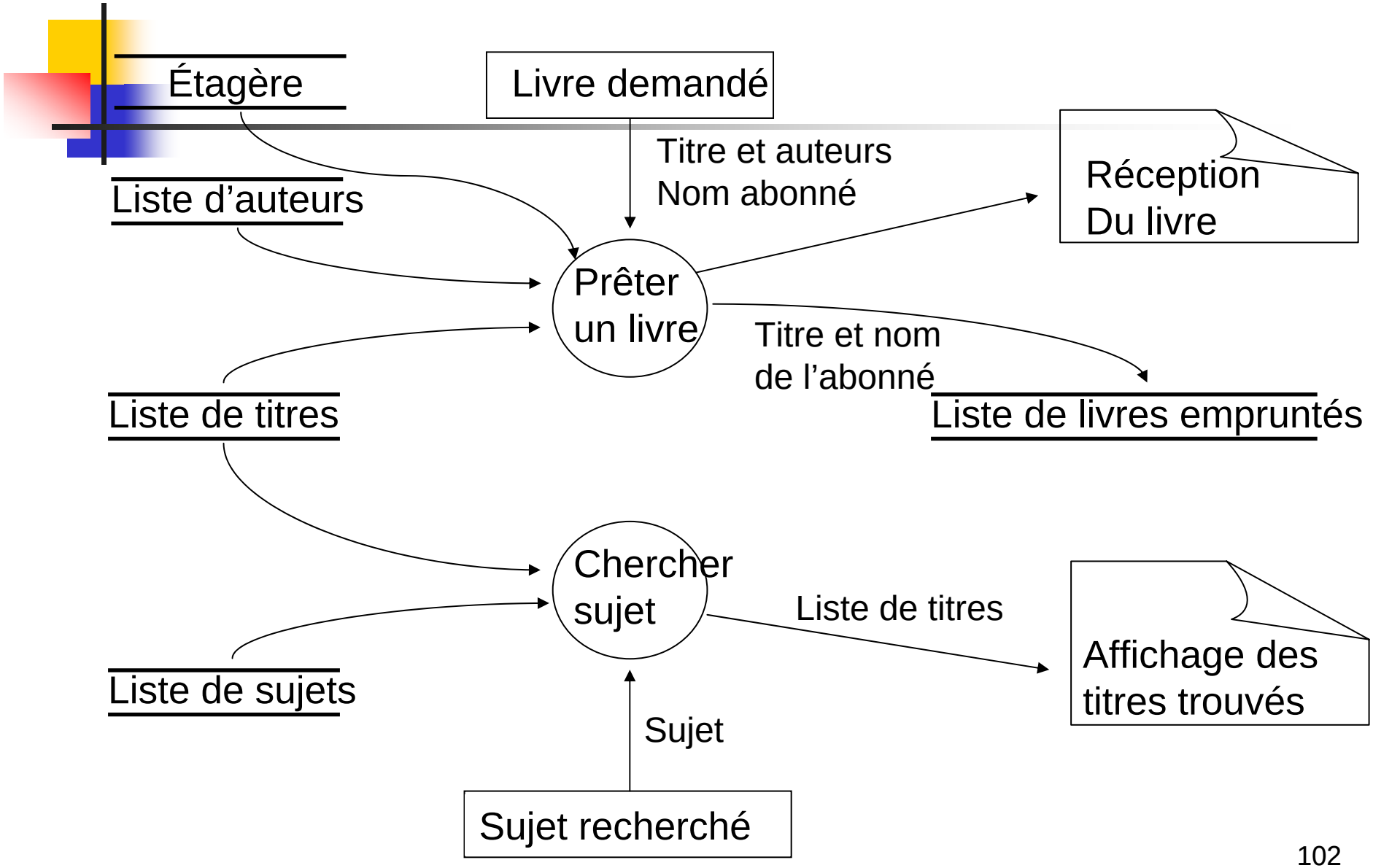


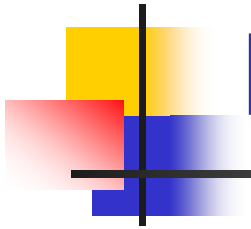
Symbole des sorties



Symbole des stockages de données

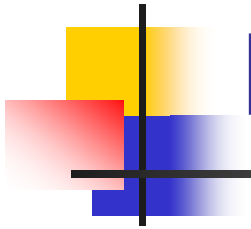
Example – from Ghezi, 2003





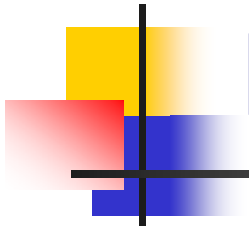
DFDs : sémantique

- Unicité des noms de fonctions et des données
  - Permet d'identifier les entités lors de l'analyse
- Pas d'ordonnancement des événements
  - Une flèche représente un flot de données
  - Il n'y a pas de notion d'ordre entre les flots de données
- Pas de points de décision
  - On ne peut pas exprimer des chemins alternatifs
- Attention aux détails
  - Les informations de bas niveau (y compris la gestion des erreurs) sont différées



DFDs: avantages et limites

- Avantages
  - Simples, compréhensibles, bien pour communiquer
  - Donnent une vision globale
  - Décomposition, abstraction, projection
- Limites
  - Peu de sémantique
    - Pas de machine abstraite possible
  - Pouvoir d'expression limité
    - Pas de point de décision
  - Gestion de beaucoup de diagrammes difficiles
    - Maintien de la cohérence

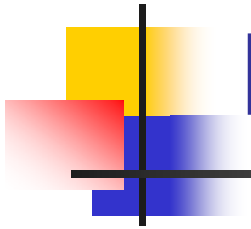


Dictionnaire des données

- Complément des DFDs
  - Noms de toutes les données et leurs alias
  - Description des données
  - Structure des données (primitives ou composées)
  - Relations entre données
  - Valeurs possibles des données
  - Flots de données générant ou utilisant les données

Exemple de dictionnaire de données

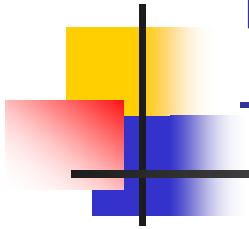
Nom	Généré par	Utilisé par	Description	Structure
Livre	Enregistrer un livre	- Prêter un livre - Réserver un livre -- Chercher un sujet	Toute forme d'ouvrage consultable à la biblio	- Titre - Auteurs - ISBN - Date
Abonné	Créer un abonné	- Prêter un livre - Réserver un livre -- Chercher un sujet - Retirer un livre	Toute personne inscrite à la biblio.	- Nom - Prénom - age - adresse



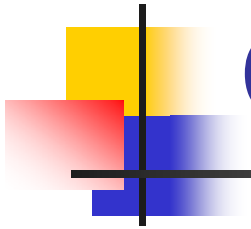
Plan

- Introduction
- Exigences
  - Fonctionnelles
  - Non-fonctionnelles
- Traçabilité des exigences
- Ingénierie des exigences
- Phase d'analyse et techniques
- Spécification d'exigences
- **Conclusion**

Erreurs dans les exigences – Van Lamsweerde



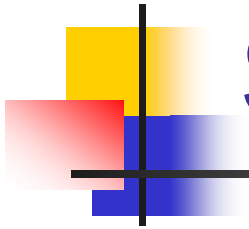
- Les plus fréquentes
  - 33% des erreurs logicielles
- Les plus persistantes
  - Souvent découvertes après la livraison
- Les plus coûteuses
  - detection/fix costs 5x more during design, 10x more during implementation, 20x more during testing, 200x more after delivery
- Les plus dangereuses



Quelques chiffres

- Principales sources d'erreur
  - Manque d'implication utilisateur 13%
  - Exigences incomplètes 13%
  - Changement d'exigences 9%
  - Attentes irréalistes 10%
  - Objectifs non-clairs 5%

  - www.standishgroup.com



Synthèse – Van Lamsweerde

- Quelques confusions
  - Les exigences ne sont pas des besoins
  - Les exigences ne sont pas les spécifications du logiciel
  - Les exigences expriment les problèmes, et non les solutions
  - L'ingénierie des exigences n'est pas une traduction de problèmes pré-existant
  - La composition des exigences ne signifie pas nécessairement la conjonction des exigences
  - "précis" ne signifie pas forcément "formel"
  - Un ensemble de notation n'est pas une méthode

L’ingénierie des exigences n’est pas populaire !

- 10 Top reasons (http://www.volere.co.uk/)
  - 10. We don't need requirements, we're using objects ...
  - 9. The users don't know what they want
  - 8. We already know what the users want
  - 7. Who cares what the users want?
  - 6. We don't have time to do requirements
  - 5. It's too hard to do requirements
  - 4. My boss frowns when I write requirements
  - 3. The problem is too complex to write requirements
  - 2. It's easier the change the system later ...
  - 1. We have already started writing code: we don't want to spoil it

Suddenly, a heated exchange took place between the king and the moat contractor.

© Gary Larson