Syllabus

REQB®

Professionnels Certifiés en
Ingénierie des Exigences

Niveau Fondation

Version 1.3 FR

31 octobre 2011

Historique des modifications

Version	Date	Commentaire
1.0	15 janvier 2008	Version 1.0 diffusée
1.1	29 mai 2008	Mise à jour : version 1.1
1.2	1er juillet 2008	Mise à jour : version 1.2
1.2 FR	15 décembre 2010	Version 1.2 en français
1.3	15 juin 2011	Mise à jour : version 1.3
1.3 FR	5 février 2012	Version 1.3 en français

Idée principale

Le thème central pour l'élaboration de ce syllabus était la croissance constante de la complexité du logiciel et de notre dépendance au logiciel. Ceci implique une exigence importante d'absence d'erreurs dans le logiciel. Le « Requirements Engineering Qualifications Board » (REQB) a donc décidé de créer des standards internationaux uniformes dans le domaine de l'Ingénierie des Exigences. Pour des standards comme les langages, c'est uniquement si vous les comprenez que vous pourrez travailler efficacement. Pour créer un tel langage standard dans ce domaine important qu'est l'Ingénierie des Exigences, des experts internationaux se sont réunis au sein du REQB et ont développé ce syllabus.

Remerciements

Le groupe de travail pour l'élaboration du niveau Fondation du ReqB (Edition 2011) :

Karolina Zmitrowicz (président), Alain Betro, Dorothée Blocks, Jérôme Khoualed, Eric Riou du Cosquer, Chris Hofstetter, Michał Figarski, Francine Lafontaine, Beata Karpińska, Folke Nilsson, Ingvar Nordström, Alain Ribault, Radosław Smilgin.

1 Sommaire

2 Introduction

Objectif de ce syllabus

Ce syllabus définit le niveau élémentaire (niveau Fondation) du programme de formation pour devenir un Professionnel Certifié REQB pour l'Ingénierie des Exigences (REQB Certified Professional for Requirements Engineering - CPRE). REQB a développé ce syllabus en collaboration avec le gasq - Global Association for Software Quality. Le sujet du périmètre REQB est tout type de produits IT pouvant inclure du matériel et des services ainsi que du logiciel avec leurs exigences validées, les exigences métier et la documentation associée.

Le syllabus a pour but de servir de base pour les organismes de formation qui sont à la recherche d'une accréditation en tant que formateur. Toutes les parties de ce syllabus doivent par conséquent être inclues dans les documents de formation. Cependant, le syllabus devrait également servir au stagiaire dans sa préparation à la certification. Toutes les parties énumérées ici sont donc utiles pour l'examen qui peut être passé après des cours accrédités ou en candidat libre.

Le syllabus fournit aussi des idées de durée recommandée pour chaque chapitre.

Examen

L'examen pour devenir un Professionnel Certifié REQB pour l'Ingénierie des Exigences est basé sur ce syllabus. Toutes les sections de ce syllabus peuvent ainsi être examinées. Les questions d'examen ne sont pas nécessairement divisées en différentes sections. Une question peut se référer à plusieurs sections.

Le format de l'examen est un Questionnaire à Choix Multiples.

Les examens peuvent être passés après avoir suivi des cours accrédités ou en candidat libre (sans cours préliminaire). Vous trouverez des informations détaillées concernant les durées des examens sur le site du Gasq (www.gasq.org), sur le site du REQB (www.reqb.org) ou sur le site du CFTL (www.cftl.fr)

Accréditation

Les organismes fournissant des cours Professionnels Certifiés REQB pour l'Ingénierie des Exigences niveau Foundation doivent être accrédités par le Gasq - Global Association for Software Quality. Les experts du Gasq revoient l'exactitude de la documentation des organismes de formation. Un cours accrédité est considéré comme conforme au syllabus. A la fin d'un tel cours, un examen officiel Professionnels Certifiés REQB pour l'Ingénierie des Exigences (examen CPRE) peut être réalisé par un organisme de certification indépendant (selon les règles de l'ISO 17024).

Les organismes de formation accrédités peuvent être identifiés par le logo officiel REQB – Organisme de Formation Accrédité.

Internationalité

Ce syllabus a été développé en collaboration par plusieurs experts internationaux. Le contenu de ce syllabus peut donc être considéré comme un standard international. Le syllabus permet ainsi de former et de faire passer des examens internationalement au même niveau.

Niveaux K

Les objectifs d'apprentissage de ce syllabus ont été classés en différents niveaux K de connaissance. Cela permet au candidat de reconnaitre le « niveau de connaissance » de chaque point.

Il y a 3 niveaux K dans ce syllabus :

- K1 – se rappeler, reconnaitre, rappeler
- K2 - comprendre, expliquer, donner des raisons, comparer, classifier, résumer
- K3 – appliquer dans un contexte spécifique

1 Fondamentaux (K2)	40 minutes

Objectifs d'Apprentissage (OA) pour le niveau Fondation de l'Ingénierie des Exigences

Les objectifs identifient ce que vous êtes capable de faire, en ayant suivi chaque module.

1.1 Exigence (K2)

OA-1.1.1 Rappeler la définition d'une exigence (K1)

OA-1.1.2 Expliquer la signification et le but des exigences (K2)

OA-1.1.3 Expliquer comment les exigences peuvent être classées (K2)

OA-1.1.4 Décrire les différents types d'exigences (K2)

OA-1.1.5 Expliquer quels sont les problèmes liés aux exigences (K2)

OA-1.1.6 Décrire quels sont les concepts importants en lien avec les exigences (K2)

OA-1.1.7 Expliquer la différence entre GE (Gestion des Exigences) et IE (Ingénierie des Exigences) (K2)

1.2 Standards et normes (K1)

OA-1.2.1 Rappeler les normes et standards importants relatifs à l'Ingénierie des Exigences (K1)

OA-1.2.2 Expliquer pourquoi l'Ingénierie des Exigences est importante (K2)

1.1 Exigences (K2)	20 minutes

Termes

Engagement, Criticité, Exigence Fonctionnelle, Exigence Non-Fonctionnelle, Exigence, Ingénierie des Exigences, Gestion des Exigences, Exigences Processus, Exigences Produit, Priorité, Solution, Validation, Vérification

1.1.1 Définition et classification (K2)

Définition de ce qu'on entend par le terme "Exigence" (K1)

Exigence [IEEE 610.12] :

- Condition ou aptitude requise par un utilisateur pour résoudre un problème ou atteindre un objectif
- Condition ou aptitude qui peut être rencontrée ou possédée par un système ou un composant d'un système pour satisfaire à un contrat, un standard, une spécification ou tout autre document composé formellement
- Document représentant une condition ou une aptitude comme décrit ci-dessus

Quels sont la signification et le but des exigences ? (K2)

- Fondement pour l'évaluation, la planification, l'exécution et le suivi de l'activité de projet
- Attentes client
- Composante d'accords, de commandes, de plans projet, …
- Définition des limites d'un système, du périmètre d'une livraison, de services contractuels

Classification des exigences (selon [Ebert05]) (K2)

Les exigences comprennent :

- des exigences processus
- des exigences produit.

Les exigences processus concernent les processus de développement et de livraison. Cela inclut, par exemple : les coûts, le marketing, le temps de traitement, les ventes et la distribution, l'organisation, la documentation.

Les exigences produit comprennent les exigences produit fonctionnelles et non fonctionnelles. Ces deux types d'exigences peuvent être considérés du point de vue de l'utilisateur (externe) ou du client et du point de vue de l'équipe de développement (interne). Il est important de rappeler que le client et le développeur peuvent être différents.

Les exigences fonctionnelles décrivent la fonction (comportement) d'un système.

Les exigences non-fonctionnelles décrivent les attributs qualité d'un système. Ils sont souvent appelés « attributs qualité ».

Les différences entre les exigences fonctionnelles et non-fonctionnelles peuvent être exprimées de la façon suivante :

- Les exigences fonctionnelles décrivent ce QUE le système fait
- Les exigences non-fonctionnelles décrivent COMMENT le système se comporte (pour faire quelque chose)

Exemples :

- Exigences produit fonctionnelles du point de vue de l'utilisateur et client : interface utilisateur, applications, services
- Exigences produit fonctionnelles d'un point de vue de l'équipe de développement : architecture, alimentation électrique, répartition de charge
- Exigences produit non-fonctionnelles du point de vue de l'utilisateur et du client : fiabilité, performance, utilisabilité (facilité d'emploi)
- Exigences produit non-fonctionnelles du point de vue de l'équipe de développement : testabilité, services offerts, outils

Types de base d'exigences (K1) :

- Exigences client
  - Désirs clients, besoins et attentes (exigences métier de haut niveau)
  - Limitations métier
- Exigences solutions / systèmes
  - Spécification des besoins client (spécifications détaillées des exigences métier de haut niveau)
- Exigences produit / composant
  - Fonctions et caractéristiques de la solution
  - Les bases pour l'analyse et la conception détaillée (exemple : cas d'utilisation)

1.1.2 Problèmes avec les exigences (K2)

Les principaux problèmes avec les exigences sont :

- Objectifs imprécis
- Problèmes de communication
- Barrières de la langue
- Barrières de connaissance
- Formulation vague
- Formulations trop formelles
- Instabilité des exigences
- Mauvaise qualité des exigences (incluant des exigences ambigües, trop spécifiées, pas claires, impossibles, contradictoires)
- Placage doré (trop de description d'une exigence qui implique que l'exigence réelle est couverte par des descriptions inutiles)
- Implication insuffisante des utilisateurs
- Classes utilisateur oubliées (impliquant des parties prenantes oubliées)
- Planning inadapté
- Spécification minimale

1.1.3 Critères qualité des exigences (K2)

Les critères qualité des exigences (d'après Wiegers05) sont les suivantes : (K2)

1. Chaque exigence doit être
    - Correcte : l'exigence doit décrire de façon précise la fonctionnalité à fournir. La base de référence pour évaluer la précision est la source de l'exigence (par exemple, les clients ou une exigence système de haut niveau)
    - Faisable : il doit être possible d'implémenter l'exigence à partir des capacités et limitations connues d'un système et de son environnement
    - Utile : l'exigence doit documenter ce dont le client (ou les autres parties prenantes) a réellement besoin et ce qui est exigé pour réaliser une exigence externe ou une interface ou un standard spécifié
    - Priorisée : l'exigence devrait avoir une priorité allouée indiquant son niveau d'importance pour une release donnée d'un produit
    - Non ambiguë : l'exigence devrait être interprétée de façon unique. Des personnes différentes prenant connaissance d'une exigence doivent avoir la même interprétation et la même compréhension de l'exigence
    - Vérifiable : il devrait être possible de vérifier si l'exigence est implémentée correctement
    - Unique : ne doit pas contenir de multiples exigences ; nécessite une granularité suffisante pour spécifier une exigence de façon unique
    - Indépendante de la conception : décrit le « QUOI », pas le « COMMENT »
2. La spécification des exigences doit être :
    - Complète : aucune exigence ni information nécessaire ne devrait être oubliée dans la spécification des exigences. La complétude est aussi exprimée comme une caractéristique désirée pour une exigence individuelle et un niveau de détail

- Cohérente : l'exigence ne peut rentrer en conflit avec d'autres exigences ou avec des exigences de haut niveau (système ou métier)
- Modifiable : la spécification doit permettre d'introduire des changements dans les exigences. L'historique des changements effectués sur chaque exigence doit être conservé
- Traçable : il doit être possible de tracer chaque exigence vers sa source (par exemple, une exigence système de plus haut niveau, un cas d'utilisation ou une assertion client) et vers les artefacts d'implémentation correspondants (par exemple, des éléments de conception, le code source et les cas de tests)

1.1.4 Solution (K1)

Solution (K1)

Une solution est l'implémentation d'une exigence.

La solution peut être un système logiciel, l'amélioration d'un processus, etc.

1.1.5 Engagement (K1)

Engagement (K1)

L'engagement est le degré d'obligation de respect des exigences.

L'engagement est défini via des mots-clés alloués aux exigences de haut niveau :

- Le système devrait …

Les mots-clés peuvent inclure : « doit », « sera », « devrait », « serait », « pourrait »

Les mots-clés «doit», «devrait», «pourrait» sont liés aux exigences métier et utilisateur avant l'engagement. Après la phase de contractualisation et de référencement des exigences, les mots-clés devraient déterminer strictement le degré d'obligation :

- Le système fera …

Le niveau d'obligation d'une exigence peut être exprimé en utilisant la notation Moscow (Must Have, Should Have, COuld have, Would have).

Une fois que le fournisseur de la solution et le client se sont mis d'accord contractuellement, l'engagement sur les exigences est obtenu des participants au projet.

Les exigences doivent évoluer tout au long du projet. Comme les exigences évoluent, obtenir l'engagement sur les exigences des participants du projet assure que les participants du projet

sont d'accord avec les exigences courantes et approuvées et avec les changements résultant dans les plans projets, les activités et éléments produits.

1.1.6 Responsabilités légales et fautes (K1)

Responsabilité légale (K1)

Il y a des responsabilités légales liées à la qualité du logiciel. Les responsabilités légales sont souvent en relation avec des exigence spécifiques (i.e. exigence sur l'environnement pour une centrale nucléaire, un avion) qui doivent être satisfaites dans le produit livré. Les responsabilités doivent être en relation avec les défauts dans le produit.

Faute (défaut) (K1)

Une faille dans un composant ou dans un système qui peut engendrer, pour ce composant ou ce système, une défaillance dans l'exécution d'une fonction exigée, par exemple une instruction ou une définition de donnée incorrecte.

Un défaut, si rencontré pendant une exécution, peut causer une défaillance d'un composant ou d'un système [ISTQB].

1.1.7 Priorité et criticité des exigences (K2)

Priorité des exigences (K1)

La priorité est l'évaluation de l'importance / urgence d'une exigence.

D'après le document [SWEBOK], en général, plus la priorité est élevée, plus l'exigence est essentielle pour répondre aux objectifs généraux du logiciel.

Souvent classées sur une échelle de points fixes comme obligatoire, hautement désiré, désiré ou optionnel, la priorité doit souvent être confrontée au coût du développement et de l'implémentation.

Exemples de priorité d'exigences :

- Haute
- Moyenne
- Basse

Criticité des exigences (K1)

Évaluation du risque d'une exigence en évaluant le dommage en cas de non-respect d'une exigence

La criticité est exprimée sous forme de niveaux : plus le niveau est élevé, plus critiques sont les conséquences dans le cas d'une défaillance fonctionnelle.

1.1.8 Vérification et validation (K1)

La validation est un processus de confirmation que la spécification d'une phase ou le système complet satisfait les exigences client.

La validation est généralement réalisée avec le support des utilisateurs dont le but est de confirmer que les exigences ou la spécification des exigences ont décrit ce dont le client a besoin.

D'après CMMi, les activités de validation démontrent que le produit ou que le composant d'un produit répond à son usage prévu quand il est placé dans l'environnement cible. Donc, nous savons que « nous avons construit le bon produit ». Les clients identifient souvent les descriptions produit ou les exigences d'une manière insuffisante et la validation aide à la compréhension de ce qui est nécessaire (en utilisant des outils comme des scenarii, des cas d'utilisation ,du prototypage, etc.).

La vérification est la comparaison d'un élément intermédiaire produit avec ses spécifications. Cela détermine ainsi si le logiciel a été développé correctement et si les spécifications fixées lors de la phase précédente sont remplies.

D'après CMMi, la vérification fournit des points de contrôle au cours desquels les livrables sélectionnés ou les éléments intermédiaires produits sont vérifiés pour confirmer qu'ils répondent à leurs exigences ; ils permettent de confirmer tôt et en cours de réalisation que le produit est construit correctement.

Les techniques les plus connues pour la vérification et la validation sont les revues, les audits, les checklists et le test.

La différence entre la validation et la vérification peut être exprimée de la façon suivante :

- Vérification : avons-nous créé le produit correctement ?
- Validation : avons-nous créé le bon produit ?

1.1.9 Ingénierie des Exigences, Gestion des Exigences et Développement des Exigences (K2)

Délimitation entre gestion, développement et ingénierie des exigences (K2)

L'Ingénierie des Exigences est une sous-discipline de l'Ingénierie Logicielle (Ingénierie Système), se focalisant sur l'identification et la gestion des exigences des systèmes matériels et logiciels. L'Ingénierie des Exigences comprend la Gestion des Exigences et le Développement des Exigences.

La discipline d'Ingénierie des Exigences implique les différents sous-processus : élucidation des exigences, analyse et négociation (incluant la priorisation des exigences), les spécifications, la modélisation système, la validation des exigences.

Ces sous-processus peuvent se recouvrir, par exemple, la modélisation système peut être à la fois une partie de l'analyse et de la spécification, et dans certains cas de l'élucidation.

La gestion des exigences constitue un cadre de travail pour l'Ingénierie des Exigences et les interfaces avec les autres processus comme la gestion de projet, la gestion de configuration et la gestion de la qualité.

Le but de la gestion des exigences est de gérer les exigences des produits de projets et les composants, pour assurer l'alignement entre ces exigences, les plans projet et les produits de travail tout au long du cycle de vie des projets de produits (cycle de développement et cycle de maintenance).

La Gestion des Exigences (GE) inclut les processus pour le management global des exigences. C'est un processus continuel de documentation, d'analyse, de traçabilité, de priorisation, de communication et d'accord sur les exigences et sur la gestion des changements des exigences.

Le Développement des Exigences (DE) est un ensemble d'activités, de tâches, de techniques et d'outils pour identifier, analyser et valider les exigences, qui inclut le processus de transformation des besoins en exigences.

Le but du développement des exigences est d'élucider, d'analyser, d'établir et de valider les exigences client, produit et composants de produits.

1.2 Exigences (K2)	20 minutes

Pour passer l'examen, il n'est pas nécessaire de connaître le contenu de toutes les normes. Il est, néanmoins, important (K1) de connaître quelles normes sont importantes pour l'Ingénierie des Exigences.

1.2.1 Standards (K1)

ISO 9000 :

Exigences d'un système de management de la qualité :

Concepts et fondements définis d'un SMQ

Indépendant d'un domaine ou de l'industrie

ISO 9126 (remplacé par l'ISO/IEC 25000) :

Définit un modèle qualité en six catégories : fonctionnalité, fiabilité, utilisabilité (facilité d'utilisation), efficacité, maintenabilité, portabilité

IEEE 610 :

Glossaire standard de la terminologie de l'ingénierie logicielle

IEEE 830 :

Pratique recommandée pour les spécifications d'exigences logicielles

IEEE 1233 :

Guide pour la Spécification d'Exigences de Système

IEEE 1362:

Guide pour la technologie de l'information – Définition Système

SWEBOK – Guide pour le « Software Engineering Body of Knowledge (connu comme un rapport technique ISO 19759)

SWEBOX décrit généralement les principes acceptés en Ingénierie Logicielle. Ses 10 parties résument les concepts fondamentaux et incluent une liste de référence pointant sur des informations détaillées.

1.2.2 Normes processus (K1)

ISO 12207 :

Standard pour le processus du cycle de vie logiciel

ISO 15288 :

Processus de cycle de vie système

ISO 15504:

Software Process Improvement and Capability Determination (SPICE)

1.2.3 Les raisons de négligence de l'Ingénierie des Exigences (K2)

L'Ingénierie des Exigences est d'une importance vitale. Et pourtant, elle est maintes fois négligée.

Les raisons de la négligence d'Ingénierie des Exigences peuvent être les suivantes (K2) :

- Pression importante des délais
- Orientation exclusive vers des résultats rapides
- Fixation exclusive des coûts
- Mauvaises interprétations (beaucoup de choses sont considérées comme connues) et manque de compréhension de l'importance de l'Ingénierie des Exigences pour le succès d'un projet

Conséquences possibles causées par la négligence de l'Ingénierie des Exigences (K2) :

- Les exigences deviennent imprécises
- Les exigences sont ambigües
- Les exigences sont contradictoires
- Les exigences changent souvent pendant le développement logiciel
- Les exigences ne remplissent pas les critères
- Les exigences peuvent être interprétées différemment
- Des exigences manquantes

2 Modèles de processus et Processus d'Ingénierie des Exigences (K2)	60 minutes

Objectifs d'Apprentissage (OA) pour le niveau Fondation de l'Ingénierie des Exigences

Les objectifs identifient ce que vous êtes capable de faire, en ayant suivi chaque module.

2.1 Modèles de processus (K2)

OA-2.1.1 Décrire les différents modèles de processus (K2)

OA-2.1.2 Décrire quelles sont les différences entre modèles de processus (K2)

2.2 Processus d'Ingénierie des Exigences (K2)

OA-2.2.1 Décrire les caractéristiques du processus d'Ingénierie des Exigences (K2)

OA-2.2.2 Expliquer les phases du processus d'Ingénierie des Exigences (K2)

2.1 Modèles de processus (K2)	30 minutes

Termes

Extreme Programming, Modèles de processus, Cycle de vie produit, Rational Unified Process, Cycle en V

2.1.1 Les modèles de processus (K2)

Les modèles de processus sont des processus de description des processus de développement, indépendants de méthodes.

Les rôles, activités, phases et documents sont ainsi pris en compte

Un modèle de processus logiciel fournit un format standard pour la planification, l'organisation et le déroulement d'un projet logiciel.

Cycle de Vie Produit (PLC – Product Life Cycle) (K2)

Définit les différentes phases du développement d'un produit.

Les phases élémentaires sont :

1. Planification
2. Développement
3. Maintenance
4. Fin de vie

La phase de planification inclut : la vision, la stratégie, le business plan et l'analyse de bénéfices.

La phase de développement inclut : la spécification, le maquettage et l'implémentation. La phase de développement est souvent divisée en quatre phases :

- Analyse
- Conception
- Implémentation
- Test

2.1.2 Modèle de cycle en V général (K2)

Les étapes de développement sont :

Définition des exigences, détermination des exigences (spécifications des exigences de haut-niveau)

Maquettage fonctionnel du système, analyses systèmes (spécifications fonctionnelles)

Maquettage technique du système, maquettage de l'architecture (conception logicielle)

Spécification des composants

Implémentation

Ce modèle a une forme de cycle en V et chaque niveau a un niveau de test correspondant.

Analyse/définition des exigences	→	Tests d'Acceptation
Conception fonctionnelle système	→	Tests Système
Conception technique	→	Tests d'intégration
Conception de composant	→	Tests unitaires
Implémentation		

Pour les organismes de formation : représentation graphique du modèle de cycle en V général ; description approfondie du modèle de cycle en V général

2.1.3 Rational Unified Process (RUP©) (K2)

Rational Unified Process est un modèle de processus défini par IBM Rational ©.

C'est un modèle itératif (i.e. le processus est répété jusqu'à ce que tous les scenarii, les risques et les changements soient adressés). Il a 9 disciplines (6 disciplines d'ingénierie et 3 disciplines de support) incluant la discipline « Exigences ». Chaque discipline est couverte par 4 phases consécutives du cycle de vie projet : création, élaboration, construction et transition.

Pour les organismes de formation : approfondissement de RUP© avec une représentation graphique, étude approfondie de la discipline relative aux exigences

2.1.4 Approches Agiles

Gestion des exigences

Dans les environnements agiles, les exigences sont communiquées et tracées via des mécanismes comme les “backlogs” produit, les « user stories » et les maquettes écran. Les engagements sur les exigences sont pris soit collectivement par l’équipe soit par un responsable d’équipe habilité. Les affectations de tâches sont ajustées régulièrement (e.g. quotidiennement, de façon hebdomadaire) sur la base de l’avancement et comme une compréhension accrue des exigences et de la solution qui émerge. La traçabilité et la cohérence entre les exigences et les éléments produits sont réalisés via des mécanismes déjà mentionnés au début d’une itération ou à la fin d’une itération comme les « rétrospectives » ou les « démonstrations ».

Développement des exigences

Dans les environnements agiles, les besoins client et les idées sont élucidés, élaborés, analysés et validés de façon itérative. Les exigences sont documentées sous différentes formes comme les « user stories », les scenarii, les cas d’utilisation, les « backlogs » produit et les résultats des itérations (code dans le cas de logiciel). Le choix des exigences à traiter dans une itération est fait suite à une analyse de risques et en fixant des priorités sur ce qui reste dans le backlog produit. Le niveau de détail des exigences (et des autres artéfacts) à documenter est fixé pour le besoin de coordination (entre les membres de l’équipe, les équipes et les prochaines itérations) et le risque de perdre ce qui a été appris. Lorsque le client est dans l’équipe, il peut tout de même y avoir un besoin pour un client différent et une documentation produit autorise l’exploration de multiples solutions. Quand la solution émerge, les responsabilités pour les exigences dérivées sont allouées aux équipes adéquates.

2.1.5 Extreme Programming (K2)

Extreme Programming est une méthodologie de développement logiciel définie par « Kent Beck et al » en réponse au changement des exigences client. La gestion de projet (par exemple SCRUM, § 2.1.6) définit comment et quand les changements des exigences client sont implémentées dans la solution logicielle (par exemple, dans quel Sprint). Elle préconise des releases fréquentes du logiciel au client avec des cycles de développement courts (unité de temps), dont le but est d’améliorer la productivité et de fournir des points de contrôle pour vérifier où les nouvelles exigences peuvent être prises en compte.

Quelques caractéristiques de l’Extreme Programming comprennent :

- La programmation par les pairs
- Revue de code intensive
- Tests unitaires du code
- Éviter de programmer des fonctionnalités tant qu’elles ne sont pas absolument nécessaires

- Structure de gestion horizontale (pas de structure hiérarchique complexe au sein de l'équipe. Cela peut être le mieux observé dans les équipes Scrum – l'équipe Scrum est « autogérée », il n'y a pas de rôle comme : rôle principal, chef de projet, etc.)
- Simplicité et clarté dans le code
- S'attendre à des changements dans les exigences client malgré le temps qui passe et le problème qui est mieux compris
- Communication fréquente avec le client et entre les développeurs
- Renoncement complet à la détermination des exigences (pas de « phase » d'exigences séparée avant que le développement débute : le développement, le raffinement et la découverte des exigences font partie du développement logiciel (codage))

2.1.6 Scrum (K2)

Scrum est un cadre de travail agile. Scrum a été originellement formalisé pour les projets de développement logiciel.

Scrum contient un ensemble de pratiques et de rôles prédéfinis. Les rôles principaux dans Scrum sont :

- Scrum Master (responsable de la maintenance des processus)
- Product Owner (représente les parties prenantes et les aspects métier)
- L'équipe (un groupe fonctionnel transversal effectuant l'analyse, la conception, l'implémentation, le test, etc.)

Une des caractéristiques principales de Scrum est le découpage du développement en « sprint » (typiquement d'une durée de 2 ou 4 semaines). Pendant chaque sprint, l'équipe crée ce qui s'appelle un incrément produit potentiellement livrable.

Scrum permet la gestion des exigences via les « backlogs ». Il y a deux types de backlogs :

- Le backlog « produit » : une liste de haut niveau maintenu tout au long du projet. Il regroupe toutes les exigences sous forme d'une description générale des fonctionnalités potentielles, priorisée via les priorités business. Le backlog produit est la propriété du « Product Owner ».
- Le backlog de « sprint » : la liste du travail à réaliser par l'équipe durant le prochain sprint. Les fonctionnalités sont découpées en tâches, qui, suivant les bonnes pratiques, doivent être comprises entre 4 et 6 heures de travail. Le backlog de « sprint » est la propriété de l'équipe.

Les caractéristiques principales de l'approche Scrum sont :

- A la fin de chaque sprint, le logiciel fonctionnel doit être livré – en pratique, les équipes commence à travailler sur l'analyse des exigences et continue pendant le sprint courant. Pendant le découpage des tâches, les exigences sont clarifiées.

- Les membres de l'équipe sont censés interagir entre eux, et l'implication régulière du client est un concept clé dans le développement agile. Les retours client peuvent être fournis lors des sessions de démonstration d'une nouvelle version logicielle à la fin d'un sprint, ou via les tests d'acceptation utilisateur.
- Les exigences évoluent via les retours client – montrer du logiciel qui fonctionne aide les clients à clarifier leurs exigences.
- Les exigences ne sont pas complètement spécifiées avant le démarrage du projet, mais les exigences sélectionnées pour un sprint sont spécifiées au début du sprint.
- Les fonctionnalités sont censées être repriorisées pendant que le projet avance.

L'impact principal de Scrum sur l'Ingénierie des Exigences est que les spécifications des exigences ne sont pas complètes ni validées avant le début du développement du projet.

Le « product owner » et l'équipe se mettent d'accord sur les fonctionnalités, provenant du backlog produit, à inclure dans le sprint à venir, en se basant sur les priorités business et l'effort de travail requis. Les exigences utilisateur sont formulées par le Product Owner sous forme de « user stories » qui contiennent des informations sur « Qui, Quoi, Pourquoi » mais pas sur le « Comment » de l'exigence. Au début du sprint, les fonctionnalités sélectionnées sont découpées sous forme de tâches dans le backlog de sprint, et ensuite développées.

L'implication des product owners, par exemple en leur présentant les fonctions implémentées du logiciel, permet de clarifier les exigences pour l'équipe et pour les product owners eux-mêmes.

Pour les organismes de formation : donner des exemples de user stories et des éléments correspondants de backlog produit et de backlog de sprint.

Pour les organismes de formation : expliquer aussi deux autres modèles agiles incluant Crystal.

2.1.7 Modèle de maturité (K2)

Les niveaux de maturité servent pour l'identification et l'amélioration de la maturité de processus (évaluation de processus et amélioration de processus)

ISO/IEC 15504 (SPICE – Software Process Improvement and Capability dEtermination) (K1)

ISO/IEC 15504 (SPICE – Software Process Improvement and Capability dEtermination) est un ensemble de standards techniques pour le processus de développement logiciel et les fonctions de gestion métier correspondant.

ISO/IEC 15504 peut être utilisé comme un moyen d'améliorer le processus et/ou la détermination des aptitudes (par exemple, l'évaluation des aptitudes des processus d'un vendeur).

SPICE définit des processus classés en cinq catégories de processus :

- Client – fournisseur
- Ingénierie
- Support
- Gestion
- Organisation

Pour chaque catégorie de processus ci-dessus, ISO/IEC 15504 définit un niveau d'aptitude :

5 Processus optimisé

4 Processus prédictible

3 Processus établi

2 Processus géré

1 Processus réalisé

0 Processus incomplet

Pour les organismes de formation : approfondissement en utilisant l'exemple de l'ISO 15504/SPICE, avec une description des exigences typiques pour l'Ingénierie des Exigences

Capability Maturity Model Integrated (CMMI)

Le CMMI définit des niveaux de maturité pour le développement, les services et l'acquisition :

1 Initial (chaotique, ad hoc, héros individuels) – le point de départ pour l'utilisation de nouveaux processus

2 Géré - le processus est géré par rapport à des métriques validées

3 Défini – le processus est défini / confirmé comme un processus métier standard et décomposé en niveaux 0, 1 et 2

4 Géré quantitativement

5 Optimisé – la gestion du processus inclut des améliorations / optimisations délibérées du processus

2.2 Processus d'Ingénierie des Exigences (K2)	30 minutes

Termes

Processus orienté client, perspective, Ingénierie des Exigences

2.2.1 Définition du processus d'Ingénierie des Exigences (K2)

L'Ingénierie des Exigences est une discipline qui inclut les processus nécessaires pour l'identification, la structuration et la gestion des Exigences. L'Ingénierie des Exigences contient les sous-processus suivants :

- Identification des exigences
- Analyse des exigences
- Spécification des exigences
- Accord sur les exigences
- Changements des exigences
- Validation et Assurance Qualité

2.2.2 Influences de l'Ingénierie des Exigences

Certains facteurs peuvent avoir une influence négative sur l'Ingénierie des Exigences :

- D'un point de vue interne (au sein de l'organisation du vendeur de logiciels) :
  - Manque de connaissance du domaine utilisateur
  - Approche / méthodologie inefficace de l'Ingénierie des Exigences
  - Expérience et compétence insuffisante du personnel
- D'un point de vue externe (en dehors de l'organisation du vendeur de logiciels) :
  - Manque de communication
  - Objectifs business non clairs ou changeant donnant des exigences instables
  - Pas de connaissance du processus de développement logiciel
  - Pas d'implication des utilisateurs et/ou des parties prenantes métier

Il y a différentes parties prenantes avec différents points de vue sur le processus d'Ingénierie des Exigences. En général, le processus peut être vu d'un point de vue du client et d'un point de vue du fournisseur (vendeur).

Exemple :

> D'un point de vue client, l'aspect le plus important de l'Ingénierie des Exigences peut être : l'interface utilisateur, les applications et les services. D'un point de vue du fournisseur, d'autres aspects pouvant être considérés sont : l'architecture, la répartition de la charge, etc.

Les méthodes de processus d'Ingénierie des Exigences avec le client au centre sont :

- Analyse et conception orientées utilisateur
- Approche par prototypage
- Utilisation de démonstrations comme moyen de validation des incréments d'un système

3 Gestion de projet et du risque (K2)	60 minutes

Objectifs d'Apprentissage (OA) pour le niveau Fondation de l'Ingénierie des Exigences

Les objectifs identifient ce que vous êtes capable de faire, en ayant suivi chaque module.

3.1 Gestion de projet (K2)

OA-3.1.1 Expliquer pourquoi l'Ingénierie des Exigences est importante dans les projets (K2)

OA-3.1.2 Rappeler les erreurs qui peuvent survenir en Ingénierie des Exigences (K2)

3.2 Gestion de risque (K1)

OA-3.2.1 Reconnaître les risques relatifs à l'Ingénierie des Exigences (K2)

3.1 Gestion de projet (K2)	30 minutes

Termes

Conception projet, négociations de contrat, définition de projet, exécution de projet

3.1.1 Nécessité de l'Ingénierie des Exigences dans les projets (K2)

Certaines des raisons principales de l'échec de projets sont liées aux exigences. La négligence vis-à-vis de l'Ingénierie des Exigences peut entraîner des exigences imprécises, contradictoires, ne répondant pas aux critères ou ne satisfaisant pas les besoins des parties prenantes. C'est pourquoi, une Ingénierie des Exigences structurée et précautionneuse est nécessaire à chaque projet.

L'Ingénierie des Exigences devrait contribuer aux domaines suivants :

- Conception projet
  - Identification des besoins et des attentes client concernant la solution du problème
  - Définition des exigences de haut niveau
- Négociations de contrat
  - Évaluation des exigences client
  - Détermination du périmètre initial et des ressources requises pour un projet
  - Détermination du coût de développement (implémentation des exigences)
  - Accord sur les priorités des exigences
- Définition du projet
  - Définition des rôles, des tâches, des activités et des processus additionnels (exemple : gestion du changement)
  - Conception détaillée métier de la solution
  - Contribution à la conception de l'architecture
  - Contribution aux livrables du processus de test
- Exécution projet
  - Fourniture d'une base pour le développement des exigences et pour la vérification et la validation des exigences (test)
  - Forcer la revue des plans et leur ajustement au périmètre courant de la solution en cas de changement des exigences

3.1.2 Quelles erreurs peuvent survenir dans l'ingénierie des Exigences ? (K2)

Les erreurs les plus communes sont les suivantes :

- Exigences non claires
- Changement des exigences (si les changements résultent d'objectifs projet non clairs ou d'imprécisions sur le domaine métier du client ; les changements des exigences ne sont pas perçus comme une erreur dans les approches itératives et agiles)
- Instabilité du produit et des bases de conception pour les commandes sous-traitées
- Manque de clarté dans les responsabilités (aussi bien du côté client que du côté vendeur)
- Écart entre les attentes client et les contenus projet
- Implication client insuffisante
- Définition projet avec des jalons qui ne peuvent pas être atteints
- Estimation imprécise des dépenses
- Estimation imprécise des impacts des changements des exigences sur les autres parties du produit en cours de développement
- Manque de traçabilité

3.2 Gestion de risques (K2)	30 minutes

Termes

Analyse des modes de défaillance et de leurs effets, risque produit, risque projet, risque, gestion de risque, plan de gestion de risque

3.2.1 Nécessité de la gestion de risque (K2)

Une gestion de risque efficace est une clé pour diminuer les risques produit et projet. L'identification des risques, leur analyse menée de façon appropriée et la planification de réactions adéquates sur les risques minimisent les chances d'occurrence d'un risque et les conséquences si un risque survient.

3.2.2 Risque (K2)

Risque (K1)

Un risque est défini comme l'effet d'une incertitude sur des objectifs, positif ou négatif [ISO 31000].

L'autre définition décrit le risque comme la chance qu'un évènement, un danger, une menace ou une situation puisse survenir et résulte en des conséquences indésirables ou un problème potentiel. Le niveau d'un risque est déterminé par la probabilité d'un événement indésirable de se produire et l'impact (le préjudice résultant de cet événement) [ISTQB].

Type de risque (K2)

Il y a deux types de risques :

- Risque projet
- Risque produit

Risques projet (K2)

Les risques projet sont les risques qui empêchent le projet d'atteindre ses objectifs, comme :

- Facteurs organisationnels
    - Compétences, formations et pénuries de ressources

    - Problématiques sur le personnel
    - Problématiques politiques, comme :
        - Problèmes avec les parties prenantes communiquant leurs besoins et leurs attentes
        - Échec pour suivre les informations trouvées dans les revues (exemple : ne pas améliorer les pratiques documentant les exigences)
    - Attitude ou attentes incorrectes concernant l'Ingénierie des Exigences
- Problématiques techniques
    - Problèmes pour définir les bonnes exigences
    - La mesure selon laquelle des exigences ne peuvent pas répondre à des contraintes existantes données
    - Environnement de développement ou de test pas prêt à temps
    - Mauvaise qualité de la conception, du code, des données de configuration, des données de tests et des tests
- Problématiques fournisseur
    - Echec d'une tierce partie (par exemple, des composants non livrés à temps)
    - Problématiques contractuelles

Risques produit (K2)

Les zones d'échec potentielles (événements futurs adverses ou dangers) dans le logiciel ou système sont connues comme des risques produit, car ils sont un risque sur la qualité du produit. Cela inclut :

- Risque important d'échec de livraison du logiciel (logiciel ou système incapable de fournir une fonction requise dans des limites spécifiées), documentation logicielle de basse qualité (incomplète, incohérente, difficile à maintenir)
- Le potentiel qu'un logiciel / matériel pourrait causer comme préjudice à un individu ou une société
- Caractéristiques logicielles médiocres (par exemple, fonctionnalité, fiabilité, facilité d'emploi ou performance)
- Mauvaises qualité et intégrité des données (par exemple, problématiques de migration de données, problèmes de conversion de données, problèmes de transport de données, violation de standards de données)
- Logiciel qui ne réalise par les fonctionnalités attendues et qui ne satisfait pas les besoins des parties prenantes
- Risques business basés sur une mauvaise qualité

3.2.3 Gestion de risque (K2)

La gestion de risque est un processus d'identification, d'évaluation et de priorisation, de planification de réaction sur les risques et de résolution et de supervision des risques. Cela permet

d'identifier des facteurs potentiels qui peuvent avoir un effet négatif sur l'exécution d'un projet et préparer une action appropriée pour faire face à un risque si il survient.

Différents risques peuvent provenir de différents groupes de parties prenantes : par exemple, l'équipe de développement verra des risques différents de ceux considérés par les parties prenantes métier ou les utilisateurs finaux.

La gestion des risques comprend les activités suivantes [ISTQB] :

- Identification du risque
- Analyse du risque
- Atténuation du risque

Traitements potentiel du risque

Les techniques pour gérer le risque peuvent être divisées en quatre catégories principales :

- Évitement
- Réduction
- Partage
- Maintien

Plan de gestion de risque

Le plan de gestion de risques devrait être créé avant et après (périodiquement mis à jour) la création du plan projet. Le plan de gestion des risques devrait fournir des contrôles de sécurité efficaces pour gérer les risques et contenir un planning pour le contrôle de l'implémentation et des personnes responsables de ces actions.

Le plan de gestion des risques comprend :

- la liste des risques
- la probabilité d'occurrence et/ou la priorité
- la sévérité de l'impact de chaque risque (incluant le coût si applicable)
- les stratégies de réduction pour chaque risque (incluant les personnes/groupes responsables de prendre les actions)
- la matrice d'évaluation des risques

3.2.4 Analyse des modes de défaillance et de leurs effets (K2)

Une technique commune de gestion de risques (identification, analyse et planning de réactivité) est l'analyse des modes de défaillances et de leurs effets (AMDE).

AMDE permet de prioriser les défaillances potentielles par rapport à la gravité des conséquences, la fréquence d'occurrence et comment elles peuvent être facilement détectées. AMDE documente aussi la connaissance actuelle et les actions à propos des risques de défaillances pour une utilisation en amélioration continue. Dans la plupart des cas, AMDE est utilisé pendant la phase de conception d'un projet et son objectif principal est d'éviter les défaillances futures. Dans des phases suivantes, elle peut être utilisée pour contrôler des processus. AMDE devrait commencer dans les premières étapes conceptuelles du projet et continuer tout au long du cycle de vie. Idéalement, AMDE devrait être planifiée dès que les premières informations sont disponibles.

Les résultats d'une AMDE sont les actions pour prévenir et réduire la gravité ou la probabilité des défaillances.

Étapes d'implémentation de l'AMDE

L'AMDE est déroulée en trois étapes principales :

Étape 1 : gravité (identification de la gravité des défauts potentiels)
Etape 2 : occurrence (identification de combien de fois un défaut potentiel peut arriver)
Etape 3 : détection (identification des techniques de détection des défauts)

4 Rôles et responsabilités (K2)	60 minutes

Objectifs d'Apprentissage (OA) pour le niveau Fondation de l'Ingénierie des Exigences

Les objectifs identifient ce que vous êtes capable de faire, en ayant suivi chaque module.

4.1 Rôles fondamentaux (K1)

OA-4.1.1 Rappeler les rôles fondamentaux qui sont dans l'Ingénierie des Exigences (K1)

OA-4.1.2 Décrire le but et le rôle d'une partie prenante (K2)

4.2 Tâches de l'Ingénierie des Exigences (K2)

OA-4.2.1 Décrire les tâches de l'Ingénierie des Exigences (K2)

OA-4.2.2 Rappeler les caractéristiques d'un Professionnel de l'Ingénierie des Exigences (K1)

4.1 Rôles fondamentaux (K1)	20 minutes

Termes

Client, Contractant, Partie prenante

4.1.1 Rôles fondamentaux (K2)

Client

Une personne, un groupe ou une organisation exigeant une solution

Contractant (fournisseur, vendeur)

Une personne, un groupe ou une organisation fournissant la solution

Le client formule ses besoins et fournit des besoins et des attentes métier initiales. Généralement, ses besoins sont fournis en même temps que la requête de service / offre. Il est de la responsabilité du vendeur d'explorer ces besoins et d'extraire des exigences à partir de ces données.

Le contractant fournit des solutions basées sur les besoins client.

Rôles dans l'Ingénierie des Exigences

Gestionnaire d'exigence

Un gestionnaire d'exigence est une personne responsable de la documentation, l'analyse, la traçabilité, la priorisation et l'accord sur les exigences et aussi le contrôle des changements et la communication vers les parties prenantes correspondantes.

Développeur d'exigence

Un développeur d'exigence est une personne orientée technique généralement impliquée dans l'élucidation, l'analyse et la priorisation des exigences.

4.1.2 Parties prenantes (K2)

La partie prenante est un groupe ou un individu qui est affecté par ou est en quelque sorte responsable du résultat d'une action. Les parties prenantes du projet sont des individus et des organisations qui sont activement impliqués dans le projet, ou dont les intérêts peuvent être touchés à la suite de l'exécution du projet ou de l'achèvement du projet.

Les parties prenantes peuvent être des personnes physiques, des sociétés ou des personnes morales.

Les parties prenantes ont souvent des conflits d'intérêts entre elles. Cela aboutit souvent à des exigences contradictoires. Le problème des exigences contradictoires doit être abordée lors de la phase d'analyse des exigences.

Il peut y avoir plusieurs catégories de parties prenantes, parmi lesquelles :

- Les clients
- Les utilisateurs finaux
- Les managers
- Les personnes impliquées dans les processus organisationnels
- Les ingénieurs chargés du développement et de la maintenance des systèmes
- Les clients de l'organisation qui vont utiliser le système
- Les organismes externes (réglementation)
- Les experts du domaine

Les parties prenantes typiques sont :

- Client
- Utilisateur final
- Chef de projet
- Chef de produit
- Analyste système
- Analyste métier
- Représentants business
- Personnel marketing et vente
- Développeur de logiciels
- Personnel de l'assurance qualité
- Experts techniques (architectes, ingénieurs base de données)
- Gestionnaire des changements
- Équipe cœur de projet
- Équipe de management

L'identification de toutes les parties prenantes est nécessaire pour prendre en considération tous les points de vue dans la solution envisagée.

Pour les organismes de formation: description des parties prenantes typiques (par exemple, directeur général, directeur de projet, client)

4.2 Tâches de l'Ingénierie des Exigences (K2)	30 minutes

4.2.1 Tâches de l'Ingénierie des Exigences (K2)

Les tâches principales de l'ingénierie des exigences sont les suivantes :

- Analyse des processus métier réalisée au sein d'une organisation
- Identification et analyse des exigences
- Structuration et modélisation des exigences
- Assurance qualité des exigences et des spécifications
- Création de la spécification des exigences
- Analyse des risques (dans le contexte des exigences)
- Gestion du changement des exigences
- Accord sur les exigences avec les parties prenantes

4.2.2 Connaissance d'un Professionnel de l'Ingénierie des Exigences (K1)

Un professionnel d'Ingénierie des Exigences devrait avoir les compétences suivantes :

- Compétence de modération
- Confiance en soi
- Capacité à convaincre
- Compétences linguistiques
- Capacité à communiquer
- Précision
- Capacité d'analyse, pensée réfléchie
- Capacité à agir de façon structurée
- Compétence méthodologique
- Résistance au stress

5 Identification des Exigences (K2)	150 minutes

Objectifs d'Apprentissage (OA) pour le niveau Fondation de l'Ingénierie des Exigences

Les objectifs identifient ce que vous êtes capable de faire, en ayant suivi chaque module.

5.1 Client (K1)

OA-5.1.1 Rappeler le contenu d'un contrat (K1)

OA-5.1.2 Identifier ce qui devrait être considéré lors de l'évaluation des exigences (K2)

5.2 Visions et objectifs projet (K2)

OA-5.2.1 Expliquer les caractéristiques d'une vision projet typique (K2)

5.3 Identification des parties prenantes (K2)

OA-5.3.1 Expliquer comment les parties prenantes peuvent être identifiées (K2)

OA-5.3.2 Identifier les parties prenantes pour un projet spécifique (K3)

5.4 Techniques pour identifier les exigences (K2)

OA-5.4.1 Identifier les objectifs de l'identification des exigences (K2)

OA-5.4.2 Appliquer les différentes techniques d'identification des exigences (K2)

5.5 Exigences fonctionnelles et non-fonctionnelles (K2)

OA-5.5.1 Décrire les caractéristiques des exigences fonctionnelles et non-fonctionnelles (K2)

OA-5.5.2 Comparer les différences entre les exigences fonctionnelles et non-fonctionnelles (K2)

5.6 Description des exigences (K2)

OA-5.6.1 Décrire le contenu d'un document standard des exigences (K2)

OA-5.6.2 Décrire les caractéristiques d'exigences de qualité (K2)

OA-5.6.3 Construire une exigence (K3)

5.1 Client (K1)	20 minutes

Termes

Client, Contrat

5.1.1 Client (K1)

Le client, organisation ou personne qui achète le logiciel, est l'une des principales parties prenantes du projet. Les besoins client doivent être satisfaits.

Le client doit toujours être impliqué. L'objectif est de comprendre le client et de développer une compréhension mutuelle avec lui. Le contractant (vendeur) doit donc toujours se mettre dans la position du client.

En évaluant les exigences à des fins de planification du projet (ce qui est l'un des sujets qui seront couverts par le projet), les différents points de vue doivent être pris en considération comme une exigence spécifique qui peut avoir des priorités et gravités différentes en fonction des parties prenantes.

5.1.2 Contrat (K1)

L'accord (contrat) devrait officiellement définir et décrire ce que veut le client. Cela doit assurer que l'intérêt du client est au centre (c'est à dire le vendeur n'est pas contraint à la solution qu'il préfère mais il analyse les besoins du client et recommande une solution qui permet de satisfaire ses besoins dans les meilleures conditions).

L'accord :

- Doit être compatible avec les ressources disponibles pour mettre en œuvre la solution,
- Est basé sur: les estimations, les délais, les prix et les plans du projet.

L'Ingénierie des Exigences fournit des informations en entrée pour de telles estimations.

Le contrat devrait inclure :

- Une description brève de la solution planifiée
- La liste des exigences de haut niveau priorisées
- Les critères d'acceptation pour chaque exigence
- La liste des éléments produits (documentation, code, logiciel fonctionnant)

- Les dates limites pour le développement et la livraison du produit
- D'autres besoins et attentes comme la technologie à utiliser de préférence, les besoins en ressources, etc.

5.2 Visions et objectifs projet (K2)	20 minutes

Termes

Objectif, Vision

5.2.1 Vision (K2)

Le développement de visions du projet est la première étape de l'ingénierie des exigences.

Une vision devrait :

- Définir les clients, les marchés et les concurrents
- Définir les objectifs à atteindre
- Permettre d'atteindre une compréhension commune de toutes les parties prenantes

Il est crucial d'avoir une définition claire des visions du projet.

Pour les organismes de formation : présentation des visions typiques d'un projet

Questions importantes à propos des visions d'un projet (K2)

- Que va changer le projet ?
- Pourquoi le projet est-il nécessaire ?
- Que se passe-t-il une fois le projet terminé ?
- Qui profitera du projet ?
- Quels sont les coûts prêts à être assumés ?
- Quels sont les risques prêts à être assumés ?

Pour chaque projet, la vision doit être ré estimée.

<u>Influences sur la vision du projet (K1)</u>

Les facteurs suivants peuvent influencer la vision :

- Les clients
  - Les objectifs des clients
  - Les préférences des clients
- Stratégie

    - Stratégie d'une organisation
    - Positionnement sur le marché
    - Directions à suivre
- Concurrence
    - Données de concurrence
    - Développement marché
- Produits
    - Niveau d'innovation
    - Groupe cible
- Technologies
    - Nouveaux outils
    - Nouveaux standards
- Ressources disponibles
    - Temps, collaborateurs, capacités

5.3 Identification des parties prenantes (K2)	20 minutes

Termes

Partie prenante

5.3.1 Identification des parties prenantes (K2)

Toutes les parties prenantes côté client et côté fournisseur doivent être identifiées.

Chaque partie prenante ou chaque groupe de parties prenantes peut fournir de nouvelles exigences et influencer la conception de la solution envisagée. Si toutes les parties prenantes ne sont pas identifiées, il y a un risque que certaines exigences importantes ou des limitations restent inconnues et ne soient pas pris en compte dans la conception. Oublier des parties prenantes peut entraîner des changements complexes exigés dans le logiciel à un stade avancé du projet ou après la sortie du système dans un environnement de production.

Certaines parties prenantes peuvent créer des groupes d'intérêt (par exemple, toutes les parties prenantes métier). Les groupes d'intérêt devraient être réunis, car cela permettrait de gérer leurs besoins de manière plus efficace.

5.3.2 Procédure d'identification et d'évaluation des parties prenantes (K2)

La procédure d'identification et d'évaluation des parties prenantes comprend les activités suivantes :

- Identification des parties prenantes (analyse des processus métier, détermination des propriétaires des processus et des produits, analyse de la structure organisationnelle et du marché)
- Classification des parties prenantes en groupes (si possible)
- Détermination des relations
- Identification des conflits potentiels
- Analyse des conflits, leurs sources et identification des opportunités gagnant-gagnant
- Identification des parties prenantes minimisant des risques pour les impliquer davantage dans les activités du projet
- Identification des points de vue des parties prenantes

Pour les organismes de formation : explication de l'identification et l'évaluation des parties prenantes

5.4 Techniques d'identification des exigences (K2)	40 minutes

Termes

Apprentissage, brainstorming, représentant du client, observation terrain, entretien, questionnaire, revue, auto-enregistrement, groupe de travail

5.4.1 Identification des parties prenantes (K2)

Les principaux objectifs de l'identification des exigences sont les suivants :

- Identifier toutes les fonctions, caractéristiques, limitations et attentes désirées
- Orienter les exigences vers la vision du projet
- Détailler les exigences de haut-niveau et décrire clairement les fonctions et les services
- Exclure les fonctions et les fonctionnalités que le client ne veut pas

5.4.2 Techniques (K1)

Les techniques les plus courantes pour identifier les exigences sont :

- Questionnaires
- Entretiens
- Auto-enregistrement
- Représentants du client sur site
- Identification sur la base de documents existants
- Réutilisation (Réutiliser la spécification d'un certain projet)
- Brainstorming
- Observation terrain
- Apprentissage
- Ateliers

5.4.2.1 Questionnaires

Un questionnaire peut être constitué de questions ouvertes ou fermées. Une question ouverte requiert de celui qui répond de formuler sa propre réponse. Dans le cas d'une question fermée, la

personne qui répond est invitée à choisir une réponse parmi un certain nombre d'options possibles. Ces options doivent être mutuellement exclusives.

Avantages :

- Coûts mineurs
- Un public large peut être ciblé

Inconvénients :

- Non applicable pour recueillir des connaissances implicites
- Faible taux de retour sans motivation des personnes qui répondent
- Des questionnaires peuvent souvent être directifs, ce qui empêche l'identification des besoins réels des utilisateurs

5.4.2.2 Entretien

L'entretien est une technique de conversation où l'intervieweur demande à l'interviewé d'obtenir des informations sur un sujet spécifique. Cette technique est très interactive et permet de modifier l'ordre des questions préalablement préparées selon les réponses de l'interviewé et de la situation.

De bons entretiens sont plus difficiles à mener que l'on pense à cause du comportement d'une conversation normale (par exemple, les phrases de fin de l'interlocuteur), ce qui peut conduire à des interprétations introduites dans les données. L'intervieweur doit poser des questions ouvertes pour obtenir des informations et poser des questions fermées juste pour confirmer les informations recueillies (par exemple, confirmer les exigences déjà identifiées).

Avantages :

- La progression peut être adaptée en fonction de la personne qui répond

Inconvénients :

- Consommateur de temps
- Reproduction insuffisante des résultats (difficulté d'obtenir les mêmes réponses lorsqu'on répète un entretien)

5.4.2.3 Auto-enregistrement

Dans cette technique, la partie prenante (par exemple, un utilisateur final) documente ses activités exercées pour accomplir une tâche spécifique.

En plus de documenter les activités «tel quel», l'utilisateur décrit également les modifications, les désirs et besoins.

Les techniques associées à cette approche sont : des démonstrations ou des revues de documents.

Avantages :

- Faible temps et effort pour l'Ingénieur des Exigences côté vendeur logiciel

Inconvénients :

- Négliger des activités «automatisées» (comme l'impression et la réception de copie imprimée)
- Très dépendante de la motivation et l'expérience de l'utilisateur

5.4.2.4 Représentants du client sur site

Cette approche est l'une des méthodes les plus efficaces pour l'identification des exigences (et la validation) car elle permet au représentant de surveiller systématiquement le progrès, de vérifier l'exactitude de la conception et de fournir des commentaires et des informations supplémentaires si nécessaire.

Avoir des représentants client sur site est l'une des principales règles des méthodes Agiles.

Avantages :

- Retours rapides
- Fourniture d'exigences orientées utilisateur qui sont facilement acceptées

Inconvénients :

- Coûts élevés pour le client
- Coûts d'adaptation

5.4.2.5 Identification des exigences sur la base de documents existants

Cette technique peut être utilisée dans le cas où il y a de la documentation déjà existante qui peut aider à identifier les besoins au sein d'une organisation. Cette documentation peut être :

Modèles de processus et cartographie

Descriptions de processus
Organigrammes
Spécification produit (dans le sens, sortie de processus spécifique)
Procédures (i.e. procédures de travail)
Standards et instructions

Les exigences identifiées sont la base pour une analyse ultérieure des exigences et les besoins doivent être détaillés et étendus avec d'autres exigences correspondantes et liées.

Avantages :

- Aucune fonctionnalité n'est négligée

Inconvénients :

- Coûteux
- Non applicable quand il n'y a pas de documents ou seulement des documents de base au sein d'une organisation
- Non applicable lorsque la documentation n'est pas maintenue correctement à jour

5.4.2.6 Réutilisation (réutilisation de la spécification d'un certain projet)

La spécification d'un certain projet peut être réalisée quand une organisation a déjà réalisé un ou plusieurs projets similaires au projet en cours. La spécification des exigences réalisée pour un (ou plusieurs) projet(s) précédent (s) peut être utilisée dans un autre projet pour raccourcir la durée de l'analyse des exigences et de la documentation et donc - permet de lancer l'implémentation plus tôt.

Dans la plupart des cas, seules certaines parties des spécifications existantes peuvent être utilisées dans de nouveaux projets. La documentation à réutiliser doit toujours être vérifiée par rapport à la conformité avec les besoins et exigences courants et correctement ajustés.

Avantages :

- Réduction des coûts

Inconvénients :

- Coûts élevés du premier projet
- La réutilisation des exigences peut demander une gestion du changement importante et coûteuse sielle n'a pas été prise en compte correctement dans les projets précédents

5.4.2.7 Brainstorming

Le brainstorming est une technique couramment utilisée pour obtenir des exigences relatives à des domaines nouveaux ou pas bien connus concernant une nouvelle activité d'une organisation ou une fonctionnalité du système planifié. Cette technique permet de recueillir de nombreuses idées de diverses parties prenantes en temps limité et à faible coût. Lors de la séance de brainstorming, les participants soumettent des idées et des concepts relatifs à problème donné.

Avantages :

- Faibles coûts
- Chance de recueillir de nombreuses idées valables en temps limité

Inconvénients :

- Difficile si participants non motivés
- Difficile à appliquer dans des équipes distribuées

5.4.2.8 Observation terrain

L'observation sur le terrain permet des activités d'observation des utilisateurs et des processus en cours et d'identifier les exigences du système sur cette base. Une observation réalisée sur site permet de regarder les utilisateurs en train de travailler et de documenter les processus, les tâches et les résultats. Dans certains cas, l'observation est prolongée par des entretiens des utilisateurs au sujet de leurs emplois et sur la façon dont ils réalisent leurs tâches.

Avantages :

- Possibilité d'observer les utilisateurs pendant leur travail et d'identifier de réels besoins
- Utile lorsque les parties prenantes ont des problèmes pour exprimer leurs besoins

Inconvénients :

- Des cas exceptionnels peuvent être omis
- Non applicable dans certaines situations (par exemple, en sécurité de fonctions et pour des raisons légales)

5.4.2.9 Apprentissage

Le but de l'apprentissage est de recueillir des exigences d'un client, en particulier dans le cas où les processus et les activités effectués par le personnel client ne sont pas faciles à décrire en utilisant

d'autres techniques, comme des entretiens, ou si le client a des problèmes pour exprimer clairement les exigences concernant le logiciel prévu.

L'apprentissage est un processus d'étude, par le client de son travail. Le client, qui connaît le mieux comment faire son travail spécifiques, l'enseigne à l'ingénieur des exigences - comme un maître à son élève.

Avantages :

- Aide pour surmonter la difficulté que les employés du client peuvent avoir à penser à des choses de façon abstraite et à décrire leurs tâches verbalement

Inconvénients

- Coûts élevés et consommateur de temps
- Non applicable dans des environnements dangereux

5.4.2.10 Ateliers

L'atelier est une sorte de réunion sur un sujet spécifique (préalablement défini et annoncé aux participants), impliquant généralement les parties prenantes représentant différents secteurs et / ou domaines pendant une période courte et intensive.

Les ateliers peuvent avoir des objectifs différents

- Identifier des exigences (i.e. afin d'établir la portée d'une solution)
- Découvrir des exigences cachées (i.e. des exigences qui ne sont pas directement indiquées ou même pas réalisées par les parties prenantes, mais nécessaires pour accomplir certains de leurs besoins ou des exigences de niveau supérieur)
- Développer des exigences (de façon détaillée) dans un secteur nouvellement identifié
- Prioriser les exigences
- Parvenir à un consensus sur les exigences quand il s'agit de s'accorder sur les exigences (signature)
- Examiner les résultats de processus spécifiés ou activités (i.e. revoir la spécification des exigences fonctionnelles) et résoudre les problèmes qui pourraient être apparus

Avantages

- Faire participer les personnes qui ont différents points de vue sur un problème donné
- Permettre de déterminer et de décrire les exigences provenant de différentes perspectives
- Permet de découvrir rapidement et de résoudre des conflits potentiels entre les exigences des parties prenantes

Inconvénients :

- Difficile dans le cas d'équipes géographiquement distribuées
- Disponibilité de toutes les personnes requises pour participer à l'atelier
- Le consensus n'est pas nécessairement facile à atteindre lors d'un atelier, et la discussion peut décrocher sur de (petites) questions problématiques, rendant ainsi le processus long et démotivant pour les participants

5.5 Exigence fonctionnelle et non-fonctionnelle (K2)	20 minutes

Termes

Exigences fonctionnelles, exigences non-fonctionnelles

Pour passer l'examen, les étudiants doivent être capables de fournir des exemples d'exigences fonctionnelles et non-fonctionnelles et de détailler les caractéristiques qualité.

5.5.1 Exigences fonctionnelles (K2)

Les exigences fonctionnelles précisent ce que le système fait. Elles précisent les fonctions du système perçues par l'utilisateur final. Les exigences fonctionnelles décrivent aussi les déclencheurs des processus (action de l'utilisateur, entrée / sortie de données causant le lancement d'un processus métier).

Les exigences fonctionnelles devraient être classées par rapport aux caractéristiques qualité suivantes [ISO / IEC 25000] (K1) :

- Pertinence
- Précision
- Interopérabilité
- Fonctionnalité
- Conformité
- Sécurité

5.5.2 Exigences non-fonctionnelles (K2)

Les exigences non-fonctionnelles spécifient comment le système fonctionne. Elles décrivent les attributs qualité de tout le système ou de ses composants ou fonctions spécifiques. Elles limitent la solution en exigeant des paramètres d'efficacité spécifiques.

Les exigences non-fonctionnelles sont difficiles à décrire, par conséquent, elles sont souvent exprimées de façon vague ou pas documentées du tout. Cela les rend difficiles à tester. Une attention particulière doit donc être mise sur les exigences non-fonctionnelles à toutes les étapes du processus IE.

En raison des problèmes d'expression des exigences non-fonctionnelles, elles peuvent être difficiles à tester. Les exigences non-fonctionnelles devraient donc être exprimées clairement et être mesurables.

Les exigences non-fonctionnelles sont [ISO / IEC 25000] (K1)

- Fiabilité
- Facilité d'emploi
- Efficacité
- Maintenabilité
- Portabilité

Les exigences non-fonctionnelles spécifient les critères qui peuvent être utilisés pour juger de l'exploitation d'un système donc elles ont un grand impact sur la satisfaction du client utilisant le logiciel. Les exigences fonctionnelles ont pour but de fournir des fonctions; les exigences non-fonctionnelles déterminent la facilité et l'efficacité de l'utilisation des fonctions.

5.6 Description des exigences (K2)	30 minutes

5.6.1 Description des exigences (K2)

Les exigences doivent être clairement et précisément spécifiées. Elles doivent être mesurables afin de s'assurer qu'elles sont testables et que leur implémentation peut être correctement vérifiée. Il est important de rappeler qu'un langage commun a certaines limites et inconvénients. Ceci peut causer une description des exigences obscure et ambiguë. Par conséquent, des standards appropriés et des modèles devraient être utilisés autant que possible. Les standards fournissent une compréhension commune et de meilleures pratiques de spécification avec des modèles qui limitent le langage qui peut être utilisé.

En plus de standards et de modèles, les vocabulaires sont un outil important en vue de faciliter la communication entre les différentes parties prenantes et d'introduire un peu de contrôle dans l'ambiguïté du langage naturel.

La description d'une exigence doit répondre à différents critères (par exemple, clair, précis, non ambiguë, mesurable, sans mots inutiles, etc.).

5.6.2 Procédure pour la construction des exigences (K2)

La construction d'une exigence est réalisée d'après les étapes suivantes :

1. Détermination du processus affecté
   - Mettre l'accent sur la fonctionnalité
   - Déterminer les entrées et sorties
2. Classification de l'activité du système
   - Identification de l'activité indépendante du système
   - Identification des interactions avec l'utilisateur
   - Identification des exigences d'interfaces
3. Détermination de l'engagement juridique
   - Clarification de l'engagement juridique avec des mots-clés (devrait, doit, etc.)
4. Raffinement de la description du processus
   - Description détaillée des objets et des points d'intégration
5. Contraintes logiques et temporelles
   - Établir les conditions aux limites

Exemple: une exigence relative à la génération d'une facture avec les données provenant d'un système externe.

Étape de la procédure	Sortie – une description d'exigence
Détermination du processus : Le processus système de génération d'une facture	Le système génère la facture
Classification de l'activité système Génération d'une facture après la commande utilisateur en cliquant sur un bouton L'interface avec le système externe est établie	Après que les utilisateurs aient soumis une requête de facture, le système génère une facture en utilisant des données provenant du système externe
Détermination de l'engagement légal Le système doit générer une facture avec données correctes provenant du système externe	Après que les utilisateurs aient soumis une requête de facture, le système doit générer une facture en utilisant des données provenant du système externe
Raffinement de la description du processus Définition du nom du système financier externe	Après que les utilisateurs aient soumis une requête de facture, le système doit générer une facture en utilisant des données provenant d'un système de Finance SAP
Contraintes logiques et temporelles Détermination de la contrainte de temps – durée maximale de l'opération : 30 secondes	Après que les utilisateurs aient soumis une requête de facture, le système doit générer une facture en utilisant des données provenant d'un système de Finance SAP dans les 30 secondes

Pour les organismes de formation : description des différentes étapes de construction d'une exigence

5.6.3 Document d'exigence (K2)

Le contenu standard du document d'exigence est [IEEE 830] :

Table des matières

1. Introduction

    1.1 Objectif

    1.2 Portée

    1.3 Définitions, acronymes et abréviations

6 Spécification des Exigences (K2)	100 minutes

Objectifs d'Apprentissage (OA) pour le niveau Fondation de l'Ingénierie des Exigences

Les objectifs identifient ce que vous êtes capable de faire, en ayant suivi chaque module.

6.1 Spécification (K2)

OA-6.1.1 Décrire l'objectif et le contenu d'une spécification d'exigences (K2)

OA-6.1.2 Décrire les caractéristiques d'une spécification d'exigences (K2)

OA-6.1.3 Décrire l'objectif et le contenu d'une spécification de solution (K2)

OA-6.1.4 Décrire les caractéristiques d'une spécification de solution (K2)

OA-6.1.5 Rappeler les standards qui sont importants pour la spécification d'exigences et la spécification de solution (K1)

OA-6.1.6 Expliquer ce qu'est une « user story » (K2)

6.2 Procédure (K3)

OA-6.2.1 Appliquer une procédure typique pour la spécification d'exigences (K3)

6.3 Formalisation (K2)

OA-6.3.1 Expliquer les différents degrés de formalisation qui existent pour la formalisation d'exigences (K2)

OA-6.3.2 Appliquer le niveau de formalisation spécifique pour un scenario donné (K3)

5.4 Qualité des exigences (K2)

OA-6.4.1 Rappeler les conséquences des erreurs dans les exigences (K1)

OA-6.4.2 Décrire les possibilités pour éviter les erreurs dans les exigences (K2)

6.1 Spécifications (K2)	30 minutes

Termes

IEEE830, Spécification d'exigences, Spécification de solution

6.1.1 Description des exigences (K2)

Une spécification est un ensemble explicite d'exigences à satisfaire par un matériel, produit ou service.

La spécification permet de suivre et de gérer les exigences. Dans la spécification, les exigences sont spécifiées de manière structurée et sont modélisées séparément (les exigences sont modélisées « indépendamment », comme les exigences de haut niveau qui sont ventilées jusqu'à un niveau où chaque exigence constitue une entité «indépendante» qui peut ensuite être développée et suivie). La spécification est un accord formel sur les exigences à implémenter dans un système logiciel prévu (ou dans toute autre forme de solution).

Le terme « spécification » peut être utilisé dans le contexte des exigences et de la solution.

6.1.2 Spécifications des exigences (K2)

La spécification des exigences décrit le domaine du problème (une zone d'intérêt, par exemple, une solution à un problème métier, une nouvelle fonctionnalité, etc.) et contient au moins les informations suivantes

- Exigences client
- Limitations
- Critères d'acceptation

La création de la spécification des exigences client devrait être la tâche du client. Toutefois, dans certains cas, le vendeur peut soutenir la préparation des spécifications des exigences.

La création d'autres types de spécifications pour les exigences système, exigences logicielles, exigences de sécurité, exigences environnementales, exigences légales, etc. sont des tâches pour d'autres rôles.

6.1.3 User stories (K2)

Les « user stories » sont utilisées dans les méthodologies de développement logiciel Agile. Les « user stories » sont une façon rapide de gérer les exigences client / utilisateur. Le but de l'utilisation des « user stories » est de permettre de répondre plus rapidement et avec moins de coût au changement rapide des exigences du monde réel.

Les « user stories » décrivent les fonctionnalités qui seront utiles. Les « user stories » sont composées de trois aspects :

- Une description écrite de la « user story » pour planification et rappel
- Échanges sur la « user story » qui servira à étoffer les détails
- Les tests qui véhiculent et documentent les détails et seront utilisés pour déterminer quand une « user story » est terminée [Mike Cohn : User Stories appliquée. 2009]

6.1.4 Spécifications de solution (K2)

Les spécifications de solution sont aussi appelées des spécifications fonctionnelles, des spécifications des exigences système ou des spécifications des exigences logicielles et décrivent le domaine de la solution.

Une spécification fonctionnelle est un document qui décrit clairement les exigences pour la solution. La spécification fonctionnelle est la base pour un futur développement système, donc elle doit fournir des informations précises sur tous les aspects fonctionnels du logiciel à implémenter. A partir de cela, les architectes et les développeurs sont en mesure de concevoir efficacement les aspects techniques du système. La spécification fonctionnelle fournit des conseils pour les testeurs pour la vérification de chaque exigence fonctionnelle (par exemple, une spécification est un élément de base du test).

La spécification fonctionnelle ne décrit pas comment la fonction du système sera mise en œuvre et quelle technologie à utiliser. Elle se concentre sur la fonctionnalité, sur les interactions entre l'utilisateur et le logiciel.

Le but de la spécification fonctionnelle peut être

- Fournir la base pour une compréhension commune de la portée et de la fonctionnalité de la solution à mettre en œuvre
- Assurer un consensus d'équipe sur ce que le système doit faire avant de commencer l'écriture du code source, les manuels, la préparation des données et des tests
- Fournir à l'équipe de développement la description détaillée de la fonctionnalité nécessaire en termes d'interactions entre les utilisateurs et le logiciel
- Fournir la base pour l'oracle de test pour l'équipe de test

6.1.5 Standards importants (K1)

Les normes suivantes peuvent être utilisées pour la rédaction de spécifications :

- IEEE 1362 (Spécifications de performance du système)
- IEEE 830 (Spécification des exigences logicielles)
- IEEE 1233 (Spécifications du système fonctionnel)

6.2 Procédure (K3)	30 minutes

6.2.1 Procédure de spécification de solution (K3)

La spécification sert comme une activité de formalisation des résultats de l'Analyse des Exigences (K2).

L'identification, l'analyse et les activités de spécification des exigences conduisent à un accord sur les exigences (voir Accord sur les exigences (K2)).

Les exigences, une fois identifiées, analysées et modélisées devraient être documentées de manière claire et sans ambiguïté.

Une procédure de spécification de solution comprend les activités suivantes :

1. Identification des parties prenantes
2. Définition de la vision et de l'objectif
3. Détermination des exigences
4. Spécification structurée des exigences
5. Description de l'environnement système
6. Détermination de la solution (définition du système et du périmètre avec des aspects pertinents extérieurs, mais influençant, le périmètre, comme les interfaces vers des systèmes externes)
7. Analyse des exigences
8. Modélisation du problème
9. Modélisation de la solution

La procédure formalise les résultats du processus d'Analyse des Exigences. Le modèle de solution est une base pour la conception et l'implémentation.

La procédure implique un certain nombre de parties prenantes qui supportent les travaux de spécification dans différents domaines. La sortie de la procédure, la Spécification de Solution, sert de point de départ pour le logiciel, le matériel et la conception de base de données. Elle décrit la fonction (spécifications fonctionnelles et non fonctionnelles) du système, la performance du système et les contraintes opérationnelles et d'interface utilisateur.

6.3 Formalisation (K2)	20 minutes

Termes

Niveau formel, niveau non-formel, niveau semi-formel

6.3.1 Degré de formalisation (K2)

Les exigences de spécification peuvent être définies avec différents degrés de formalisation :

- Non-formel
- Semi-formel
- Formel

6.3.1.1 Niveau non-formel

L'approche non-formelle pour la rédaction de spécification signifie que le document est écrit en langage naturel, sans aucune notation formelle. Cette approche peut être utilisée lorsque les lecteurs n'ont pas d'expérience avec des langages de spécification plus formels et techniques et auraient des difficultés pour comprendre le contenu du document. La faiblesse principale de cette approche est qu'elle est ambiguë et peut conduire à des problèmes de compréhension et d'interprétation. En outre, la spécification non formelle n'est pas une bonne base ni pour l'implémentation ni pour le test parce qu'elle n'est pas suffisamment claire ni précise.

6.3.1.2 Niveau semi-formel

Une spécification semi-formelle comprendra quelques notations formelles et sera bien structurée. Habituellement, de telles spécifications sont basées sur un modèle spécifique (souvent issus de standards appropriés). Des spécifications semi-formelles peuvent exprimer des exigences dans une forme de modèles et utiliser un langage commun formalisé.

UML et SysML sont parmi les notations semi-formelles les plus utilisées pour rédiger une documentation semi-formelle.

6.3.1.3 Niveau formel

Une spécification formelle est une description mathématique du logiciel qui peut être utilisée pour réaliser une implémentation. La spécification formelle décrit ce que le système doit faire, ne décrit généralement pas comment le système devrait le faire. Comme elle est basée sur des formules mathématiques et qu'elle est plus difficile à apprendre, la spécification formelle est utilisée assez rarement et nécessite des connaissances mathématiques.

6.4 Qualité des exigences (K2)	20 minutes

Termes

Checkliste, caractéristique qualité, revue, validation, vérification, traçabilité

6.4.1 Contexte (K2)

Défauts dans les exigences comme une cause de coûts élevés (K2).

Comme les exigences sont la base du développement système, toute erreur ou exigence manquante va se propager à tous les autres processus de développement dans le projet. Il est important de noter, que les défauts résultant d'exigences de faible qualité sont plus chers à corriger dans les phases ultérieures du projet que les autres types de défauts. En outre, plus les défauts sont détectés tard, plus ils coûtent chers.

Par conséquent, l'utilisation de la vérification (avons-nous réalisé le produit correctement) et de la validation (avons-nous réalisé le bon produit) des exigences est nécessaire.

Les exigences doivent être documentées et testées au regard des critères de qualité (voir chapitre 1.1 Exigence (K2)).

6.4.2 Mesures d'amélioration de la qualité et assurance qualité des exigences (K2)

Les outils et techniques suivants peuvent être utilisés pour améliorer la qualité et assurance qualité des exigences :

- Standards et modèles
- Revues et inspections
- Traçabilité
- Prototypage
- Respect des critères de qualité (les critères de qualité peuvent être : complétude, exactitude, conformité de la spécification des exigences à des standards appropriés)

La qualité d'une spécification d'exigences peut être améliorée en incluant les éléments suivants :

- Description du but du document, de la portée, des définitions et du glossaire
- Description des objectifs à différents niveaux (i.e. une spécification des exigences de haut niveau a des objectifs différents de ceux d'une spécification d'exigences fonctionnelles détaillée)

- Définition des contraintes de conception et d'implémentation
- Calibration / priorisation des exigences
- Déclarations claires de ce que le système devrait faire plutôt que comment il doit le faire
- Documentation des aspects législatifs, des hypothèses, des règles métier et des dépendances
- Éviter des descriptions complémentaires de schémas qui sont déjà clairs et suffisent à eux-mêmes (remplacer, si possible, des textes difficiles, abstraits par des schémas)
- Catalogue utilisateur et schéma de permissions clairement spécifiés (droits utilisateurs et privilèges)
- Utilisation de présentation structurée
- Utilisation d'un langage simple, clair, précis et sans ambiguïté

7 Analyse des Exigences (K2)	140 minutes

Objectifs d'Apprentissage (OA) pour le niveau Fondation de l'Ingénierie des Exigences

Les objectifs identifient ce que vous êtes capable de faire, en ayant suivi chaque module.

7.1 Exigences et solutions (K1)

OA-7.1.1 Rappeler l'objectif pour l'analyse des exigences (K1)

OA-7.1.2 Décrire la procédure à suivre pendant l'analyse des exigences (K2)

OA-7.1.3 Expliquer le concept de rupture structurelle entre les exigences et les solutions (K2)

7.2 Méthodes et Techniques (K2)

OA-7.2.1 Rappeler les différents modèles d'analyse des exigences (K1)

OA-7.2.1 Appliquer les différentes méthodes d'analyse pour des types spécifiques de modèles (K3)

7.3 Analyse orientée-objet (K2)

OA-7.3.1 Décrire les caractéristiques d'UML (K2)

OA-7.3.2 Décrire les caractéristiques de SysML (K2)

7.4 Estimations de coût (K2)

OA-7.4.1 Rappeler la raison de l'estimation de coût (K1)

OA-7.4.2 Reconnaître les facteurs importants pour l'estimation de coût (K2)

7.5 Priorisation (K2)

OA-7.5.1 Appliquer la procédure de priorisation pour un scénario donné (K3)

7.6 Accord sur les exigences (K2)

OA-7.6.1 Identifier ce qui doit être considéré lors de l'accord sur les exigences (K2)

7.1 Exigences et solutions (K1)	20 minutes

7.1.1 Objectif de l'analyse des exigences (K2)

Le but de l'analyse des exigences est de créer une solution pour l'implémentation des exigences. L'analyse des exigences prend en compte différentes parties prenantes de la solution, les conflits potentiels entre exigences, les analyses des relations et des dépendances entre les exigences, etc.

7.1.2 Procédure d'analyse des exigences (K2)

La procédure d'analyse des exigences comporte les étapes suivantes

- Analyse des besoins
- Description de la solution
- Estimation des coûts et priorisation

7.1.3 Rupture structurelle entre les exigences et les solutions (K2)

La rupture structurelle entre les exigences et les solutions signifie qu'il y a une différence entre les exigences et les solutions. Une solution est l'implémentation d'une exigence. Le modèle d'exigences peut être vu comme un modèle métier, présentant le problème à résoudre par le modèle de solution. Un modèle de solution est plus détaillé et inclut la spécification technique des exigences ; c'est une base pour le développement et le test.

Il existe trois niveaux de base des modèles

- Modèle des exigences
  - Souvent spécification non-formelle
  - Notations de modélisation métier (BPMN - Business Process Modeling Notation)
- Modèle de solution
  - Structures fonctionnelles (algorithmes, procédures, workflows)
- Modèle conceptuel
  - Spécifications techniques logiciel / matériel

Il devrait y avoir une traçabilité entre les modèles.

7.2 Méthodes et Techniques (K2)	20 minutes

Termes

Modèle de contexte, modèle de flot de données, modèle relation-entité, décomposition fonctionnelle, modèles conceptuels, modèles d'exigence, modèles de solution, modèle de transition d'états

7.2.1 Méthodes et modèles d'analyse (K2)

Les différents aspects d'un système sont représentés par des points de vue différents.

Les modèles sont développés grâce à des méthodes d'analyse appropriées.

Méthode d'analyse	Modèle d'analyse
Analyse du contexte	Modèle de contexte
Analyse de l'architecture	Décomposition fonctionnelle
Analyse du flot de données	Modèle de flot de données
Analyse des conditions	Modèle de conditions Réseau de Pétri
Analyse des décisions	Table de decisions
Analyse des données	Modèle sémantique de données Modèle relations-entités Dictionnaire de données

7.2.2 Types de modèles (K2)

Il y a trois types de modèles de base :

- Modèle d'exigences
  - Décrit le domaine du problème
  - Conçu dès les premières étapes du projet
  - Sert pour l'analyse des exigences et l'estimation des coûts
  - Fournit une base pour le modèle de solution

    - Exemple : modèle de cas d'utilisation métier, modèle de classes métier, modèle de processus métier
- Modèle de solution
    - Décrit le domaine de solution à partir de vues différentes du système
    - Conçu en même temps que le modèle d'exigences
    - Détermine la forme de l'implémentation des exigences fonctionnelles et non fonctionnelles
    - Fournit une base pour la conception système
    - Exemple : modèle de cas d'utilisation, diagramme de séquences, diagramme d'activités, diagramme de transitions d'états
- Modèle conceptuel
    - Spécifications techniques logiciel / matériel (modules, composants matériels, caractéristiques des PCs)

7.2.3 Différentes perspectives du système (K2)

Certains des points de vue en relation avec le système sont les suivants

- Vue logique
    - Exigences fonctionnelles
- Vue processus
    - Communication
    - Interaction
    - Exigences non fonctionnelles
- Vue implémentation
    - Composants (modules)
- Vue de l'installation
    - Intégration
    - Architecture système

7.2.4 Différents modèles (K1)

Les modèles suivants peuvent être utilisés

- Modèle de contexte
    - Description statique du système
    - Exprime l'architecture de base
- Décomposition fonctionnelle
    - Description statique du système
    - Exprime les décompositions successives du système

- Modèle de flot de données
  - Description dynamique du système
  - Représentation graphique des flots de données au sein d'un système
  - Ne fournit pas d'information sur les temps processus et si les processus fonctionnent en séquentiel ou en parallèle
- Modèle de transitions d'états
  - Description dynamique du système
  - Représente le comportement du système en une série d'événements, qui pourraient faire passer le système dans un ou plusieurs états possibles
- Modèle Entités – Relations
  - Représentation conceptuelle et abstraite des données
  - Contient des blocs de construction : entités, relations et attributs

Quand un modèle spécifique devrait-il être appliqué ? Différents modèles répondent à d'autres types de questions concernant la solution :

- Modèle Contexte ➔ QUOI (représentation des principaux flux entre le système et l'extérieur)
- Décomposition fonctionnelle ➔ QUOI (quelles sont les fonctions et les caractéristiques dans le périmètre du système ?)
- Modèle de flot de données ➔ QUOI (flux entre processus métier / fonctions / activités)
- Modèle Entités Relations ➔ QUOI (quelles relations existent entre les entités spécifiques (objets) du système ?)
- Modèle états – transitions ➔ POURQUOI (cause et effet)

7.3 Analyse orientée-objet (K2)	30 minutes

Termes

Diagramme d'activité, diagramme de comportement, diagramme de classe, diagramme de communication, diagramme de composants, diagramme de structure composite, diagramme de déploiement, diagramme d'interaction, diagramme objet, analyse orientée-objet, diagramme de package, diagramme d'exigences, diagramme de séquence, diagramme de machine à états, diagramme structurel, SysML, UML, diagramme des cas d'utilisation

7.3.1 UML (K1)

Unified Modeling Language est une notation unifiée pour l'analyse et la conception de systèmes. Elle contient différents types de diagrammes pour les points de vue différents sur le système (diagrammes de structure ou de comportement).

7.3.1.1 Diagramme de comportement (K2)

Les diagrammes de comportement montrent les caractéristiques comportementales d'un système ou d'un processus métier.

Ces diagrammes comprennent les types de diagrammes suivants

- Diagrammes d'activité
  - Modèle des comportements d'un système, et la manière par laquelle ces comportements sont liés dans un flux global du système
- Diagrammes de cas d'utilisation
  - Capture des cas d'utilisation et des relations entre les acteurs et le système
  - Description des exigences fonctionnelles du système, de la manière dont les opérateurs externes interagissent à la limite du système, et les réponses du système
- Diagrammes de machines d'état
  - Montrer comment un élément peut bouger entre les états, en classant son comportement en fonction des déclenchements de transitions et contraintes de gardes
- Diagrammes de temps (chronogramme)
  - Définir le comportement des différents objets au sein d'une échelle de temps
  - Fournir une représentation visuelle des objets changeant d'états et interagissant dans le temps

- Diagrammes de séquences
  - Représentation structurée du comportement comme une série d'étapes séquentielles dans le temps
  - Utilisés pour décrire le flux de travail, le passage de messages et comment les éléments, en général, coopèrent dans le temps pour atteindre un résultat
- Diagrammes de communication
  - Montrer les interactions entre les éléments lors de l'exécution, la visualisation des relations inter-objets
- Diagrammes des contextes d'interaction
  - Visualisation de la coopération entre d'autres diagrammes d'interactions pour illustrer un flux de contrôle servant un but à atteindre

7.3.1.2 Diagramme structurel (K2)

Les diagrammes structurels représentent les éléments structurels qui composent un système ou une fonction. Ces diagrammes reflètent des relations statiques de la structure, tels que les diagrammes de classes ou de paquetages, ou des architectures à l'exécution telles que des diagrammes de structures d'objets ou composites.

Les diagrammes structurels incluent les types de diagrammes suivants

- Diagrammes de classes
  - Capture de la structure logique du système, les classes et les objets créant le modèle, décrivant ce qui existe et les attributs et comportements associés
- Diagrammes de structure composite
  - Réfléchir à la collaboration interne des classes, des interfaces et des composants (et leurs propriétés) pour décrire une fonctionnalité
- Diagrammes de composants
  - Présenter les morceaux de logiciel en créant un système ainsi que leur organisation et leurs dépendances
- Diagrammes de déploiement
  - Décrire l'architecture d'exécution du système
- Diagrammes d'objets
  - Présenter les instances d'objets de classes et leurs relations à un instant donné
- Diagrammes de paquetages
  - Représenter l'organisation des éléments du modèle dans des paquetages et leurs dépendances

7.3.2 SysML (K2)

Modeling Language System est un langage de modélisation spécial pour l'Ingénierie Système. C'est une extension d'UML 2.1.

SysML offre quelques améliorations par rapport à UML. Ces améliorations sont les suivantes :

- Sémantiques plus souple et expressive
  - Réduit les restrictions UML centrées sur le logiciel et ajoute deux nouveaux types de diagrammes, diagramme des exigences et des paramètres
  - Permet la modélisation d'une large gamme de systèmes, qui comprennent du matériel, du logiciel, de l'information, des processus, du personnel et des installations.
- Facile à apprendre et à appliquer
  - Plus petit qu'UML (n'utilise pas beaucoup de constructions UML centrées sur le logiciel)
- Support des modèles et des vues
  - Les modèles et les vues sont architecturalement alignés sur le standard IEEE-Std-1471-2000
- Réutilisation des sept diagrammes UML et fourniture de deux nouveaux diagrammes (diagrammes d'exigences et de paramètres) pour un total de neuf types de diagrammes
  - Diagrammes d'exigences pour capturer les exigences fonctionnelles, de performance et d'interface
  - Diagrammes de paramètres pour définir des contraintes de performance et quantitatives

7.4 Estimations de coût (K2)	20 minutes

Contexte

Les estimations de coût permettent de relier l'Ingénierie des Exigences à la Gestion de Projet.

7.4.1 Types d'estimation (K2)

Les aspects les plus sujets à estimation sont

- Les coûts
- Le temps
- Les exigences

Les estimations de coûts permettent de reconnaître les coûts de développement, de changement, etc.

7.4.2 Influence sur les coûts de développement (K2)

Le coût du projet dépend de divers facteurs

- Type de projet
- Maturité des processus
- Méthodes et outils de conception et de test
- Technologie
- Complexité de la solution envisagée
- Objectifs qualité (par exemple niveau souhaité de qualité du logiciel)
- Qualifications de l'équipe
- Distribution de l'équipe
- Expériences

L'exactitude des estimations de coûts dépend de l'état d'avancement du projet et de sa maturité.

7.4.3 Les approches d'estimations de coût

L'estimation des coûts peut être réalisée en utilisant

- Déduction analogique
- Procédure algorithmique

7.4.3.1 Déduction par analogie (K2)

L'estimation des coûts est basée sur la comparaison des coûts de projets similaires. Elle est basée sur l'expérience, et non sur des formules mathématiques. Avec cette technique, le projet actuel est comparé à des projets antérieurs. La comparaison peut comprendre

- Le nombre d'exigences
- Le périmètre de la solution
- La technologie utilisée
- Les caractéristiques du personnel (compétences, expérience)

Sur la base des résultats de cette comparaison, une estimation peut être créée.

Les procédures d'estimation sont toujours basées sur des données historiques et des conditions connues.

Méthode Delphi

C'est une technique de communication structurée utilisée pour effectuer des prévisions interactives. Cela implique un panel d'experts [Linstone75].

Les experts sont invités à répondre à des questionnaires en un certain nombre de tours. Après chaque tour, il y a un résumé anonyme des prévisions des experts ainsi que les raisons des jugements fournis. Puis les experts révisent leurs réponses précédentes en prenant en considération les réponses des autres membres de leur panel.

On croit que pendant ce processus, le panel des réponses va diminuer et que le groupe va converger vers la «bonne» réponse.

Le processus s'arrête sur un critère d'arrêt prédéfini. Les scores moyens de la phase finale déterminent les résultats.

Estimations agiles

Dans les projets Agiles souvent gérés en Scrum, il ya une méthode d'estimation appelée « Planning game » ou « Planning poker ». Cette méthode est utilisée pour obtenir un consensus dans l'équipe. La capacité de l'équipe est alors mesurée à travers quelque chose qui s'appelle le taux de « Burn Down » et améliorée via les sessions de rétrospective menées après chaque sprint, où les chiffres prévus sont comparés aux chiffres actuels. Cela permet d'améliorer la capacité de l'équipe à estimer.

7.4.3.2 Procédure algorithmique (K2)

Dans cette approche, les coûts sont calculés sur la base de paramètres. Les paramètres peuvent décrire le produit (volume, durée), les conditions aux limites (efficacité), etc.

Les méthodes suivantes peuvent être utilisées :

- Formule de Putnam (équation)
- Points de fonction
- Modèle des coûts constructifs (COCOMO)

Équation de Putnam [Putnam91]

Effort = [Taille / Productivité * Temps $^{4/3}$] ^{3} * B

Où :

- Taille - la taille du produit (estimée par n'importe quelle estimation de taille utilisée par une organisation). Putnam utilise ESLOC (Effective Source Lines Of Code)
- B - facteur d'échelle, fonction de la taille du projet
- Productivité - la productivité des processus, la capacité d'une organisation logiciel particulière de produire des logiciels d'une taille donnée, à un taux de défaut particulier
- Temps - le calendrier global du projet en années
- Effort - effort total appliqué au projet en hommes.années

Points de fonction

Un point de fonction est une unité de mesure pour exprimer la quantité de fonctionnalités métier fournies par un système d'information à un utilisateur. Le coût d'un point de fonction unique est calculé sur la base de projets antérieurs.

Il y a cinq « fonctions »standards à prendre en compte comme points de fonction.

- Fonctions de données

1. Fichiers de logique interne
    - Tables dans une base de données relationnelle
    - Informations de contrôle de l'application
2. Fichiers d'interface externe
    - Tables dans une base de données relationnelle, les informations de contrôle de l'application ne sont pas maintenues par l'application en cours d'examen

- Fonctions transactionnelles

1. Entrées externes
    - Données d'entrée par utilisateur
    - Alimentations de données ou de fichiers par des applications externes
2. Sorties externes
    - Rapports créés par l'application, y compris des données dérivées
3. Enquêtes externes
    - Les rapports comptabilisés créés par l'application, où un rapport ne comporte pas de données dérivées

Les fonctions identifiées sont pondérées en fonction de trois niveaux de complexité : le nombre de points attribués à chaque niveau diffère selon le type de point de fonction.

Un exemple est montré ci-dessous :

Complexité	Points
Mineure	7
Moyenne	10
Élevée	15

Cocomo

Cocomo permet de calculer l'effort et le coût de développement logiciel en fonction de la taille du programme. La taille du logiciel est exprimée en EKLOC (estimé en milliers de lignes de code).

Cocomo s'applique à trois catégories de projets logiciels

- Les projets organiques
- Les projets semi-détachés
- Les projets embarqués

Les équations de base COCOMO sont :

- Effort appliqué (E) = ab (KLOC) bb [hommes.mois]
- Temps de développement (D) = cb (effort appliqué) db [mois]
- Personnel requis (P) = Effort appliqué / Temps de développement [nombre]

KLOC – nombre estimé de lignes livrées (en milliers) de code pour le projet

a_b, b_b, c_b et d_b :

Projet logiciel	a_b	b_b	c_b	d_b
Organique	2.4	1.05	2.5	0.38
Semi-détaché	3.0	1.12	2.5	0.35
Embarqué	3.6	1.20	2.5	0.32

7.5 Priorisation (K2)	20 minutes

Termes

Graduation, Graduation à 3 niveaux

7.5.1 Priorisation (K2)

La priorisation permet d'établir l'importance relative des exigences et l'implémentation pour répondre d'abord aux exigences les plus cruciales.

La priorisation supporte le développement incrémental, car elle permet de regrouper des exigences et d'établir des priorités d'implémentation.

7.5.2 Procédure de priorisation (K2)

La procédure pour définir les priorités des exigences comprend les activités suivantes :

1. Regroupement des exigences
   - Identifier les exigences qui s'influencent mutuellement et celles qui dépendent les unes des autres (par exemple : des exigences décrivant une fonctionnalité complexe)
2. Analyse des exigences
   - Analyse de toutes les parties prenantes impliquées afin de convenir du niveau d'importance
   - Analyse d'impact comme un moyen de supporter l'analyse des exigences
   - Établir les priorités des exigences
3. Création du plan projet des exigences
   - Établir un plan dans lequel les exigences avec des priorités élevées sont développées en premier
   - Attribution des responsabilités (qui met en œuvre l'exigence ?)
4. Planification des tests des incréments système
   - Concevoir des cas de test pour tester chaque incrément système (établi sur la base des exigences prioritaires) sur la base des priorités des exigences

7.5.3 Échelle de priorisation (K2)

Une approche commune des priorisations consiste à regrouper les exigences dans des catégories prioritaires. Habituellement, une échelle à trois niveaux est utilisée (par exemple : élevé, moyen et faible).

Comme ces niveaux sont subjectifs et imprécis, toutes les parties prenantes concernées doivent s'entendre sur la signification de chaque niveau de l'échelle qu'ils utilisent. La définition de la priorité doit être clairement précisée et devrait être un des attributs clé de chaque exigence.

Exemples d'échelles à trois niveaux :

Nom	Description
Haut	Exigence critique, exigée dans la première livraison du logiciel
Moyen	Supporte nécessairement les opérations système mais peut attendre une livraison future si nécessaire
Bas	Une amélioration fonctionnelle ou qualité, du type « ça serait bien de l'avoir »

Nom	Description
Essentiel	Le produit n'est pas acceptable tant que l'exigence n'est pas remplie
Conditionnel	Permet d'améliorer le produit, mais le produit n'est pas inacceptable si absent
Optionnel	Les fonctions qui peuvent ou non être utiles

7.6 Accord sur les exigences (K2)	20 minutes

Termes

Signature

7.6.1 Accord (K2)

S'entendre sur les exigences, souvent appelé « engagement sur les exigences » est un accord formel précisant que le contenu et le périmètre des exigences sont précis et complets.

L'entente officielle est une base pour le projet. Un accord sur les exigences de haut niveau (exigences métier) doit être fait avant le démarrage du projet. Les exigences détaillées (exigences fonctionnelles et non fonctionnelles) devraient être acceptées et signées avant de passer à la phase d'implémentation.

L'obtention de l'engagement sur les exigences est habituellement la tâche finale de l'analyse des exigences et de la conception.

L'engagement sur les exigences devrait être fait par les parties prenantes du projet, comprenant :

- Chefs de projets à la fois côté client et fournisseur
- Représentant de l'activité du client
- Analystes métier et système
- Ingénieurs des exigences
- Représentants de l'Assurance Qualité, équipes de développement et de test.

La liste des exigences doit être contraignante à la fois pour le côté client et le côté vendeur.

Un des objectifs de l'engagement sur les exigences est d'assurer que les exigences soient stables et que les modifications soient gérées via les demandes de changement formelles. Par conséquent, un accord formel réduit le risque d'introduction de nouvelles exigences pendant ou après l'implémentation.

S'entendre sur les exigences est considéré comme acquis, lorsque les parties prenantes pertinentes de tous les projets ont signé le document des exigences.

La finalisation de l'engagement sur les exigences devrait être communiquée à l'équipe du projet et c'est généralement une étape du projet.

7.6.2 Avantages de l'accord sur les exigences (K2)

- L'accord assure le développement du bon produit (travaux basés sur la liste convenue d'exigences et couvre uniquement ce qui est nécessaire pour obtenir une valeur ajoutée pour les parties prenantes)
- Réduction des risques de malentendus entre le client et le vendeur concernant le périmètre du projet
- Base pour les futurs travaux de conception

8 Suivi des Exigences (K2)	60 minutes

Objectifs d'Apprentissage (OA) pour le niveau Fondation de l'Ingénierie des Exigences

Les objectifs identifient ce que vous êtes capable de faire, en ayant suivi chaque module.

8.1 Suivi au sein du projet (K2)

OA-8.1.1 Rappeler les moyens de traçabilité (K1)

OA-8.1.2 Rappeler les raisons typiques pour des changements d'exigences (K1)

OA-8.1.3 Décrire les objectifs de traçabilité (K2)

OA-8.1.4 Reconnaître les différentes sortes de traçabilité (K1)

8.2 Gestion du changement (K2)

OA-8.2.1 Décrire les caractéristiques de gestion du changement (K2)

OA-8.2.1 Rappeler la constitution du comité de contrôle du changement (K1)

8.1 Suivi au sein du projet (K2)	20 minutes

Termes

Suivi horizontal et vertical

8.1.1 Évolution des exigences (K1)

Les exigences ne sont pas stables, mais continuent d'être développées au cours du cycle de vie du projet.

Les raisons d'un développement continu et de changements proposés peuvent être les suivantes

- Nouvelles idées
- Nouveaux besoins clients (résultant, par exemple, d'une nouvelle règlementation, de nouvelles orientations business, de nouveaux produits, etc.)
- Poursuite des travaux (par exemple, prochaine phase du projet, amélioration et optimisation des fonctionnalités déjà implémentées, etc.)
- Nouvelles connexions au sein du projet (par exemple, intégration avec de nouveaux systèmes, nouveaux canaux de communication comme internet ou de nouveaux canaux mobiles, etc.)

8.1.2 Traçabilité (K2)

La traçabilité fournit une solution de gestion du développement des exigences et des autres artefacts liés à ces exigences.

La traçabilité permet de vérifier que toutes les étapes importantes du processus de développement ont été réalisées. Cela devrait être implémenté de façon bidirectionnelle pour tous les artefacts (par exemple : des exigences vers les artefacts de conception et des artefacts de conception vers les exigences). La traçabilité est également importante pour les tests, la vérification et la validation.

Objectifs de la traçabilité :

- Analyse d'impact
- Analyse de couverture
- Preuve d'implémentation

- Utilisation de l'exigence (exigences tracées comme une preuve que les exigences sont utilisées et comment elles sont utilisées)

Afin d'assurer une bonne traçabilité, il est important d'identifier les exigences de façon unique.

8.1.3 Traçabilité (K2)

- Traçabilité horizontale
- Présente les dépendances entre les exigences d'un même niveau (relations entre les différents types d'exigences)
- Traçabilité verticale
- Présente les dépendances entre les différents artefacts (les exigences, la solution, la spécification de performance, les cas de test, le code, les modules, les plans, etc.)

8.2 Gestion du changement (K2)	20 minutes

Termes

Changement, commission de gestion du changement, gestion du changement, demande de changement

8.2.1 Changements des exigences (K1)

Les changements des exigences peuvent être demandés à n'importe quel moment pendant la réalisation du projet et après la sortie du logiciel final dans son environnement de production. Des changements arriveront toujours et il est important de planifier les changements par rapport au processus et au temps.

Les sources de changement peuvent être les suivantes :

- Extension des fonctionnalités existantes
- Défauts trouvés dans le logiciel / la documentation
- Autres choses trouvées, par exemple dans les tests telle que une mauvaise performance, etc.
- Demande d'une nouvelle fonctionnalité ou d'une modification d'une fonctionnalité existante
- Changements résultant de facteurs externes (changements organisationnels, modifications réglementaires)

8.2.2 Gestion du changement (K2)

Le processus de gestion du changement est le processus de demande, d'étude de faisabilité, de planification, d'implémentation et d'évaluation des changements dans un système logiciel, des documents ou autres produits du projet. L'objectif de la gestion du changement est de permettre et de supporter le traitement des changements et d'assurer la traçabilité des changements.

Le processus de gestion des changements comprend les activités suivantes

- Identification des changements potentiels
- Demande d'une nouvelle fonctionnalité
- Analyse de la demande de changement

- Évaluation du changement
- Planification du changement
- Implémentation du changement
- Revue et clôture du changement
- Déploiement du changement

Selon sa complexité et son impact, un changement peut avoir des impacts différents sur le système. De petits changements peuvent nécessiter des modifications mineures, tandis que des changements complexes peuvent changer radicalement la logique du système. Tout changement doit être soigneusement analysé afin d'établir les risques relatifs et d'évaluer la valeur de la modification par rapport aux risques prévus.

8.2.3 Demande de changement (K2)

Un changement doit être soumis via un document de demande de changement formelle (appelé aussi Request For Change (RFC)). Habituellement un tel document décrit la raison du changement, la priorité et la solution demandée ainsi que des détails supplémentaires comme :

- Le nom de la personne / du département ou d'une autre entité demandant le changement
- La date de soumission
- La date prévue d'implémentation de la modification (si applicable)
- Le coût du changement

8.2.4 Commission de Gestion des Changement (K2)

Les changements sont contrôlés et décidés par une Commission de Contrôle des Changements (CCB Change Control Board). L'introduction d'un CCB supporte d'une façon contrôlée l'implémentation d'un changement ou d'une modification proposée, pour un produit ou service.

La Commission de Contrôle des Changements est un comité qui repose sur des informations fournies (comme le risque lié à un changement, son impact, l'effort requis pour l'implémenter) qui prend des décisions quant à la réalisation ou non des modifications proposées. Le CCB est constitué de parties prenantes du projet ou de leurs représentants.

La Commission de Contrôle des Changements peut comprendre les rôles suivants :

- Gestion de projet
- Gestion du Développement
- Assurance qualité (gestion de la qualité, gestion des tests)

- Gestion business, le cas échéant
- Client, le cas échéant

8.2.5 Cycle de vie d'une exigence (K2)

Selon le niveau d'analyse et / ou d'implémentation des exigences, différents états sont assignés à une exigence. Le cycle de vie d'une exigence peut être exprimé en utilisant les états suivants :

- Nouveau (proposé)
- Pour revue
- Approuvé
- Conflictuel
- Implémenté
- Mis à jour
- Supprimé
- Testé
- Déployé

Différentes approches et organisations peuvent utiliser différents cycles de vie des exigences (et des états différents). Dans de nombreux cas de cycles de vie des exigences, les demandes de changement et les défauts sont très similaires et gérés à l'aide du même outil.

8.2.6 Distinction entre Gestion des Défauts et Gestion du changement (K2)

Il est important de faire la différence entre un changement et un défaut. Un défaut est défini comme une faille dans un composant ou un système qui peut empêcher un composant ou un système de remplir une fonction requise. Il s'agit d'une déviation par rapport à un état requis du système. Un changement est la modification de fonctionnalités existantes, d'exigences ou de fonctions, ou la création de nouvelles.

8.2.7 Impact d'un changement sur le projet (K2)

Les changements dans les exigences peuvent avoir des impacts différents sur le projet. Les changements les plus communs sont :

- Changement de calendrier, de budget, de ressources
- Travaux résultant de changements (en fonction de la phase du projet)

- Mise à jour de l'analyse et de la conception des artefacts (par exemple : les spécifications)
- Mise à jour de la documentation technique et utilisateur
- Changements dans la stratégie de test et les tests
- Mise à jour du plan de test
- Mise à jour des besoins en formation / plan
- Extension / raccourcissement du périmètre des travaux de programmation
- Changement du périmètre d'exécution et de la préparation des tests

9 Assurance Qualité (K2)	30 minutes

Objectifs d'Apprentissage (OA) pour le niveau Fondation de l'Ingénierie des Exigences

Les objectifs identifient ce que vous êtes en mesure de faire, en ayant suivi chaque module.

9.1 Facteurs d'influence (K1)

OA-9.1.1 Rappeler les facteurs qui influencent l'Ingénierie des Exigences (K1)

9.2 Vérification des exigences à l'étape d'élucidation des exigences (K2)

9.3 Assurance qualité via la testabilité (K2)

OA-9.3.1 Expliquer comment les produits de l'Ingénierie des Exigences supportent les tests (K2)

OA-9.3.2 Décrire l'utilité du critère d'acceptation (K2)

OA-9.3.3 Décrire dans quelle mesure l'Ingénierie des Exigences peut contribuer aux tests (K2)

9.4 Métriques (K2)

OA-9.4.1 Rappeler la définition de métriques (K1)

OA-9.4.2 Décrire quelles métriques peuvent être utilisées pour l'Ingénierie des Exigences (K2)

9.1 Les facteurs d'influence (K1)	10 minutes

9.1.1 Influences sur l'Ingénierie des Exigences (K1)

La qualité des produits issus de l'Ingénierie des Exigences dépend des facteurs suivants :

- Le produit en cours de développement
- L'environnement dans lequel il est produit
- Le domaine (complexité du domaine métier, le niveau d'innovation, la fréquence des modifications dans le métier, etc.)
- Les facteurs légaux, environnementaux et sécuritaires
- La pression par rapport au temps et au coût (les contraintes temps et coût peuvent diminuer les possibilités d'exécuter les processus de l'Ingénierie des Exigences)
- Les facteurs culturels (langue, éducation, etc.)
- Les contraintes technologiques et de conception

De tels facteurs doivent être pris en considération lors de la planification des activités d'assurance qualité.

9.2 Vérification des exigences à l'étape d'élucidation des exigences (K2)	20 minutes

La vérification doit être faite de façon continue durant le développement de la solution. Afin de vérifier les exigences aux étapes d'élucidation, les techniques suivantes peuvent être utilisées :

- Revues techniques
- Simulations
- Présentations
- Démonstrations

Il est important de planifier la vérification dès le début d'un projet.

9.3 L'assurance qualité via la testabilité (K2)	20 minutes

Termes

Critère d'acceptation, testabilité

9.3.1 L'Ingénierie des Exigences et le test (K2)

L'ingénierie des Exigences est fortement liée au test. Des cas de test satisfaisants nécessitent des exigences parfaitement définies, et qui peuvent être testées. L'implication des testeurs dans les spécifications est donc primordiale.

9.3.2 Le critère d'acceptation (K2)

Selon la nomenclature Prince2, le critère d'acceptation, également appelé critère de succès, représente la norme requise pour satisfaire les attentes qualité du client et obtenir son adhésion au produit fini. En d'autres termes, ce sont des critères émis sur des fonctions ou composants spécifiques, ou tout autre élément du produit logiciel ou du projet qui doivent être remplis afin de satisfaire le client et gagner son adhésion.

Le critère d'acceptation doit être convenu entre les deux parties (le vendeur et le client) avant le début du projet (cela devrait constituer une clause contractuelle) et servir de base au Plan de Qualité du Projet. Chaque exigence définie doit aboutir à au moins un critère d'acceptation. Ces critères forment la base du test d'acceptation.

Chacun d'entre eux doit être quantifiable, et sa méthode de mesure doit être réaliste et convenue.

Exemple de critère d'acceptation : l'application doit répondre en moins d'une seconde, et délivrer un message d'attente dans le cas contraire.

9.3.3 Méthodes de test (K2)

Les produits de l'Ingénierie des Exigences supportent le test en fournissant ce qui est appelé les « bases de tests ». Les exigences et leurs spécifications peuvent servir de bases de tests.

Les produits de l'Ingénierie des Exigences soutiennent les tests :

- En acceptant la couverture fonctionnelle (couverture de toutes les exigences fonctionnelles par les cas de test, et ces cas de test sont écrits pour couvrir la totalité des exigences fonctionnelles)

  Couverture fonctionnelle = ∑ tests des exigences via des cas de test / ∑ les exigences

- En fournissant une base de test fiable pour la boite-noire ou les tests basés sur les spécifications, tels que :

Valeurs limites pour les catégories de données d'entrée

États d'application

Contraintes logiques et économiques

Etc.

- En acceptant les techniques de test telles que : partition d'équivalence, analyse des valeurs limites, table de décision, test de transition d'état, test de cas d'utilisation.

  Par exemple, la partition d'équivalence est une technique de test qui divise les données d'entrée utilisées dans un module spécifique du logiciel en partitions de données, à partir desquelles il est possible d'exécuter des cas de test. Ces derniers sont développés pour couvrir au moins une fois chaque partition.

Exemple de partition d'équivalence :

1 La plage valide concernant l'âge de l'utilisateur est de 1 à 99. Cette amplitude est appelée partition valide. Il y a donc deux autres partitions de plage invalide : la première sera ≤ 0 et la seconde sera ≥ 100.
2 La plage valide de réponses à un questionnaire comprend les caractères suivants : A, B, C et D. La partition invalide contiendra tout autre caractère de texte, y compris les caractères spéciaux.

Les cas de test doivent prendre en compte à la fois les plages valides et invalides.

D'autres techniques de test peuvent également être utilisées, comme l'analyse des valeurs limites, les tests de transition d'état, les tests de cas d'utilisation, etc. Elles doivent être employées après une analyse de tests des exigences et appliquées selon le contenu des exigences.

9.3.4 Exigences et processus de test (K2)

Les exigences sont les informations d'entrée de base au processus de développement et de test du système. Clairement définies, elles réduisent le risque d'échec du projet (ou même du produit) en permettant des tests rigoureux. La stabilité de ces exigences entraine des délais sur des tâches spécifiques.

Il est important de rappeler que les exigences doivent être validées par des tests statiques (ce qui inclus les testeurs) et acceptées par les responsables de test (en effet, dans certains cas les exigences ne seront pas considérées comme testables). Les testeurs jouent un rôle prépondérant dans l'amélioration de la qualité des exigences en remontant les points faibles et les possibles imperfections. Ils participent également aux revues des exigences afin de garantir leur testabilité.

9.4 Les Métriques (K2)	20 minutes

Termes

Métrique

9.4.1 La métrique (K1)

La métrique est une échelle de mesure et une méthode utilisée pour le métrage (ISO 14598).

Les métriques permettent d'émettre un avis quantifiable sur le statut du projet et sa qualité.

Il est important de rappeler que les résultats du métrage (chiffres collectés pendant la mesure) doivent toujours être comparés aux données de référence (mesure correspondante).

9.4.2 Les métriques dans le contexte des exigences (K1)

Les métriques suivantes peuvent être appliquées aux exigences :

- Le coût du projet
  - Nombre des exigences
- Le suivi du projet
  - Nombre des exigences reportés sur les autres objets
- La stabilité du projet
  - Nombre des exigences modifiables
- L'évolution du processus
  - Raisons des modifications concernant les exigences (défauts, améliorations)
- La qualité des spécifications
  - Couverture des exigences par les modèles
- Le nombre des défauts
  - Type de défaut
  - Nombre de fautes dans les exigences pour chaque type (logique, cohérence, donnée, défaut, etc.)

9.4.3 Les mesures de la qualité des exigences (K2)

Quelques questions permettent d'évaluer la qualité des exigences :

- Les exigences sont-elles exactes ?
- Sont-elles compréhensibles ?
- Sont-elles réalisables ?
- Sont-elles traçables ?
- Sont-elles identifiables ?
- Sont-elles testables ?

Une formule possible pour mesurer la qualité des exigences :

Clarté = ∑ Exigences sans défauts ni problèmes détectés / ∑ Exigences

Il est également possible de mesurer le rapport de variation (celui-ci ne s'applique pas aux projets Agile). Plus le rapport de variation du lot complet des exigences est élevé (cela peut être la cause d'efforts apportés à la clarification d'exigences incomplètes, incohérentes ou incompréhensibles), plus le projet est risqué. Aussi, ce taux doit être mesuré pour contrôler les risques d'un projet.

10 Outils (K2)	40 minutes

Objectifs d'Apprentissage (OA) pour le niveau Fondation de l'Ingénierie des Exigences

Les objectifs identifient ce que vous êtes en mesure de faire, en ayant suivi chaque module.

10.1 Avantage des outils (K2)

OA-10.1.1 Expliquer l'objectif du support des outils dans l'Ingénierie des Exigences (K2)

OA-10.1.2 Décrire quelles activités peuvent être supportées par les outils dans l'Ingénierie des Exigences (K2)

10.2 Catégories des outils (K2)

OA-10.2.1 Quelles sont les exigences sur les outils utilisés pour l'Ingénierie des Exigences ? (K2)

OA-10.2.2 Qu'est-ce qui doit être pris en compte dans le coût des outils ? (K2)

10.1 Avantages des outils (K2)	20 minutes

Termes

Outils de l'Ingénierie des Exigences

10.1.1 Utilisation des outils dans l'Ingénierie des Exigences (K2)

Les outils de stockage et d'administration des exigences facilitent l'Ingénierie des Exigences. Ils prennent en charge les activités répétitives et mécaniques, et assurent une vue d'ensemble. Il est donc possible de maintenir des documents difficiles cohérents et à jour. La sélection d'outils doit s'effectuer avant que le produit ne soit développé. Dans le cas contraire, cela occasionnerait des problèmes importants.

Les outils jouent un rôle dans les activités suivantes :

- Identification et stockage des exigences
- Modélisation des exigences (y compris les prototypes)
- Documentation des exigences (création des spécifications)
- Définition et maintien de la traçabilité des exigences

10.1.2 Avantages de ces outils (K2)

- Assurer que toutes les exigences sont stockées en un seul endroit et accessibles à tous les participants
- Maintenir la traçabilité des exigences (des cas de test, etc.) et permettre la vérification de la couverture des exigences correspondantes
- Permettre une gestion facilitée concernant la modification des exigences
- Améliorer la qualité des spécifications des exigences en imposant l'usage de modèles de documents définis et de notation.
- Gagner du temps grâce à l'automatisation de certaines activités (telle que générer l'ensemble des spécifications depuis un des outils)

10.2 Catégorie des outils (K2)	20 minutes

Termes

Catégories d'outils

10.2.1 Catégories des outils (K2)

- Outils d'incitation aux exigences
  - Carte heuristique
- Outils de modélisation
  - Outils UML
  - Outils SysML
- Outils de prototypes
- Outils de gestion des exigences
- Outils de gestion des anomalies
- Outils de gestion des modifications
- Outils de gestion de projet

Le coût d'acquisition de ces outils varie amplement. Les outils commerciaux peuvent être très onéreux alors que ceux qui sont « open source » sont gratuits. La décision de choisir tel ou tel outil doit donc être murement réfléchie. Avant la sélection, une analyse doit être réalisée. Elle doit prendre en compte :

- Le modèle de notation utilisé dans l'entreprise et qui doit être supporté par l'outil
- S'il est prévu d'utiliser d'autres modèles de notation dans le futur (dans ce cas, il serait raisonnable d'acquérir un outil supportant les deux types de modèles)
- Les exigences de l'entreprise concernant les fonctionnalités intégrées à l'outil (les outils commerciaux fournissent généralement plus de fonctions que ceux qui sont « open source »)
- Le cout de l'outil, en fonction de son utilisation (s'il sera utilisé pour un seul projet spécifique ou bien sur la plupart des projets)
- S'il est possible de l'intégrer à d'autres outils nécessaires (tels qu'un outil de gestion des erreurs, gestion de projet ou autre, en fonction des besoins de l'entreprise)
- S'il peut échanger des informations avec les outils utilisés par les clients de l'entreprise (parfois, le client mène sa propre analyse des exigences, les produits de l'analyse détaillée du vendeur devraient donc être migrés dans l'environnement du client).
- La facilité d'utilisation et de prise en main de l'outil (avec d'éventuels coûts de formation), la disponibilité de l'aide en ligne, des manuels, tutoriaux et autres supports.

Un choix trop hâtif peut entrainer des coûts importants :

Cout d'acquisition d'un outil qui ne satisfait pas les exigences des utilisateurs et qui ne remplit pas son objectif

- Cout d'acquisition d'un outil onéreux qui est utilisé sur un projet unique ou acquisition d'un outil onéreux disponible en open source avec des fonctions similaires.
- Cout de formation en cas d'acquisition d'un outil qui ne présente pas d'aide système suffisante (en particulier dans le cas où seules les fonctionnalités de base sont requises depuis l'outil)
- Cout d'extension de l'outil acquis avec des fonctionnalités additionnelles, requises par les utilisateurs et qui ne sont pas supportées par l'outil (alors qu'il y a d'autres outils incluant ces fonctionnalités)
- Cout de l'intégration de l'outil avec les autres outils utilisés dans l'entreprise.

11 Littérature

Beck, K.: *Extreme Programming*. Munich 2003

Beck, K.: Extreme Programming Explained: Embrace Change. Boston 2000

Beck, K.: Test Driven Development. By Example. Amsterdam 2002

Beck, K.: Refactoring: Improving the Design of Existing Code. Addison-Wesley Longman 1999

Boehm, B.: *Software Engineering Economics*. Englewoods Cliffs, NJ 1981

Bohner, S.A. and R.S. Arnold, Eds. *Software Change Impact Analysis*. Los Alamitos, California, USA, IEEE Computer Society Press 1996.

Bundschuh, M.; Fabry, A.: *Aufwandschätzung von IT-Projekten*. Bonn 2004

Cockburn, A.: Agile *Software Development*. Addison Wesley 2002

Cockburn, A.: *Writing Effective Use Cases*. Amsterdam 2000

Cohn M.: *Estimating With Use Case Points*, Fall 2005 issue of Methods & Tools

Cotterell, M. and Hughes, B.: *Software Project Management*, International Thomson Publishing 1995

Newman, W.M. and Lamming, M.G.: *Interactive System Design*, Harlow: Addison-Wesley 1995

Davis A. M.: *Operational Prototyping: A new Development Approach*. IEEE Software, September 1992. Page 71

Davis, A. M.: Just Enough Requirements Management. Where Software Development Meets Marketing, Dorset House, 2005, ISBN 0932633641

DeMarco, T. et al.: Adrenalin-Junkies und Formular-Zombies – Typisches Verhalten in Projekten. Munich 2007

DeMarco, T.: Controlling Software Projects: Management, Measurement and Estimates. Prentice Hall 1986

DeMarco, Tom: *The Deadline: A Novel About Project Management*. New York 1997

Dorfman, M. S.: Introduction to Risk Management and Insurance (9 ed.). Englewood Cliffs, N.J: Prentice Hall 2007. ISBN 0-13-224227-3.

Dynamic Systems Development Method Consortium. See: http://na.dsdm.org

Ebert, Ch.: Systematisches Requirements Management. Anforderungen ermitteln, spezifizieren, analysieren und verfolgen. Heidelberg 2005

Evans, E. J.: Domain-Driven Design: Tackling Complexity in the Heart of Software. Amsterdam 2003

Graham, D. et al: *Foundations of Software Testing*. London 2007

Gilb, T.; Graham, D.: *Software Inspection*. Reading, MA 1993

Gilb, T.: *What's Wrong with Requirements Specification.* See: www.gilb.com

Heumann, J.: *The Five Levels of Requirements Management Maturity,* see: http://www.ibm.com/developerworks/rational/library/content/RationalEdge/feb03/ManagementMaturity_TheRationalEdge_Feb2003.pdf

Linstone H. A., Turoff M.: *The Delphi Method: Techniques and Applications, Reading*, Mass.: Adison-Wesley, ISBN 9780201042948, 1975

Hull, E. et. All: *Requirements Engineering*. Oxford 2005

IEEE Standard 610.12-1990 IEEE Standard Glossary of Software Engineering Terminology

IEEE Standard 829-1998 IEEE Standard for Software Test Documentation

IEEE Standard 830-1998 IEEE Recommended Practice for Software Requirements Specifications

IEEE Standard 1012-2004: IEEE Standard for Software Verification and Validation

IEEE Standard 1059-1993: IEEE guide for software verification and validation plans

IEEE Standard 1220-1998: IEEE Standard for Application and Management of Systems Engineering Process

IEEE Standard 1233-1998 IEEE Guide for Developing System Requirements Specifications

IEEE Standard 1362-1998 IEEE Guide for Information Technology-System Definition – Concept of Operations (ConOps) Document

ISO 9000

ISO/EIC 25000

ISO 12207

ISO 15288

ISO 15504

ISO 31000: RiskManagement - Principles and Guidelines on Implementation

IEC 31010: Risk Management - Risk Assessment Techniques

ISO/IEC 73: Risk Management – Vocabulary

ISTQB: ISTQB Glossary of Testing Terms 2 1

ISTQB: Certified Tester, Foundation Level Syllabus, version 2011

Jacobsen, I. et al.: The Unified Software Development Process. Reading 1999

Jacobson, I.et al.: Object-Oriented Software Engineering. A Use Case Driven Approach. Addison-Wesley 1993

Kilpinen, M.S.: The Emergence of Change at the Systems Engineering and Software Design Interface: An Investigation of Impact Analysis. PhD Thesis. University of Cambridge. Cambridge, UK 2008.

Lauesen, S.: Software Requirements: Styles and Techniques. London 2002

Mangold, P.: IT-*Projektmanagement kompakt*. Munich 2004

McConnell, S.: *Aufwandschätzung für Softwareprojekte*. Unterschleißheim 2006

McConnell, S.: *Rapid Development: Taming Wild Software Schedules (1st ed.)*. Redmond, WA: Microsoft Press. ISBN 1-55615-900-5, 1996

Paulk, M., et al: The Capability Maturity Model: Guidelines for Improving the Software Process. Reading, MA 1995

Pfleeger, S. L.: Software Engineering: Theory and Practice, 2nd edition. Englewood Cliffs, NJ 2001

Pfleeger, S.L. and J.M. Atlee: *Software Engineering: Theory and Practice.* Upper Saddle River, New Jersey, USA, Prentice Hall 2006.

Pohl, K.: Requirements Engineering. Grundlagen, Prinzipien, Techniken. Heidelberg 2007

Project Management Institute: *A Guide to the Project Management Body of Knowledge* (PMBOK ® Guide). PMI 2004

Putnam, L.e H.; Ware M. *Measures for excellence: reliable software On time, within budget*. Yourdon Press. ISBN 0-135676-94-0, 1991

Robertson, S.; Robertson, J.: *Mastering the Requirements Process*, Harlow 1999

Rupp, C.: Requirements-Engineering und Management. Professionelle, Iterative Anforderungsanalyse in der Praxis. Munich 2007

Sharp H., Finkelstein A. and Galal G.: Stakeholder Identification in the Requirements Engineering Process, 1999

Sommerville, I.: *Requirements Engineering*. West Sussex 2004

Sommerville, I.: *Software Engineering 8*. Harlow 2007

Sommerville, I.; Sawyer, P.: Requirements Engineering: A Good Practice Guide. Chichester 1997

Sommerville, I.; Kotonya, G.: Requirements Engineering: Processes and Techniques. Chichester 1998

Spillner, A. et all: *Software Testing Foundations*. Santa Barbara, CA 2007

SWEBOK - The Guide to the Software Engineering Body of Knowledge: http://www.computer.org/portal/web/swebok/home

Thayer, R. H.; Dorfman, M.: Software Requirements Engineering, 2nd edition. Los Alamitos, CA 1997

V-Modell® XT: http://www.vmodellxt.de/

Wiegers, K.E.: *First Things First: Prioritizing Requirements.* Software Development, September 1999

Wiegers, K. E.: *Software Requirements*. Redmond 2005

Wiegers, K. E.: More About Software Requirements: Thorny Issues and Practical Advice. Redmond, Washington 2006

Young, R. R.: *Effective Requirements Practices*. Addison-Wesley 2001

12 Index