CHAPITRE 3

PHASE 1 - DÉFINITION DU CADRE CONCEPTUEL

La phase 1 de la recherche consiste à définir les principaux éléments du cadre conceptuel présenté au niveau de la description de la méthodologie de recherche. Les éléments en extrant de la phase 1 seront :

- Les concepts à la base du génie
- Les concepts du génie logiciel
- Les concepts de l'informatique comme principale discipline frontière
- Les activités du génie logiciel présentées par la norme ISO/IEC 12207
- Les définitions des termes concept et principe

3.1 Les concepts à la base du génie

Bourque et al. (2002) soulignent que les principes fondamentaux du génie logiciel devraient être un sous-ensemble spécifique de principes plus généraux du génie. De même, Moore (1998) souligne que: *« The principles of SE would be regarded as selected, adapted and specialized from principles of engineering in general » (p.5).*

Une avenue logique serait d'identifier les principes fondamentaux du génie, ce qui permettrait de baliser la démarche pour notre recherche. Cependant, on constate que peu d'auteurs ont écrit sur le génie en général et encore moins sur l'identification des principes fondamentaux du génie. La forte spécialisation des disciplines du génie pourrait être une des sources de cette carence.

Dans le but d'apporter certaines balises à notre cadre conceptuel de recherche, la présente section a pour objectif de définir le génie et son essence, de le situer par rapport à la science, à l'art et d'identifier les principaux concepts qui le caractérisent.

Ces concepts seront utiles pour la suite de cette recherche afin de pallier au manque de principes identifiés pour le génie. Pour atteindre ces objectifs, une synthèse a été faite des propos et de la vision de trois auteurs ayant contribués à décrire ce qu'est le génie et le travail de l'ingénieur (Aslaksen (1996), Davis (1998) et Rodgers (1983)).

3.1.1 Qu'est ce que le génie?

Rodgers (1983) définit le génie comme suit :

> *« Engineering refers to the practice of organising the design and the construction of any artifice which transform the physical world around us to meet some recognised need. » (p.51).*

Rodgers souligne que le terme « engineering » a la forme d'un verbe qui signifie une forme d'action. Cette action se concrétise dans le déroulement des activités de design et de construction d'un artéfact (un produit ou un service) en tenant compte de contraintes de nature économique, sociale et environnementale. Cette action requiert aussi la prise de multiples décisions et de compromis tout au long du processus d'ingénierie. Également, Rodgers affirme que *l'essence* même du génie est la *conception et la production d'artéfact.*

Davis (1998) cite la définition de l'ingénierie donnée par le National Research Council (NRC) :

> *« Business, government, academic, or individual efforts in which knowledge of mathematics and/or natural science is employed in research, development, design, manufacturing, system engineering, or technical operations with the objective of creating and/or delivering systems, products, process, and/or services of a technical nature and content intended for use » (p.33).*

Davis (1998) critique cette définition par le fait qu'elle contient une référence circulaire en comportant en elle-même le terme « engineering ». Cependant, il en extrait trois éléments qui caractérisent le génie :

1. Les mathématiques et les sciences naturelles sont au centre des activités du génie.
2. Le génie met l'accent sur la production d'objets plutôt que sur l'élaboration de théories.
3. Le génie ne vise pas à expliquer le monde, mais à le modifier.

Davis (1998) ajoute que le génie est principalement une façon d'organiser et de gérer les activités de design, de développement et de fabrication en s'assurant que tout sera fait dans les temps prévus, en respectant les budgets et à la satisfaction du client. Durant le déroulement du processus de génie, l'ingénieur doit tenir compte de considérations économiques, de sécurité, de fiabilité et de durabilité. Le processus d'ingénierie comporte un élément important de management des activités techniques.

Davis (1998) souligne que l'ingénieur sait organiser le travail technique des différentes activités du processus. De plus, l'ingénieur est en mesure de donner les directives requises pour la réalisation des activités et d'assurer la responsabilité du projet. Enfin, il doit vérifier les résultats obtenus et respecter le code d'éthique de la profession.

Aslaksen (1996) souligne que le génie a une signification différente selon les personnes. Le génie englobe un large spectre de connaissances et d'activités telles le design, la fabrication, la construction, l'entretien, la recherche et développement et le management. Aslaksen définit le génie à son origine comme étant *« The art of getting things done » (p.14).* Le génie serait une discipline basée sur la science et elle comporterait un élément essentiel de créativité. Aslaksen met l'accent sur le fait que le génie est avant tout un processus multidisciplinaire dont l'objectif est de combler un besoin exprimé par un produit ou par un service considéré comme satisfaisant dans le respect des contraintes

économiques, sociales, environnementales et légales. À cause de ces contraintes omniprésentes tout au long du processus, le génie produit des solutions issues de nombreux compromis. Aslaksen affirme que *l'essence* même du génie est un produit issu du cerveau humain qu'aucune machine ne peut élaborer à cause de la complexité des concepts impliqués.

Aslaksen propose que le génie repose sur deux composants majeurs tel qu'illustré à la figure 14.

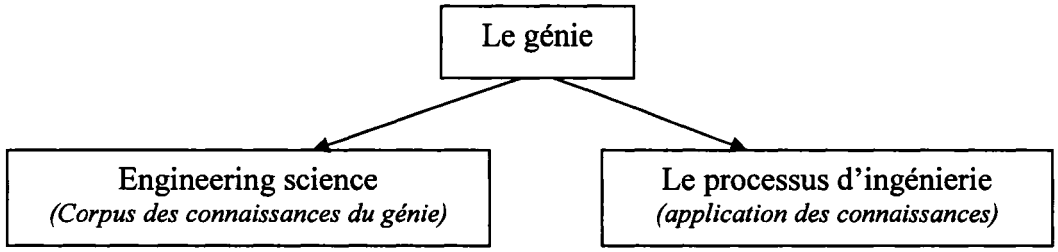

Figure 14 Composant à la base du génie (adaptation des propos de Aslaksen 1996)

Le « engineering science » est définit comme une technologie dont les bases proviennent de la physique et de la chimie (Aslaksen 1996). Rodgers (1983) ajoute la biologie et les mathématiques comme éléments de base du « engineering science ». En plus des connaissances propres au « engineering science », le génie nécessite des connaissances provenant de disciplines frontières telles l'économie, la finance, le marketing et la gestion des ressources humaines (Aslaksen 1996).

Aslaksen (1996) souligne que le génie est fondamentalement un *processus* qui, à partir d'un besoin exprimé, conçoit un produit pour le satisfaire. Ce processus multidisciplinaire joue un rôle intégrateur en intégrant tous les processus individuels en un processus unique et cohérent. Le processus d'ingénierie intègre les activités de développement, de design, de production et la gestion de ces activités. Aslaksen souligne

aussi que ce processus doit être conçu, décrit, paramétrisé et optimisé, et ce même s'il comporte une dimension de créativité. De plus, pour le bon déroulement du processus d'ingénierie, des connaissances spécialisées sont requises, ainsi que des méthodes, des techniques et des outils.

Aslaksen (1996) mentionne que le processus classique d'ingénierie débute après la définition des besoins et se termine par une évaluation du produit issu du processus en fonction des besoins d'origine, de ses performances et de son coût. Même si les besoins de la société sont exprimés à l'extérieur du processus d'ingénierie, l'auteur souligne l'importance que le processus tienne compte de la phase de l'expression des besoins et de son management. L'intégration de la phase des besoins au processus d'ingénierie fait partie des changements nécessaires dans l'évolution du génie.

3.1.2 Caractéristiques du processus d'ingénierie

Aslaksen (1996) souligne que le processus d'ingénierie ne peut se faire d'une façon isolée. Ce processus serait plutôt un composant central interagissant avec d'autres entités telles la recherche, le marketing, la finance et les services d'après vente. Le processus intègre aussi les étapes classiques de la résolution de problème : définition du problème, définition des critères de décisions, identification des solutions possibles et le choix d'une solution. De plus, Aslaksen affirme que la « tracabilité » serait une propriété importante du processus d'ingénierie et que sans celle-ci, le processus perdrait la qualité d'être rationnel. Le processus d'ingénierie serait aussi caractérisé par l'utilisation de méthodes, d'outils et de procédures dont l'efficacité peut être évaluée par la mesure.

Aslaksen (1996) présente trois propriétés fondamentales qui caractérisent le processus d'ingénierie : la complexité, l'incertitude et le risque.

L'auteur affirme que la *complexité* des tâches associées aux projets de génie a décuplé au fil des ans. Une des sources de croissance de la complexité identifiée par l'auteur serait la croissance importante des interactions entre les technologies impliquées dans les projets. De plus, les ingénieurs doivent tenir compte, dès la phase du design, des aspects reliés à l'entretien du produit. Ceci a pour conséquence d'accroître la complexité de la phase du design. Ainsi, le génie ne peut plus se contenter de concevoir un artéfact, mais il doit aussi prévoir l'entretien de cet artéfact et de sa disposition à la fin de sa vie utile.

Aslaksen (1996) affirme que le cerveau humain a des limites importantes pour faire face à la complexité croissante des projets. Ainsi, le cerveau aurait tendance à laisser de côté les détails pour se concentrer sur les principaux éléments. De ce fait, le processus d'ingénierie doit aider les ingénieurs à maîtriser la complexité en les aidant à formuler les différentes tâches à exécuter dans une séquence déterminée afin que le cerveau ne soit pas surchargé. L'auteur cite, à titre d'exemple, la méthode du partitionnement, bien connue dans l'ingénierie des systèmes. Cette méthode permet de diviser chacune des tâches complexes en plusieurs plus petites tâches dont chacune sera plus facilement traitée par le cerveau humain.

Étroitement liée à la complexité, *l'incertitude* serait une autre caractéristique du processus d'ingénierie. À partir du moment où le design doit anticiper les activités d'entretien et que la durée des projets s'étale sur plusieurs années, il est incontournable que des approximations de certaines variables soient faites. De plus, la taille des projets rend difficile le traitement en profondeur de toutes ces variables. Ainsi, le processus d'ingénierie doit composer avec des approximations et des moyennes statistiques. Dans certains projets, les données disponibles sont extraites de modèles à échelle réduite qui n'offrent pas l'exactitude du projet réel. Pour pallier à ces approximations, le génie s'est doté de marges de sécurité qui permettent d'absorber certaines variations entre les approximations et les données réelles (Rodgers (1983)). Aslaksen (1996) mentionne

aussi que le développement du « *statistical engineering* » pourrait fournir des outils afin de mieux affronter la complexité et l'incertitude des projets de génie.

Le *risque* est la troisième caractéristique fondamentale du processus d'ingénierie identifiée par Aslaksen. Le risque peut se manifester par des changements importants à tenir compte en cours de projet et ayant des impacts significatifs sur celui-ci. Aslaksen souligne que le risque est une propriété composite qui se définit comme la probabilité qu'un événement survienne avec ses effets négatifs sur le projet. Le processus d'ingénierie tente de minimiser les deux composants du risque soient la probabilité qu'un événement se manifeste et les impacts négatifs de l'événement sur le projet. Le processus d'ingénierie comporte un volet de gestion du risque. Ce volet tente d'identifier et de contrôler les événements ayant un potentiel d'effets négatifs sur le projet.

3.1.3 Le design : phase prédominante du processus d'ingénierie

Rodgers (1983) souligne qu'une des phases majeures dans le processus d'ingénierie est la phase du design. Le design comporte deux volets : le design industriel et le design d'ingénierie.

Le *design industriel* aborde les aspects cosmétiques et ergonomiques du produit. Les interactions personne-machine sont évaluées en considérant, entre autres, les aspects de sécurité et de santé.

Le *design d'ingénierie* est le volet majeur de la phase du design. Ce volet se concentre sur la conception d'un artéfact qui va répondre au besoin exprimé tout en étant efficace à un coût raisonnable. Il est à noter que l'objectif de la phase du design se rapproche de près de la définition donnée au génie.

Rodgers (1983) souligne que le design est une activité intellectuelle qui procède à l'élaboration et à l'évaluation des solutions possibles en tenant compte des contraintes d'ordre économique, légal et environnemental. Également, le jugement et l'expérience de l'ingénieur sont mis à contribution dans le design. Le design nécessite une série de décisions prises par l'ingénieur et des compromis doivent être souvent faits. L'ingénieur puisera des informations nécessaires à son évaluation dans les synthèses du corpus de connaissances du « engineering science ». Cependant, si l'ingénieur ne trouve pas ce dont il a besoin pour réaliser son design, il peut alors faire appel à des groupes d'ingénieurs-chercheurs du « engineering science » pour développer, par exemple, un nouveau matériau comportant les caractéristiques et les comportements recherchés. Ainsi, des activités de recherche sont possibles et nécessaires pour soutenir certains projets du génie.

Rodgers (1983) souligne que l'ingénieur peut choisir trois orientations possibles pour réaliser le design.

- Faire un design en utilisant les codes de pratiques éprouvés
- Modifier une solution existante et éprouvée, et l'adapter à un nouveau contexte ou à une échelle plus grande
- Remplacer une façon de faire éprouvée par un nouveau concept qui rendra désuète les solutions existantes

Vincenti (1990) regroupe les deux premières orientations sous le titre du *normal design*, alors que la troisième orientation est nommée *radical design*. À cause des aspects de sécurité, légaux et des conséquences catastrophiques d'un échec, le design sera imprégné d'un esprit favorisant l'évolution de solutions (design normal), plutôt que la recherche d'une solution révolutionnaire (design radical).

Au niveau du design, l'ingénieur doit tenir compte de plusieurs facteurs, parfois même contradictoires. De plus, il doit tenir compte des conseils et des recommandations de plusieurs spécialistes tant au niveau technologique qu'au niveau économique, environnemental et social. Ainsi, le processus d'ingénierie serait ponctué d'une suite de compromis entre l'idéal à atteindre et les diverses contraintes dont l'ingénieur doit tenir compte.

3.1.4 Le génie et la science

Rodgers (1983) souligne que la science trouve ses racines dans une discipline nommée la philosophie naturelle. Cette discipline était basée essentiellement sur l'observation des phénomènes naturels sur lesquels les philosophes de l'époque émettaient des hypothèses sur leurs structures et leurs comportements. La philosophie naturelle a évolué vers la science telle que connue aujourd'hui lorsque le vocabulaire du langage écrit a atteint un niveau de maturité suffisant pour identifier, expliquer et représenter les phénomènes expliqués. À l'instar des scientifiques, les philosophes naturels recherchaient des explications pouvant être vérifiées par des expérimentations répétables et donnant des résultats stables. Ces expériences répétables étaient un pré requis à un consensus entre les philosophes.

Rodgers (1983) souligne que le scientifique a comme motivation d'expliquer les phénomènes et les choses du monde réel. Également, les scientifiques auraient tendance à rechercher des théories qui provoquent une révolution dans les concepts d'un domaine pointu et être ainsi des aspirants aux prix Nobel. Cependant, ces propos de Rodgers sont fortement nuancés par Thomas Kuhn (1970) à l'effet que la majorité des scientifiques exécutent leurs travaux dans un contexte de science dite normale. Dans ce contexte, les scientifiques basent leurs travaux sur des contributions scientifiques antérieures associées à un paradigme spécifique. Un paradigme définit les éléments de base, comme les théories, les lois, les méthodes, les instruments et les orientations de solutions aux

problèmes reconnus par le paradigme. Associé de près à un paradigme, une communauté scientifique est formée et partage, sous forme de consensus, les éléments du paradigme scientifique. Cette communauté fera l'évaluation des travaux de recherche effectués au sein du paradigme.

Ainsi, à l'encontre des propos de Rodgers, Kuhn affirme que la majorité des travaux de recherche scientifique n'a pas pour objectif de produire des révolutions scientifiques. Le but de la science normale est d'améliorer, de raffiner, de préciser et de valider les éléments de base du paradigme. Kuhn souligne que la révolution scientifique n'est pas exclue, mais qu'elle ne se présente que très rarement. Une révolution scientifique bouleverse les paradigmes en place soit en les modifiant en profondeur ou en les remplaçant tout simplement.

La science est basée sur un système déductif raffiné qui ne laisse pas de place à l'approximation qui pourrait fausser le choix d'une théorie par rapport à une autre. La science peut être précise. Les expérimentations sont réalisées à l'aide d'instruments précis, le nombre de variables est limité et l'environnement est rigoureusement contrôlé. Historiquement, les résultats des expérimentations de la science ont pu être généralisés sous forme de lois et reléguer aux oubliettes plusieurs croyances de l'époque à l'effet que les nombreux *dieux* étaient responsables de tous les événements du monde naturel. Le scientifique fait aussi preuve de créativité dans son travail. Cette créativité est motivée par l'explication des phénomènes du monde naturel et le sentiment de fierté d'effectuer des découvertes.

Le génie trouve ses racines au niveau de la construction de « crafts » tels des outils et des machines. Ces « crafts » permettaient de faciliter le travail des gens de l'époque en réalisant des tâches difficilement faisables à la main. La conception de « craft » trouve son origine bien avant la philosophie naturelle et la science, puisque le « craft » ne

requérait que de la dextérité et de la créativité manuelle. Quant à la science, elle requiert un langage écrit évolué et des éléments des mathématiques.

À l'instar des scientifiques, les ingénieurs font eux aussi des expérimentations. Cependant, celles-ci se concentrent sur les caractéristiques macroscopiques des objets comme le comportement des matériaux dans un contexte donné avant de les utiliser dans un projet. De ces expérimentations, les connaissances acquises sont consolidées dans le « handbook » de la spécialité sous forme de théories, de règles ou de formules. L'ingénieur fait aussi preuve de créativité dans sa passion de concevoir des objets utiles et surtout dans ceux que l'on ne pensait pas réalisables.

Rodgers (1983) souligne que le corpus de connaissances du génie diffère de celui de la science sous trois aspects. En premier lieu, les buts et objectifs de chacun diffèrent. La science tente de comprendre le monde qui nous entoure alors que le génie tente de combler les besoins de la société en modifiant le monde qui nous entoure.

En deuxième lieu, la science et le génie reposent sur des présuppositions différentes. La science présuppose une certaine régularité dans la nature alors que le génie présuppose que la nature peut être modifiée et manipulée afin de satisfaire les besoins de la société. Le progrès au niveau de la science est axé sur de meilleures théories, de meilleurs concepts permettant de mieux expliquer et prédire les phénomènes. La généralisation des théories sous forme de lois est aussi un autre volet du progrès scientifique. Au niveau du génie, le progrès peut s'observer sous deux volets : le produit et le processus de fabrication du produit. Ainsi, les produits créés peuvent être plus performants ou rendre caduques d'autres produits. Au niveau du processus, le progrès permet de fabriquer les produits plus efficacement du point de vue économique et environnemental.

En dernier lieu, le génie est plus préoccupé que la science par les considérations sociales, économiques et environnementales. Le génie tient aussi compte des aspects de sécurité, de fiabilité, de durabilité et d'esthétique. Le génie est aussi une activité beaucoup plus visible tant au niveau des succès que des échecs. La science est une activité plutôt privée en laboratoire où les préoccupations sociales et économiques ne seraient pas au centre des préoccupations des scientifiques. Cependant, la science n'est pas exempte de ces préoccupations. À titre d'exemple, les manipulations génétiques et leurs conséquences sur l'humanité sont débattues par la communauté scientifique.

Rodgers (1983) propose une classification des connaissances techniques du génie présentée à la figure 15. L'ingénieur doit recourir à plusieurs technologies pour atteindre ses objectifs. Ainsi, ces technologies se regroupent sous deux catégories : les *matériaux* et les *produits*. Afin de soutenir ces catégories, le génie dispose aussi d'une couche de connaissances plus théorique nommée le « *engineering science* ». Le « engineering science » représente la science de nature analytique à la base du génie. Cette catégorie de connaissances effectue le lien entre le génie et la science. Le « engineering science » partage des caractéristiques communes avec la science. Les connaissances se développent à partir des problèmes rencontrés et des solutions trouvées. Ainsi, les connaissances acquises sont cumulatives et enrichissent continuellement le corpus des connaissances propres au génie. De ce fait, Rodgers (1983) affirme que la caractéristique du « *selft generating* » des connaissances de la science s'applique aussi au génie par l'entremise du « engineering science ». Le génie a des problématiques spécifiques qui stimulent les chercheurs du génie à explorer, expérimenter et à trouver des explications et des solutions.

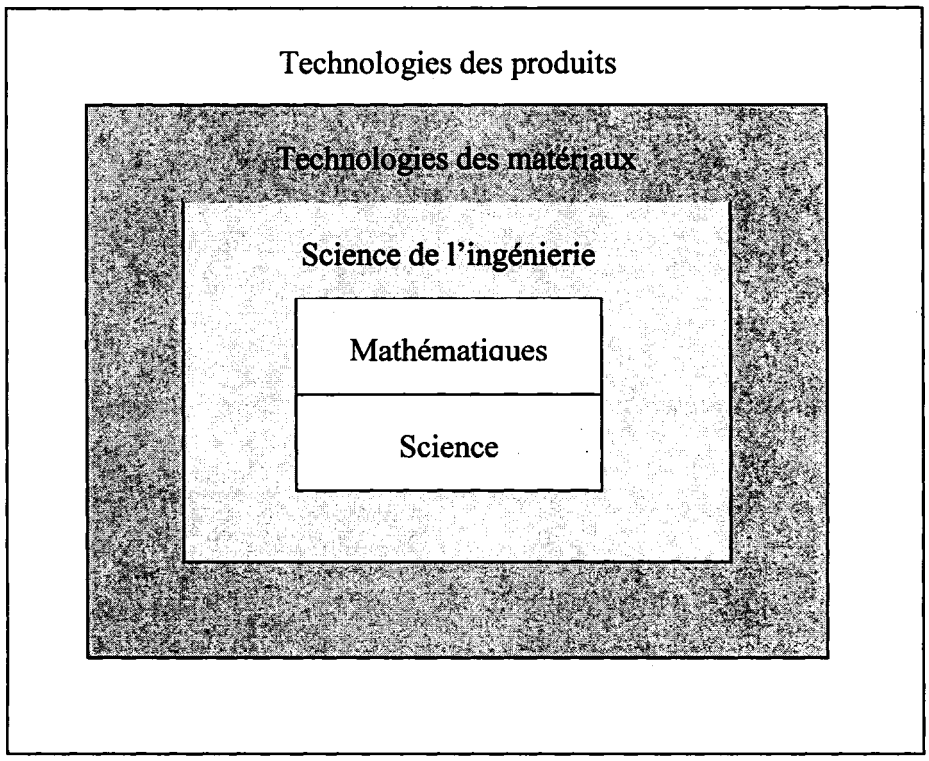

Figure 15 Classification des connaissances techniques du génie (adaptation du modèle de Rodgers (1983))

Les théories empiriques obtenues aident à analyser de nouvelles situations. Le « engineering science » se développe sur une base continue et alimente les catégories *produits* et *matériaux* du génie.

Le « engineering science » se distingue de la science par les caractéristiques suivantes. En premier lieu, les théories du génie peuvent très bien soutenir le design et être très utiles en pratique, mais ne pas être adéquates du point de vue scientifique. Le succès en pratique d'une théorie ne la valide pas automatiquement. Le génie doit aussi composer avec des approximations qui ne pourraient pas être acceptables du point de vue de la science. En dernier lieu, le génie ne peut se permettre de falsifier ses résultats compte tenu de la responsabilité professionnelle et des conséquences importantes des échecs généralement bien médiatisés.

3.1.5 Le génie et les arts

Le génie comporte un volet essentiel de créativité. Cette créativité amène à se demander si le génie a des liens avec l'art.

Selon Rodgers (1983), l'art concerne essentiellement l'exploration et l'expression des émotions conscientes ou non d'une personne (l'artiste) par des moyens tels la littérature, les arts graphiques et la musique. Ainsi, l'art est une activité de nature émotive tandis le génie est une activité plutôt rationnelle. La créativité en art est motivée par l'expression des émotions de l'artiste face à des situations du monde externe.

Jusqu'au 18^{ième} siècle, le mot art référait plutôt à un « craft » ou à de l'expertise spécialisée telle, entre autres, un sculpteur, un menuisier et un médecin. Par la suite, une distinction s'est faite entre les beaux arts et l'art utile. Également, une distinction s'est aussi faite entre l'artiste et l'artisan.

Rodgers (1983) souligne que le génie se distingue des arts selon trois caractéristiques. En premier lieu, le génie est une *activité planifiée* et le *résultat à obtenir est déterminé* avant le début du processus d'ingénierie. La finalité en art n'est pas toujours planifiée. L'artiste peut faire évoluer son œuvre au fil de l'exploration de ses émotions durant la réalisation de son œuvre.

En second lieu, au niveau du génie, il est possible de distinguer les matériaux de base du produit final. À son tour, le produit final peut servir de matériel de base à un autre produit. Il est ainsi possible d'établir une classification hiérarchique (i.e. est composé de) des différents produits et matériaux. En arts, cette distinction est plus futile et il est souvent difficile de distinguer les deux. De plus, une œuvre peut rarement servir de matériel de base à une autre œuvre. La classification hiérarchique en arts est donc difficilement réalisable.

En dernier lieu, le génie peut différencier les moyens utilisés pour fabriquer le produit, du produit lui-même. En arts, cette distinction n'est pas aussi claire. Un poète n'a pas besoin d'outils pour composer son œuvre. L'utilisation de mots et d'un crayon ne serait pas considérée comme des moyens au même titre que ceux utilisés par le génie. Une autre personne n'arriverait pas au même résultat, même en connaissant les mots et en utilisant le même crayon.

Le génie et l'art partagent une caractéristique commune. Ainsi, les deux disciplines se caractérisent par la création d'artéfacts et non de théories pour les expliquer. La créativité n'est pas réservée uniquement aux arts puisqu'on la retrouve aussi en science et en génie.

Le génie ne serait donc ni de l'art, ni de la science pure, mais il se situerait quelque part entre les deux disciplines. De plus, l'utilisation de l'expression en génie de l'état de l'art ne serait pas appropriée. Rodgers souligne à cet effet que l'utilisation de l'état de la technologie serait plus appropriée, tout en reconnaissant que la formule est moins élégante. On peut aussi ajouter que l'expression de l'état de l'art fait souvent référence à une synthèse sur un sujet spécifique.

3.1.6 L'éducation en génie

Le génie trouve ses racines dans les activités militaires. En 1676, en France, on assiste à la création du premier groupe d'ingénieurs nommé « le corps du génie » (Davis 1998). Les membres de ce groupe étaient exclusivement des officiers militaires spécialisés en artillerie et en travaux de génie militaire tel des ponts d'appoint. Le « corps du génie » n'était pas une école, mais un regroupement de spécialistes. Ce groupe était considéré comme le « gardien » du savoir-faire militaire entre les guerres.

En 1690, la France crée la première école d'artillerie afin de répondre au besoin d'offrir une formation structurée aux ingénieurs de cette spécialité (Rodgers 1983). Toujours en France, en 1716, le « corps des ponts et chaussées » voit le jour. L'objectif de ce groupe était de bâtir et de maintenir de réseaux des routes et des canaux navigables de la France.

Les français ont été ainsi les premiers à reconnaître la nécessité d'offrir une formation structurée aux ingénieurs. En 1749, « l'École du génie » fut créée. Le corps professoral de cette école entreprit d'écrire une synthèse des connaissances du génie. Les thèmes traités portaient sur les structures, les matériaux, l'hydraulique, la mécanique et les mathématiques appliquées. L'aboutissement de ces travaux mènera à la création de l'École polytechnique en 1794. L'École offrait dès ses débuts plusieurs spécialisations du génie. De plus, elle offrait un cheminement rigide composé d'aspects théoriques et pratiques d'une durée de quatre ans. Les trois premières années composaient le tronc commun de tous les étudiants et la quatrième année était celle de la spécialisation. L'École appliquait une sélection stricte au niveau de l'admission des candidats. Le modèle de l'École polytechnique servit de modèle à la création de plusieurs autres écoles de génie dans le monde telle l'Académie militaire de West Point aux USA en 1817. Cette dernière est considérée comme la première école de génie aux États-Unis (Davis 1998).

Vers 1850, on assiste à la mise en place du modèle moderne d'éducation en génie. Ce modèle comporte un *curriculum formel*, un *examen de certification* à la fin des études et la constitution d'un *code d'éthique* régissant la profession.

Davis (1998) souligne que l'origine militaire du génie a teinté la profession. D'une part, les ingénieurs militaires de l'époque devaient accorder une attention particulière à la fiabilité des objets conçus (les canons, les bombes, les fusils, etc). Ainsi, les ingénieurs procédaient à des *tests systématiques* des matériaux et des procédures de fabrication avant la production en volume. D'autre part, ces ingénieurs étaient avant tout des

officiers de l'armée formés d'une façon *disciplinée* et aptes à prendre en charge des responsabilités. Également, la formation en mathématiques et en physique permettait à ces ingénieurs d'aborder les problèmes d'une façon *systématique.*

Le titre d'ingénieur est contrôlé par les corporations ou les ordres d'ingénieurs. Ainsi, dans certains pays, l'ingénieur doit être reconnu et obtenir une licence de l'ordre des ingénieurs pour exercer sa profession. Les corporations d'ingénieurs peuvent poursuivre les faux ingénieurs et suspendre le permis de pratique d'un ingénieur qui n'aurait pas respecté le code d'éthique ou qui serait reconnu coupable de fautes professionnelles.

3.1.7 Concepts du génie

Dans les sections antérieures, il a été souligné que le génie s'appuie sur deux assises constituant sa fondation. La première est le corpus de connaissances du génie représenté par le « engineering science ». La seconde est le processus d'ingénierie qui applique les connaissances du « engineering science » dans la conception d'artéfact. Le produit ou service issu du processus d'ingénierie doit satisfaire les besoins et les exigences exprimés par le client. Le processus sera réalisé par des individus (les ingénieurs) impliqués dans les activités. La figure 16 présente les concepts généraux du génie.

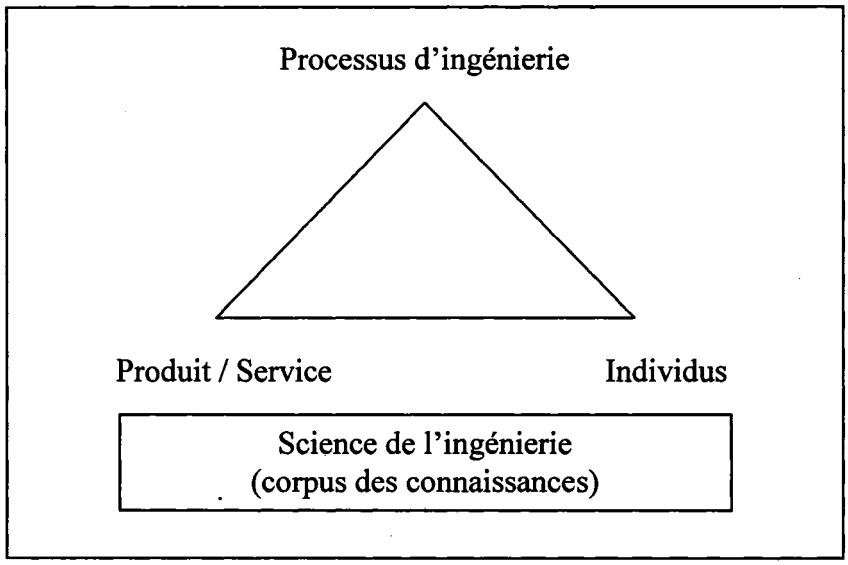

Figure 16 Concepts généraux du génie

3.1.7.1 Processus d'ingénierie

Aslaksen (1996) souligne que le processus d'ingénierie est un composant majeur du génie. C'est au sein de ce processus que les connaissances du génie sont appliquées. Le processus d'ingénierie est composé d'une suite d'activités intellectuelles et rationnelles qui débute avec la phase de définition des besoins. L'objectif du processus est de produire l'artéfact (produit/ service) qui répondra aux besoins du client et qui tient compte des contraintes imposées. La figure 17 présente une modélisation des activités principales du processus d'ingénierie.

Définition des besoins	Design	Construction Fabrication	Entretien
Management des activités			
Activités de recherche et développement			

Figure 17 Activités du processus d'ingénierie (modélisation du texte de Aslaksen 1996)

Le processus d'ingénierie doit être planifié, décrit, évalué et optimisé (Aslaksen 1996). De plus le processus doit permettre la traçabilité des choix et des décisions prises tout au long de la réalisation des activités. Le processus intègre d'une façon cohérente d'autres processus, des méthodes, des techniques, des outils et des normes pour standardiser le déroulement du processus, tel que présenté à la figure 18. Également, le processus dans son entier fait l'objet d'une planification en divers volets tels, les plans de projet, de risques, etc. Le processus d'ingénierie a comme caractéristiques d'être complexe, de comporter des risques et de l'incertitude (Aslaksen 1996). Cependant, ces caractéristiques ne sont pas exclusives à l'ingénierie.

Plans (planification)	N
Processus	o
Méthodes	r
Techniques	m
Outils	e
	s

Figure 18 Principaux concepts associés au processus d'ingénierie

3.1.7.2 Produit ou service

Le produit ou le service conçu est l'aboutissement du processus d'ingénierie. Le produit est le résultat des décisions et des compromis faits tout au long du processus depuis la définition des besoins jusqu'à sa fabrication et tests. À l'origine du génie, lorsque les ingénieurs étaient avant tout des officiers militaires, les armes ou produits militaires conçus se devaient d'être fiables. Des bombes qui n'explosent pas, des fusils qui s'enraillent et autres équipements qui se brisent sont, entre autres, des exemples d'événements qui peuvent changer l'issue d'une bataille. Ainsi, la fiabilité du produit serait dès l'origine du génie, une qualité recherchée par les ingénieurs. Pour y arriver, les ingénieurs-officiers de l'époque ont mis en place des procédures systématiques de tests pour valider la fiabilité du produit. Le produit peut aussi posséder d'autres qualités telles la durabilité, la faciliter d'entretien, la possibilité de la fabriquer économiquement. De plus, le produit doit aussi se conformer aux contraintes environnementales et légales[16].

Le produit doit évidemment satisfaire les besoins exprimés par le client. Ainsi, la définition des besoins est une activité importante qui se doit d'être bien faite dans la mesure où les activités subséquentes s'y réfèrent. Tel que souligné par Aslaksen (1996), cette activité doit être incluse dans le processus d'ingénierie, ainsi que son management.

3.1.7.3 Les individus

Les ingénieurs ont reçu une formation formelle et accréditée par les corporations d'ingénieurs. L'ingénieur a été formé pour aborder les problèmes et les résoudre d'une façon systématique i.e. avec méthode, dans un ordre et un but déterminé. Découlant des origines militaires du génie, l'ingénieur effectue son travail avec un sens du devoir, de l'ordre et de l'obéissance. Les ingénieurs de l'époque étaient en premier lieu des

---

[16] Ces contraintes ne sont pas toujours respectées dans le contexte militaire.

officiers militaires et il était donc naturel qu'ils organisent et dirigent les travaux et qu'ils assument la responsabilité des résultats obtenus. Même si l'ingénieur moderne n'est plus d'office un militaire, il est formé à organiser le travail à effectuer, à prendre des décisions et à assumer la responsabilité de ses actes. L'ingénieur doit aussi se conformer et respecter le code d'éthique de sa profession. Il doit garder à l'esprit qu'il engage sa responsabilité professionnelle dans chaque activité qu'il réalise. L'ingénieur pris en défaut peut être sanctionné par l'ordre des ingénieurs et à l'extrême se voir radier et ainsi ne plus être en mesure d'exercer sa profession.

Le tableau XVII résume les concepts principaux à la base génie identifiés dans ce chapitre. Ces concepts, entre autres, serviront de référence dans l'évaluation des principes du génie logiciel. Ils seront utilisés pour établir des critères permettant d'identifier des principes se rattachant au génie.

Tableau XVII

Concepts principaux à la base du génie

Processus d'ingénierie	**Produit ou service**	**Individus**
Processus doit être : ▪ Planifié ▪ Décrit ▪ Évalué ▪ Optimisé Intègre les activités : ▪ Définition des besoins ▪ Design ▪ Fabrication ▪ Entretien ▪ Management ▪ R & D	Satisfaisant les besoins exprimés par le client Qualités ▪ Fiable ▪ Durable Respectant les contraintes ▪ Économiques ▪ Environnementales ▪ Légales ▪ Sociales ▪ Des matériaux	À reçu une formation formelle et accréditée Apte à appliquer des connaissances techniques et théoriques Reconnu comme ingénieur par l'Ordre Respecter le code d'éthique Être en mesure de prendre des responsabilités

Tableau XVII (suite)

Processus d'ingénierie	**Produit ou service**	**Individus**
Caractérisé par ▪ La complexité ▪ L'incertitude ▪ Les risques		Engager sa responsabilité professionnelle Aborder les problèmes d'une façon systématique, disciplinée et avec créativité

3.1.8 Modèle du génie

Moore (2006) présente un modèle intéressant schématisant le processus d'ingénierie. Ce modèle modélise le processus d'ingénierie qui à partir des besoins exprimés et des ressources réalisera un produit satisfaisant aux attentes du client. Ce modèle précise également, le volet contrôle du processus. Le contrôle s'alimente en mesures diverses au sein du processus. Ces mesures sont, pas la suite analysées en considérant, entre autres, les buts et les contraintes du projet, afin de corriger le processus via une action. La figure 19 présente le modèle de Moore (2006).

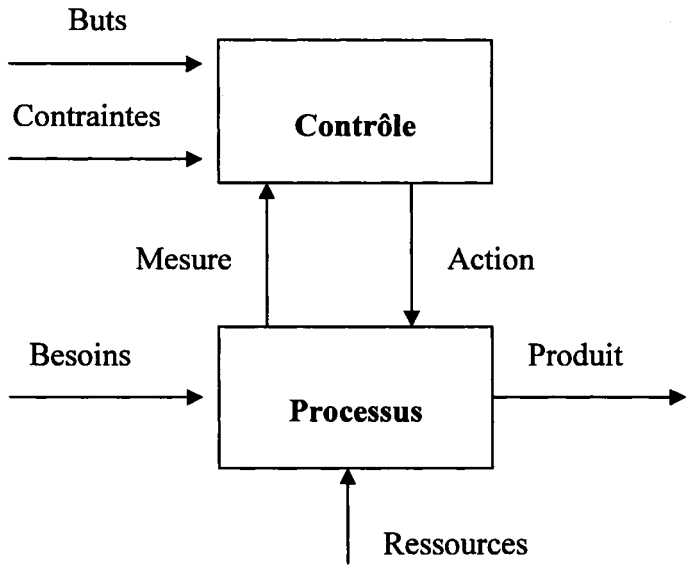

Figure 19 Modèle de l'ingénierie (Traduit de Moore 2006)

3.2 Le génie logiciel

L'expression « software engineering » fut le titre donné à une conférence de l'OTAN en 1968. Comme le rapporte Naur et al. (1968), le choix de ces termes avait pour objectif de sensibiliser les développeurs à l'effet que le logiciel devrait être produit en utilisant des connaissances théoriques et pratiques tel que fait dans les disciplines établies du génie. Cette conférence rassembla des experts en développement de logiciels intéressés à trouver et à explorer des pistes de solutions pour faire face aux problèmes identifiés tels, entre autres, la croissance de la complexité dans le développement des logiciels, les dépassements de coûts et d'échéanciers ainsi que les aspects de fiabilité et d'entretien du logiciel.

Dans le cadre de cette recherche, nous ne traiterons pas du sujet délicat à savoir si le génie logiciel est une discipline du génie à part entière ou s'il est plutôt un sous-domaine de l'informatique. Comme les corporations d'ingénieurs reconnaissent maintenant les programmes de génie logiciel menant au titre d'ingénieur, nous prenons pour acquis que le génie logiciel est une nouvelle discipline du génie en pleine émergence. Cependant, les définitions du génie logiciel abondent, tantôt se rapprochant de l'informatique appliquée et tantôt se rapprochant du génie. Pour le cadre de cette recherche, nous adoptons la définition du génie logiciel tel que cité dans le guide du corpus de connaissances du génie logiciel (SWEBOK 2004). Ce guide a fait l'objet de trois versions majeures, soutenues par des organismes réputés telles IEEE Computer Society et ISO (ISO TR19760). De plus, SWEBOK a été revu par plusieurs centaines d'experts internationaux et son contenu fait l'objet d'un large consensus concernant les connaissances acceptées par la communauté du génie logiciel, ce qui représente un atout significatif.

Le guide SWEBOK a retenu la définition du génie logiciel donnée par la IEEE Computer Society :

> *« The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software" (p.1-1).*

Cette définition donnée au génie logiciel est en fait presque la même que l'IEEE a donné au terme génie (norme IEEE 610). Ainsi, on y retrouve les concepts importants associés au génie tels l'approche *systématique*, *disciplinée* et *quantifiable*, ainsi que les activités de développement et d'entretien.

Les connaissances référencées par le guide SWEBOK font l'objet d'un consensus à l'effet qu'elles sont généralement utilisées et requises dans un projet de développement d'un logiciel. Le guide se divise en dix domaines de connaissances du génie logiciel tel que présenté à la figure 20. À ces domaines, s'ajoutent un chapitre sur les disciplines frontières du génie logiciel.

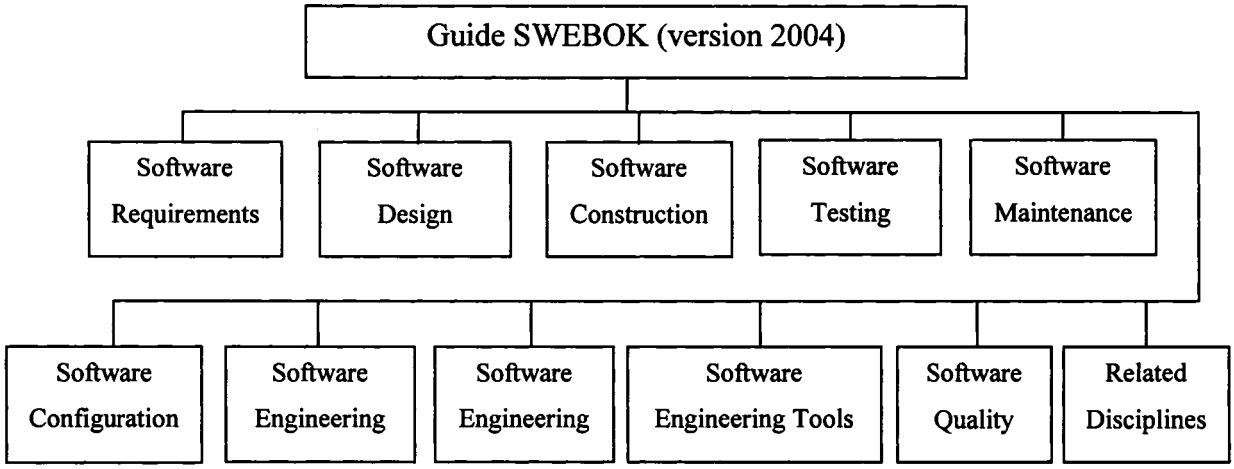

Figure 20 Domaines de connaissances du guide SWEBOK (Abran et al. 2004)

3.2.1 Les concepts du génie logiciel

Nous allons maintenant présenter sommairement chacun des domaines de connaissances et identifier les principaux concepts de chacun. Ces concepts seront nécessaires pour l'analyse des principes qui suivra. Les concepts sont extraits à partir de la structure de base de chacun des chapitres de SWEBOK. Ainsi, un tableau est produit pour chacune des sections d'un domaine de connaissances. La colonne *sous-section* représente les sous-sections. Nous avons conservé le titre exact des sous-sections en anglais pour éviter des confusions de traduction inutiles considérant que les principes à analyser sont tous formulés en anglais. La colonne *concept* représente les concepts présents dans chacune des sous-sections. Seuls les concepts sont énumérés et non un résumé du contenu de la sous-section. Les prochaines sections présentent essentiellement des tableaux, ainsi, le lecteur peut à sa guise passer directement à la section 3.2.2.

3.2.1.1 Les exigences logicielles (Software Requirements)

Ce domaine de connaissance se concentre sur l'élucidation, l'analyse, la spécification et la validation des exigences logicielles. Ces activités sont fondamentales dans le processus de développement de logiciel. Les exigences concrétisent les besoins exprimés par le client et les contraintes spécifiques auxquelles le futur logiciel devra se conformer. Une exigence est une propriété que le logiciel doit comporter afin de répondre aux demandes du client. Le domaine de connaissance est subdivisé en sept sections. Nous présentons sept tableaux mettant en évidence les concepts de chacune des sections. Il est à noter que les concepts ont été laissés en langue originale anglaise afin d'éviter les problèmes de traduction. Compte tenu que l'ensemble des 308 propositions analysées sont toutes en anglais, il est primordial d'éviter les traductions qui peuvent amener des flous.

Tableau XVIII

Concepts de la section "Software Requirements Fundamentals"

Software Requirement Fundamentals	
Sous-sections	Concepts
Definition	Verifiable property which must be exhibited by software to solve a particular problem
Product and Process Requirements	▪ Product: requirement on software ▪ Process: a constraint on the development of software
Functional and Non-functional Requirement	▪ Functions that the software is to execute ▪ Constraints or quality requirement ▪ Performance, maintainability, safety, reliability
Emergent Properties	Requirement which depend on integration of component
Quantifiable Requirements	Can be verified
System Requirements and Software Requirement	▪ System : the whole system ▪ Software : requirement derived from system

Tableau XIX

Concepts de la section "Requirement Process"

Requirements Process	
Sous-sections	Concepts
Process Models	▪ Iterative activity ▪ Requirement part of Configuration Management ▪ Must adapted to the organization and project context ▪ Elicitation, analysis, specification, validation
Process Actors	▪ Users, Customers, Marketing, regulator, engineers
Process Support and Management	▪ Cost, human resources, training, and tools
Process Quality and Improvement	▪ Quality standard, Customer's satisfaction ▪ Software Quality Attributes, Measurement and Benchmarking ▪ Improvement planning and implementation

Tableau XX

Concepts de la section "Requirements Elicitation"

Requirements Elicitation	
Sous-sections	Concepts
Requirement Sources	▪ Business goals, Domain knowledge, Stakeholders ▪ Operational & Organizational environment
Elicitation Techniques	▪ Interviews, Scenarios, Prototypes, Meetings, Observations

Tableau XXI

Concepts de la section "Requirements Analysis"

Requirements Analysis	
Sous-sections	Concepts
	▪ Detect and resolve conflicts between requirements ▪ Bounds of the software ▪ System requirements
Classification	▪ Functional and Non-Functional ▪ Derived from a high level requirement ▪ Product or Process requirement ▪ Priority and Scope ▪ Volatility or Stability of requirement
Conceptual Modeling	▪ Data and Controls flows, State models, Event Traces ▪ User Interaction, object models, data models
Architectural Design and Requirement allocation	▪ Software architect, Software components ▪ Requirement allocation to components
Requirement negotiation	▪ Conflict resolution : o Stakeholder o Between requirement and resources o Between functional and non-functional requirement

Tableau XXII

Concepts de la section "Requirement Specification"

Requirements Specification	
Sous-sections	Concepts
System Definition Document	▪ Concept of operations ▪ High level system requirements from the domain perspective ▪ IEEE 1362
System Requirement definition	▪ System engineering ▪ IEEE 1233
Software Requirement Specification	▪ The basis for agreement between customers and contractors or suppliers ▪ Rigorous assessment of requirements ▪ Basis for estimating product costs, risks and schedules, software enhancement, acceptation tests

Tableau XXIII

Concepts de la section "Requirement validation"

Requirements validation	
Sous-sections	Concepts
Requirements Reviews	▪ Inspection or Reviews Requirements o Errors, mistaken assumptions, lack of clarity and deviation from standard practice o Checklist
Prototyping	▪ Validating the software engineer's interpretation of the requirements ▪ Eliciting new requirements ▪ Feed-back from users
Model validation	▪ Static analysis and Formal reasoning
Acceptance Tests	▪ Validate the final product

Tableau XXIV

Concepts de la section "Practicals considerations"

Practical Considerations	
Sous-sections	Concepts
Iterative nature of the requirement process	▪ Iterate to towards a level of quality and detail ▪ Revision of requirements (changes)
Change Management	▪ Change management procedures ▪ Change analysis
Requirement Attributes	▪ Classification ▪ Acceptance Test plan ▪ Identifier : unique and unambiguous
Requirement tracing	▪ Source of requirement (backwards) ▪ Impact of changes ▪ To design entities (forwards)
Mesures	▪ Volume of requirements (number) ▪ Size of changes (evaluating) ▪ Cost estimation (development & maintenance0

3.2.1.2 Le design du logiciel (Software Design)

Le design est un processus qui fait l'analyse des exigences logicielles dans le but de concevoir une description de l'architecture interne du logiciel et de ses composants. L'étape du design peut produire plusieurs solutions possibles. Celles-ci seront évaluées et une sera choisie. Le design comprend deux activités principales. En premier lieu le design architectural exposera les grandes lignes de la solution proposée et identifiera ses principaux composants. Par la suite, le design détaillé prendra en charge la description de chacun des composants identifiés à un niveau de détail suffisant pour le construire (le programmer). Les concepts du design se regroupent en six sections au niveau des six prochains tableaux.

Tableau XXV

Concepts de la section "Software design fundamentals"

Software Design Fundamentals	
Sous-sections	Concepts
General Design Concepts	▪ Problem-solving ▪ Goals, constraints, alternatives, representations, and solutions.
Context of Software Design	▪ Phase of software engineering life cycle
Design process	▪ Architectural Design ▪ Detailed Design
Enabling techniques	▪ Abstraction ▪ Coupling and Cohesion ▪ Decomposition and modularization ▪ Encapsulation / information hiding ▪ Separation of interface and implementation ▪ Suffiency, completeness and primitiveness

Tableau XXVI

Concepts de la section "Key issues in software design"

Key Issues in Software Design	
Sous-sections	Concepts
Concurrency	▪ How to decompose the software into processes, tasks and threads ▪ Efficiency, atomicity, synchronization, scheduling
Control and Handling of events	▪ Organize data and control flow ▪ Events handling
Distribution of components	▪ Distribute the software across hardware ▪ Component communication ▪ Middleware and heterogeneous software
Errors, exception handling Fault Tolerance	▪ To prevent and tolerate faults, exception handling
Interaction and presentation	▪ Structure and organization of the interactions with users and business logic
Data persistence	

Tableau XXVII

Concepts de la section "Software structure and architecture"

Software Structure and Architecture	
Sous-sections	Concepts
Architectural structures and viewpoints	▪ Logical view (functional requirement) ▪ Process view (concurrency) ▪ Physical view (distribution) ▪ Development view (implementation units)
Architectural Styles	▪ General structure, distributed systems, interactive systems, adaptable systems, rule-based systems
Design patterns	▪ Micro architecture ▪ Creational patterns, Structural patterns, Behavioural patterns
Families of programs and frame work	▪ Maximize reuse

Tableau XXVIII

Concepts de la section "Software design quality analysis and evaluation"

Software Design Quality Analysis and Evaluation	
Sous-sections	Concepts
Quality attributes	▪ Discernable at run-time o Performance, security, availability, functionality, usability) ▪ Non-discernable at run-time o Modifiability, portability, reusability, testability, integrability
Quality analysis and evaluation techniques	▪ Software design reviews & inspections ▪ Static analysis ▪ Simulation and prototyping
Measures	▪ Function-oriented design measures ▪ Object-oriented design measures

Tableau XXIX

Concepts de la section "Software Design Notations"

Software Design Notations	
Sous-sections	Concepts
Structural descriptions (static view)	▪ Architecture description languages ▪ Class and object diagrams ▪ Component diagrams ▪ Collaboration responsibilities cards ▪ Deployment diagrams ▪ Entity-relationship diagrams ▪ Interface description languages ▪ Jackson structure diagrams ▪ Structure charts
Behaviour descriptions (dynamic view)	▪ Activity diagrams ▪ Collaboration diagrams ▪ Data flow diagrams ▪ Decision tables and diagrams ▪ Flowcharts and structured flowcharts ▪ Sequence, state transition and statechart diagrams ▪ Formal specification languages ▪ Pseudo-code and program design languages

Tableau XXX

Concepts de la section "Software design strategies and methods"

Software Design Strategies and Methods	
Sous-sections	Concepts
General Strategies	▪ Divide and conquer ▪ Stepwise refinement ▪ Top-down & Bottom-up ▪ Data Abstraction & information hiding ▪ Heuristics use ▪ Patterns use ▪ Iterative and incremental approach
Function-oriented design	▪ Identifying software functions ▪ Data flows diagram
Object-oriented design	▪ Inheritance and polymorphism ▪ Component-based design

Tableau XXX (suite)

Software Design Strategies and Methods	
Sous-sections	Concepts
Data-structure design	▪ Data structure
Component design	▪ Independence ▪ Interfaces ▪ Reuse

3.2.1.3 Construction du logiciel (Software Construction)

Les connaissances du domaine de la construction logiciel concernent les activités de réalisation des composants du logiciel telle la programmation, les tests unitaires et d'intégration et la mise au point. Pour réaliser ces activités, des outils sont utilisés tels des compilateurs, des environnements de programmation, des débogueurs et des utilitaires facilitant les tests des composants développés. Ce domaine est divisé en trois sections présenté au niveau des trois prochains tableaux.

Tableau XXXI

Concepts de la section "Software construction fundamentals"

Software Construction Fundamentals	
Sous-sections	Concepts
Minimizing complexity	▪ Code : simple and readable ▪ Use of standards and quality techniques
Anticipating change	
Constructing for verification	▪ Detect faults ▪ Coding standards ▪ Code review, unit testing, automated testing
Standards in construction	▪ Tools (UML) ▪ External standards o Languages, tools, interfaces o IEEE, ISO, OMG ▪ Internal standards o At corporate level or specific projects

Tableau XXXII

Concepts de la section "Managing Construction"

Managing Construction	
Sous-sections	Concepts
Construction Models	▪ Waterfall models ▪ Iterative models
Construction Planning	▪ Choice of construction method ▪ Reduce complexity, anticipate change, construct for verification
Construction Measurement	▪ Code developed, modified, reused, destroyed ▪ Code complexity and inspection statistics ▪ Fault-fix and fault-find rates

Tableau XXXIII

Concepts de la section "Practical Considerations"

Practical Considerations	
Sous-sections	Concepts
Construction Design	▪ Constraints of the real-world problem
Construction languages	▪ Configuration files ▪ Toolkit languages o Programming languages : linguistic, formal, visual
Coding	▪ Techniques for code writing ▪ Use of classes, enumerated types, variables, constant ▪ Control structures ▪ Handling errors conditions ▪ Code-level security breaches ▪ Resources usage ▪ Source code organization ▪ Code documentation and tuning
Construction Testing	▪ Unit testing ▪ Integration testing ▪ IEEE 829 et 1008

Tableau XXXIII (suite)

Practical Considerations	
Sous-sections	Concepts
Reuse	▪ Integrating reuse processes and activities in life cycle ▪ Selection of reusable units, databases, test procedures ▪ Evaluation of code or test reusability ▪ Reporting reuse
Construction quality	▪ Unit testing and integration testing ▪ Code stepping ▪ Use of assertions ▪ Debugging ▪ Technical reviews ▪ Static analysis
Integration	▪ Routines, classes, components, subsystems ▪ Planning the sequence of integration ▪ Tests

3.2.1.4 Tests du logiciel (Software Testing)

Les tests sont effectués afin d'évaluer la qualité du logiciel et de l'améliorer en identifiant les défauts. Les tests du logiciel se concentrent sur la vérification dynamique du comportement du logiciel à l'aide d'un jeu d'essais. Les activités de tests s'intègrent maintenant plus tôt dans les activités de développement et ne sont plus confinés qu'à la suite de la programmation. Le domaine des tests se subdivise en cinq sections qui sont présentées au niveau des cinq prochains tableaux

Tableau XXXIV

Concepts de la section "Software testing fundamentals"

Software Testing Fundamentals	
Sous-sections	Concepts
Testing-related Terminology	▪ Dynamic verification ▪ Finite set of test cases ▪ Selection criterion ▪ Expected behaviour ▪ Fault & defect: cause ▪ Failure: undesired effect
Keys issues	▪ Test selection criteria ▪ Testing effectiveness ▪ Testing for defect identification ▪ Theoretical and practical limitations of testing ▪ Testability
Relationships of testing with others activities	▪ Quality ▪ Formal Verification ▪ Debugging ▪ Programming ▪ Certification

Tableau XXXV

Concepts de la section "Test levels"

Test Levels	
Sous-sections	Concepts
Target of the test	▪ Unit testing (IEEE 1008) ▪ Integration testing (components) ▪ System testing
Objectives	▪ Conformance, Correctness or Functional testing ▪ Acceptance testing ▪ Installation testing ▪ Alpha, Beta testing ▪ Reliability ▪ Regression and stress testing ▪ Back to back testing ▪ Configuration testing ▪ Usability testing ▪ Test-driven development

Tableau XXXVI

Concepts de la section "Test Techniques"

Test Techniques	
Sous-sections	Concepts
Intuition & Experience	▪ Ad hoc testing ▪ Exploratory testing
Specification-based	▪ Equivalence partitioning ▪ Boundary-value analysis ▪ Decision table ▪ Finite-state machine-based ▪ Formal specifications ▪ Random testing
Code-based	▪ Control-flow based ▪ Data flow based ▪ Reference models o Flowgraph, Call graph
Fault-based	▪ Error guessing ▪ Mutation testing
Usage-based	▪ Operational profile ▪ Software reliability Engineered testing
Nature of Application	▪ Object-oriented ▪ Component based ▪ Web-based ▪ GUI testing ▪ Concurrent testing ▪ Protocol conformance testing ▪ Real-time testing ▪ Safety-critical testing
Selection & combining techniques	▪ Functional & structural ▪ Deterministic vs random

Tableau XXXVII

Concepts de la section "Test related measures"

Test Related Measures	
Sous-sections	Concepts
Evaluation of program	▪ Program size and structure (complexity) ▪ Fault types, classification and statistics ▪ Fault density ▪ Life test ▪ Reliability growth models
Evaluation of the tests	▪ Coverage/thoroughness measures ▪ Fault seeding ▪ Mutation score

Tableau XXXVIII

Concepts de la section "Test Process"

Test Process	
Sous-sections	Concepts
Practical considerations	▪ Attitude / Egoless programming ▪ Test Guide ▪ Test process management o People, tools, policies, measurements, ▪ Test documentation o Test plan o Test procedure specification o Test log o Test incident & problem report ▪ Test cost estimation ▪ Termination ▪ Test reuse and patterns
Tests Activities	▪ Planning ▪ Test cases generation ▪ Test environment development ▪ Test execution ▪ Test results evaluation ▪ Problem reporting ▪ Defect tracking

3.2.1.5 La maintenance du logiciel (Software Maintenance)

La maintenance du logiciel s'intéresse aux activités de modifications du logiciel suite à sa livraison pour effectuer des correctifs et des améliorations ou pour l'adapter à un nouvel environnement. Les activités de maintenance englobent la mise à jour des différents documents du logiciel incluant, entre autres, les guides d'utilisation. La norme ISO/IEC 14764 ajoute les activités de planification des travaux de maintenance. Le domaine de connaissance de la maintenance se subdivise en quatre sections présentées par les quatre prochains tableaux

.

Tableau XXXIX

Concepts de la section " Software Maintenance Fundamentals"

Software Maintenance Fundamentals	
Sous-sections	Concepts
Definitions and Terminology	▪ Modification of a software product after delivery to correct faults, to improve performance or to adapt the product to a modified environment ▪ Modification to code and documentation due to a problem or the need of improvement
Nature of maintenance	▪ Operational life cycle ▪ Modification request are logged and tracked ▪ Testing is conducted ▪ New release of the software ▪ Training and support to users ▪ Maintainer ▪ Problem and modification analysis
Need for maintenance	▪ Correct faults ▪ Improve the design ▪ Implement enhancements ▪ Interface with other systems ▪ Adapt programs ▪ Migrate legacy software ▪ Retire software
Maintenance costs	
Evolution of software	▪ Complexity increases
Categories	▪ *Corrective, adaptive, perfective, preventive*

Tableau XL

Concepts de la section " Key Issues in Software Maintenance"

Key Issues in Software Maintenance	
Sous-sections	Concepts
Technical issues	▪ Limited understanding, Testing, Impact analysis ▪ Maintainability
Management issues	▪ Alignment with organizational objectives ▪ Staffing, Process ▪ Organizational aspects of maintenance ▪ Outsourcing
Maintenance Cost Estimation	▪ Cost estimation, Parametric models, Experience
Software Maintenance Measurement	▪ Benchmarking techniques ▪ Analyzability, Changeability, Stability, Testability

Tableau XLI

Concepts de la section "Maintenance Process"

Maintenance Process	
Sous-sections	Concepts
Maintenance Process	▪ Problem and modification analysis ▪ Modification Implementation ▪ Maintenance review, acceptance ▪ Migration ▪ Software retirement
Activities	▪ Unique activities o Transition o Modification request : acceptance/rejection o Problem report help desk o Software support o Service level agreements and contracts ▪ Software maintenance planning ▪ Concept document for maintenance ▪ Software configuration management ▪ Software Quality

Tableau XLII

Concepts de la section "Techniques for Maintenance"

Techniques for Maintenance	
Sous-sections	Concepts
Program Comprehension	▪ Code browsers ▪ Documentation
Re-engineering	▪ Legacy systems
Reverse Engineering	▪ Identify the components of software ▪ Create high level abstraction ▪ Date reverse engineering

3.2.1.6 Gestion des configurations du logiciel

Ce domaine s'intéresse à l'identification de tous les composants d'un logiciel, incluant les différents documents et plans afin d'être en mesure dans le temps de gérer d'une façon systématique les changements apportés à ceux-ci, tout en conservant leur intégrité. Ce domaine se divise en six sections présentées par les six prochains tableaux.

Tableau XLIII

Concepts de la section "Management of the SCM process"

Management of the SCM process	
Sous-sections	Concepts
Organizational Context	▪ Designated individuals ▪ Closest relationship with development and maintenance
Constraints and Guidance for the SCM process	▪ Corporate policies and procedures ▪ Contract ▪ Best practices ▪ CMMI

Tableau XLIII (suite)

Management of the SCM process	
Sous-sections	Concepts
Planning for SCM	▪ SCM resources and Schedules ▪ Tool selection and implementation o SCM library o Change request and approval procedures o Code and change management tasks o Reporting and SCM measurements o Auditing o Managing and tracking documentation o Performing software builds o Managing and tracking software release ▪ Vendor/Subcontractor control ▪ Interface control
SCM Plan	▪ Introduction (purpose, scope, terms used) ▪ Management (organisation, responsibilities, policies, procedures) ▪ Activities (identification, control) ▪ Resources (tools, resources, human resources) ▪ Maintenance
Surveillance of SCM	▪ SCM measures and measurement ▪ In-process audits

Tableau XLIV

Concepts de la section "Software Configuration Identification "

Software Configuration Identification	
Sous-sections	Concepts
Identifying items to be controlled	▪ Software configuration ▪ Software configuration item and relationships ▪ Software version ▪ Baseline ▪ Acquiring software configuration items
Software Library	▪ Access control, Security,

Tableau XLV

Concepts de la section " Software Configuration Control "

Software Configuration Control	
Sous-sections	Concepts
Requesting, Evaluating, Approving Software Changes	▪ SCM Control Board ▪ Software change request ▪ Deviations and Waivers
Implementing software changes	▪ Software procedures ▪ Tools for tracking changes ▪ Library tools (check-in check-out)
Deviations and Waivers	

Tableau XLVI

Concepts de la section "Software Configuration Status Accounting "

Software Configuration Status Accounting	
Sous-sections	Concepts
SCM Status information	▪ Capture and reporting information ▪ Configurations : identified, collected, maintained
SCM Status reporting	▪ Development, maintenance, project management, SQA teams

Tableau XLVII

Concepts de la section " Software Configuration Auditing "

Software Configuration Auditing	
Sous-sections	Concepts
Functional Audit	▪ Software component consistent with its specifications
Physical Audit	▪ Software design & documentation is consistent to software component
In-process audit	▪ Investigate the current status of specific elements

Tableau XLVIII

Concepts de la section "Software Release Management and Delivery"

Software Release Management and Delivery	
Sous-sections	Concepts
Software building	▪ Combining the correct versions of software configuration items and the appropriate configuration data into an executable program
Software Release management	▪ Identification, packaging and delivery of elements of a product

3.2.1.7 Software Engineering Management

Le thème de la gestion de projet pose un problème. D'une part, il est identifié et décrit au sein d'un chapitre du SWEBOK, combiné avec le management de l'ingénierie du logiciel et d'autre part, il est aussi identifié comme étant une discipline limitrophe du génie logiciel. Au même titre, entre autres, que l'informatique. Nous avons déjà pris position à l'effet que l'informatique n'est du génie logiciel, malgré son influence certaine. De plus, SWEBOK ne consacre pas de chapitre spécifique à l'informatique. Ainsi, les principes candidats contenant des concepts de l'informatique ne seront pas retenus comme étant des principes du génie logiciel. Cependant, qu'en sera-t-il pour les principes contenant des concepts de gestion de projet ? Nous choisissons l'orientation suivante basée sur les choix des auteurs de SWEBOK à l'effet que les concepts généraux de la gestion de projet ne sont pas spécifiques à la gestion de projet logiciel, mais que certains aspects décrits dans le chapitre de SWEBOK sont particuliers au génie logiciel. Nous considérons donc que les concepts généraux répertoriés au sein du *Project Management Body of Knowlewdge (PMI2004)* appartiennent à la discipline de la gestion de projet. Ainsi, PMBOK identifie 44 processus regroupés en neuf domaines tel que présentés au tableau XLIX.

Tableau XLIX

Domaines de connaissances de la gestion de projet (PMBOK-PMI 2004)

1. Project Integration Management	2. Project Scope Management	3. Project Time management
4. Project Cost Management	5. Project Quality Management	6. Project Human Resource management
7. Project communication Management	8. Project Risk Management	9. Project Procurement Management

SWEBOK intitule le chapitre de la gestion de projet comme « *Software engineering Management* » en renforçant les particularités des projets logiciels. Globalement, les cinq processus présentés au sein du chapitre de SWEBOK ne sont pas exclusifs au génie logiciel, cependant, les façons de faire, telles les méthodes, les techniques et les outils utilisés sont spécialisés pour le logiciel. Une nette emphase est constatée au niveau de l'implémentation d'un processus de mesure et de son intégration à la gestion d'un projet.

Les auteurs du chapitre soulignent certaines particularités de la gestion de projet logiciel que nous structurons en fonction des axes suivants : produits, processus et individus.

Le logiciel est un *produit* caractérisé, entre autres, par sa complexité peu perceptible pour les clients. Ainsi, le client perçoit difficilement l'impact sur le produit d'un changement ou d'un ajout, contrairement à un immeuble en construction où l'impact d'un changement peut, en général, se visualiser d'une façon concrète. Le logiciel peut être aussi d'une grande nouveauté, sans équivalent jusqu'à présent, ce qui augmente les facteurs de risques sur le projet. Également, les technologies utilisées pour le développement du logiciel sont caractérisées par un cycle de vie très court qui apportent des contraintes sur le déroulement du projet, telles le manque de soutien technique, de fiabilité, de maturité et de formation des développeurs.

Au niveau du *processus*, celui-ci peut engendrer en cours d'exécution des changements et même la création de nouvelles exigences. À titre d'exemple, suite à la revue d'un prototype avec le client, celui-ci peut demander des changements ou des ajouts qu'il n'avait pas encore envisagés jusque là. Egalement, le processus de développement du logiciel est de nature itérative, plutôt que d'être séquentielle. Le processus est aussi caractérisé par un équilibre délicat entre la créativité et le maintient d'une certaine discipline et de rigueur dans la réalisation des étapes du projet. Le guide SWEBOK (2004) souligne également que la mesure joue un rôle important dans le management de l'ingénierie du logiciel. En effet, ils soulignent que la gestion de projet logiciel sans mesure (qualitative et quantitative) est un manque de rigueur et prive le management d'indicateurs importants sur le déroulement du projet. Ainsi, un projet logiciel doit inclure un processus de prise de mesures en cours de réalisation du projet. Ce processus de mesure est décrit dans la norme ISO15939. Les techniques de mesures utilisées sont spécifiques au génie logiciel.

La gestion des *individus* a aussi des particularités pour le logiciel. Les individus doivent recevoir ou maintenir une formation sur les technologies dont le cycle de vie est très court. Ainsi, les gestionnaires de projet doivent s'assurer que la formation des individus est maintenue à jour. De plus, l'expertise pointue est difficile à obtenir, ainsi, les gestionnaires de projet doivent mettre en place des mesures pour l'acquérir et la retenir au sein de l'organisation. C'est ainsi que le gestionnaire de projet doit trouver un équilibre entre l'autonomie du personnel et conserver un processus discipliné.

Ce chapitre de SWEBOK souligne un autre aspect propre au logiciel, soit la gestion du porte folio des logiciels en développement et en exploitation dans l'organisation. Cette vue globale est non seulement nécessaire pour les arrimages entre les systèmes logiciels, mais aussi pour évaluer les impacts des changements et de détecter les possibilités de réutilisation des composants logiciels.

SWEBOK subdivise le domaine en six sections, les cinq premières traitent du processus de gestion de projet et la dernière concerne la mesure. Le processus de gestion de projet n'est pas particulier au génie logiciel, cependant, nous tenterons d'identifier et de retenir les concepts plus près des particularités du génie logiciel.

Tableau L

Concepts de la section "Initiation and Scope Definition"

Initiation and Scope Definition	
Sous-sections	Concepts
Determination & negotiation of Requirements	▪ Elicitation methods and techniques ▪ Boundaries for the tasks undertaken ▪ Validation and changes procedures
Feasibility analysis	▪ Expertise
Process for the review & Revision of Requirements	▪ Agreement between stakeholders ▪ Revised at predetermined points ▪ Traceability analysis and risk analysis

Tableau LI

Concepts de la section "Software Project Planning"

Software Project Planning	
Sous-sections	Concepts
Process planning	▪ Selection of the appropriate life cycle model ▪ Adaptation and deployment of processes ▪ Selection of tools and methods
Determine deliverables	▪ For each tasks ▪ Reuse of software components evaluated ▪ Use off the shelf software product ▪ Use of third parties
Effort, Schedule and cost estimation	▪ Tasks, inputs and outputs ▪ Expected effort based on historical size-effort data ▪ Iterative activity
Resource allocation	▪ Equipment, facilities and people ▪ Allocation of responsibilities

Tableau LI (suite)

Software Project Planning	
Sous-sections	Concepts
Risk management	▪ Software unique aspects of risk, such adding unwanted features
Quality management	▪ Quality attributes ▪ Product verification and validation
Plan management	▪ Reporting, monitoring and control

Tableau LII

Concepts de la section "Software Project Enactment"

Software Project Enactment	
Sous-sections	Concepts
Implementation of plans	▪ Resources, deliverables
Supplier contract management	▪ Prepare and execute agreements ▪ Monitor performance and accept products
Implementation of measurement process	▪ Relevant data collection
Monitor process	▪ Adherence to the various plans ▪ Outputs for eack tasks analyzed ▪ Effort, schedule and costs are investigated ▪ Measurement data are modeled and analyzed
Control process	▪ Corrective action ▪ Revision of plans ▪ Abandonment of the project ▪ Configuration Management
Reporting	

Tableau LIII

Concepts de la section " Review and Evaluation"

Review and Evaluation	
Sous-sections	Concepts
Determining satisfaction of Requirements	▪ User and customer satisfaction ▪ Variances are identified

Tableau LIII (suite)

Review and Evaluation	
Sous-sections	Concepts
Reviewing and evaluating Performance	▪ Methods, tools, and techniques are evaluated ▪ Process relevance, utility and efficacy

Tableau LIV

Concepts de la section "Closure"

Closure	
Sous-sections	Concepts
Determining Closure	▪ Requirements satisfied ▪ Stakeholders involment
Closure activities	▪ Archival of project materials ▪ Measurement data base updated

Tableau LV

Concepts de la section " Software Engineering Measurement "

Software Engineering Measurement	
Sous-sections	Concepts
Establish and sustain measurement commitment	▪ Accept requirements for measurement ▪ Commit resources for measurement
Plan the measurement process	▪ Characterize the organizational unit ▪ Identify information needs ▪ Select measures ▪ Define data collection, analysis and reporting procedures ▪ Define criteria for evaluating the information products ▪ Review, approve and provide resources for measurement tasks ▪ Acquire and deploy supporting technologies

Tableau LV (suite)

Software Engineering Measurement	
Sous-sections	Concepts
Perform the measurement process	▪ Integrate measurement procedures with relevant processes ▪ Collect data ▪ Analyze data and develop information products ▪ Communicate results
Evaluate measurement	▪ Evaluate information products ▪ Evaluate the measurement process ▪ Identify potential improvements

3.2.1.8 Software Engineering Process

Ce domaine regroupe les connaissances de la gestion de projet de génie logiciel ainsi que les mesures associées aux différents processus. La première section couvre la définition du projet et son envergure, incluant les exigences, les études de faisabilité et le processus de revue des exigences. La deuxième section est la planification du projet de développement du logiciel incluant, entre autres, les livrables, l'estimation de l'effort, des coûts et de l'échéancier. La troisième section couvre la mise en place des différents plans, de la gestion des sous-contractants, mise en place des processus de prise de mesure, de contrôle et de rapport. La quatrième section couvre la revue et l'évaluation. La cinquième section couvre les activités de fin de projet. La dernière section regroupe les connaissances reliées à la prise de la mesure tant au niveau du processus que du produit.

Tableau LVI

Concepts de la section "Process Implementation and Change"

Process Implementation and Change	
Sous-sections	Concepts
Process Infrastructure	▪ Resources (staff, tools, funding) ▪ Responsibilities assigned ▪ Infrastructure o Software engineering process group o Experience factory (models, guides, courses)
Software process management cycle	▪ Establish Process Infrastructure ▪ Planning ▪ Process implementation and change ▪ Process evaluation
Models for process implementation and change	▪ Quality improvement paradigm ▪ IDEAL model ▪ Quantitative or qualitative
Practical considerations	▪ Changes to the process ▪ Changes to the process outcomes (returning investment)

Tableau LVII

Concepts de la section "Process Definition"

Process Definition	
Sous-sections	Concepts
Software Life Cycle Models	▪ Waterfalls, prototyping, evolutionary, incremental iterative, spiral, reusable, automated software synthesis
Software Life cycle processes	▪ Process definition
Notations for process definitions	▪ Data flow diagrams ▪ Statecharts ▪ Actor-dependency modeling ▪ SADT notation ▪ Petri nets ▪ Rule-based
Process adaptation	▪ Local adaptations
Automation	▪ Tolls to support process notations

Tableau LVIII

Concepts de la section "Process Assessment"

Process Assessment	
Sous-sections	Concepts
Process assessment models	▪ Capture the good practices ▪ Maturity model (ex : CMMI)
Process assessment methods	▪ CBA-IPI : process improvement ▪ SCAMPI : CMMI

Tableau LIX

Concepts de la section "Process and Product Measurement"

Process and Product Measurement	
Sous-sections	Concepts
Process measurement	▪ Quantitative information about the process, collected, analysed and interpreted. ▪ Identify the strengths and weaknesses of processes ▪ Evaluate processes after implementation or changes ▪ Productivity of teams
Software Product Measurement	▪ Product size ▪ Product structure ▪ Product quality
Quality of measurement results	▪ Accuracy, reproducibility, repeatability, convertibility, random measurement errors
Software Information models	▪ Models using data collected and knowledge ▪ Model building o Calibration and evaluation of the model ▪ Model implementation o Interpretation and refinement of models o
Process measurement techniques	▪ Process measurement techniques o Analyze software processes and to identify strengths and weaknesses o Instruments based and judgmental ▪ Analytic and benchmarking

3.2.1.9 Outils et méthodes

Ce domaine regroupe les connaissances associées aux outils et aux méthodes utilisées en génie logiciel. La section des outils suit la structure même des chapitres du guide. La deuxième section se concentre sur les méthodes applicables au développement de logiciel.

Tableau LX

Concepts de la section "Software Engineering Tools"

Software Engineering Tools	
Sous-sections	Concepts
Software Requirements Tools	▪ Requirement modeling ▪ Requirements traceability
Software Design Tools	
Software Construction Tools	▪ Program editors ▪ Compilers & code generators ▪ Interpreters ▪ Debuggers
Software Testing Tools	▪ Test generators ▪ Test execution frameworks ▪ Test evaluation ▪ Test management ▪ Performance analysis
Software maintenance Tools	▪ Comprehension ▪ Reengineering
Software CM Tools	▪ Defect enhancement issue and problem tracking ▪ Version management ▪ Release and build
Software Engineering Management Tools	▪ Project planning and tracking ▪ Risk management ▪ Measurement

Tableau LX (suite)

Software Engineering Tools	
Sous-sections	Concepts
Software engineering Process Tools	▪ Process modeling ▪ Process management ▪ Integrated CASE ▪ Process-centered software
Software Quality Tools	▪ Review and audit ▪ Static analysis
Miscellaneous Tools Issues	▪ Tools integration techniques ▪ Meta tools ▪ Tool evaluation

Tableau LXI

Concepts de la section "Software Engineering Methods"

Software Engineering Methods	
Sous-sections	Concepts
Heuristic Methods	▪ Structured Methods ▪ Data Oriented methods ▪ Object oriented methods
Formal Methods	▪ Specification languages and notations ▪ Refinement ▪ Verification/ proving properties
Prototyping Methods	▪ Styles ▪ Prototyping target ▪ Evaluation techniques

3.2.1.10 Qualité du logiciel

Cette section regroupe les connaissances concernant la qualité du logiciel. Et elle recoupe les différentes étapes du développement. La première section élabore les fondements de la qualité du logiciel incluant, entre autres, les modèles, les coûts, les caractéristiques de qualité et l'amélioration de la qualité. La seconde section regroupe les éléments de connaissances reliés à la gestion de la qualité incluant les audits et les

revues. La dernière section présente des considérations pratiques de la qualité du logiciel.

Tableau LXII

Concepts de la section "Software Quality Fundamentals"

Software Quality Fundamentals	
Sous-sections	Concepts
Culture and Ethics	▪ To share a commitment to software quality ▪ Code of ethics to reinforce attitudes related to quality
Value and Cost of quality	▪ Prevention cost, appraisal cost, interval & external failure cost
Models and Quality Characteristics	▪ Software engineering process quality ▪ Software product quality
Quality Improvement	▪ Iterative process of continuous improvement o Management control, coordination, feedback ▪ Prevention and detection of errors ▪ Customer focus

Tableau LXIII

Concepts de la section "Software Quality Management Processes"

Software Quality Management Processes	
Sous-sections	Concepts
Software quality Assurance	▪ Processes, products and resources ▪ Assurance that software products and processes conform to their specified requirements
Verification and Validation	▪ To ensure that quality is built into software ▪ Verification : ensure that the product is built correctly ▪ Validation : ensure that the right product is built ▪ Planning o Assign roles, resources, responsibilities
Reviews and Audits	▪ Management Reviews ▪ Technical reviews ▪ Inspections ▪ Walk-through ▪ Audit

Tableau LXIV

Concepts de la section " Practical Considerations "

Practical Considerations	
Sous-sections	Concepts
Software Quality Requirements	▪ Influence factors o Domain of the system, requirements, standards, budget, staff, etc. ▪ Dependability o In case of system failure ▪ Integrity levels of software o Consequences of failure and probability
Defect Characterization	▪ Error, Fault, Failure, Mistake
Software quality management Techniques	▪ Static Techniques ▪ People-intensive techniques ▪ Analytical techniques ▪ Dynamic techniques ▪ Testing
Software Quality Measurement	▪ Management decision making ▪ Find problematic areas and bottlenecks in process ▪ Testing

3.2.2 Les disciplines frontières du génie logiciel

Le guide SWEBOK identifie huit disciplines frontières avec le génie logiciel. La figure 21 présente ces disciplines.

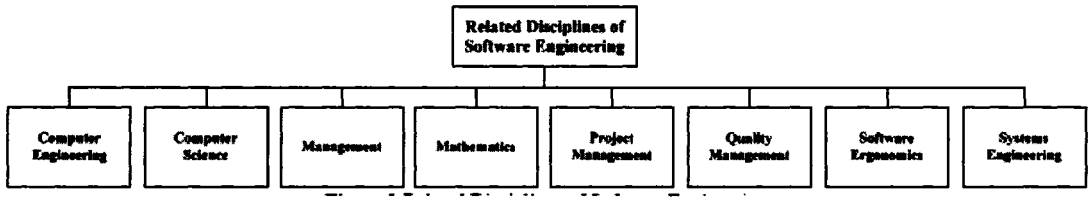

Figure 21 Disciplines frontières du génie logiciel (SWEBOK Abran et al. 2004)

Parmi les disciplines frontières, deux d'entre elles attirent plus notre attention. Le génie informatique et l'informatique sont les disciplines qui ont des frontières très importantes avec le génie logiciel et pas toujours très nettes. Ces deux disciplines ont aussi plusieurs thèmes en commun entre elles. Cependant, le génie informatique est plus marqué par l'aspect matériel des ordinateurs que l'informatique.

Afin d'obtenir certaines balises précises sur les thèmes et les concepts de l'informatique, nous détaillons cette discipline.

3.2.3 L'informatique

L'informatique est une des disciplines qui comporte des liens forts avec le génie logiciel. D'ailleurs, ces liens sont tellement étroits qu'il est parfois difficile de distinguer si un aspect particulier est du génie logiciel ou de l'informatique. Ces zones grises sont une source de discussions entre les membres des ces deux disciplines. De plus, est-ce que le génie logiciel est une sous discipline de l'informatique ou une discipline à part entière? Les discussions à ce sujet sont à la source de plusieurs écrits et conférences.

Pour les besoins de cette recherche, nous avons besoin d'identifier les principaux concepts de l'informatique. Au niveau de la revue de littérature, nous avons observé que plusieurs énoncés de principe comportaient des concepts de l'informatique. Dans le but de raffiner l'analyse de ces principes, l'identification des principaux concepts de l'informatique permettra de mieux distinguer les principes relevant du génie logiciel de ceux de l'informatique.

À l'instar du génie logiciel, les définitions de l'informatique abondent. À notre grande surprise, ni SWEBOK et ni le curriculum de l'informatique (CC2001) nous offrent une définition explicite de l'informatique en tant que discipline. Pour pallier à ce problème,

nous choisissons la définition proposée par « l'Encyclopédie de l'informatique » (Ralston et al.2000) : « *...the systematic study of algorithmic processes that describe and transform information : their theory, analysis, design, efficiency, implementation and application.* » *(p.405)*

Ralston et al.(2000) soulignent que la discipline se divise en deux grandes parties, les *applications informatiques* et les *systèmes informatiques*. Les applications informatiques regroupent les thèmes concernant le traitement de l'information et sa représentation. Les applications se subdivisent en deux groupes. Le premier regroupe les applications de nature numérique où il y a dominance des modèles mathématiques et des données numériques. Le deuxième groupe concerne les applications dites « non-numériques » où les problèmes et l'information sont représentés par des symboles et des règles. Le tableau LXV synthétise les catégories et les thèmes associés de l'informatique.

Tableau LXV

Catégories et thèmes de l'informatique (Adaptation de Ralston et al. (2000))

Applications informatiques	**Systèmes informatiques**
1 - Numériques ▪ Analyse numérique ▪ Méthodes d'optimisation ▪ Simulation ▪ Librairies de fonctions mathématiques ▪ Géométrie	1 - Systèmes logiciels ▪ Programmes ▪ Compilateurs, assembleurs ▪ Traitement des langages intermédiaires ▪ Outils de programmation ▪ Systèmes d'exploitation ▪ Gestion des réseaux télécom.
2 - Non numériques ▪ Intelligence artificielle ▪ Systèmes multimédias ▪ Traitement du langage ▪ Graphiques ▪ SGBD	2 - Systèmes matériels ▪ Design logique ▪ Organisation des ordinateurs ▪ Architecture matérielle ▪ Processeurs, mémoire ▪ Composants physiques

Les systèmes informatiques se subdivisent également en deux catégories : les systèmes logiciels et les systèmes matériels. La catégorie des systèmes logiciels traite, entre autres, des outils de programmation tels les compilateurs, les assembleurs et les systèmes d'exploitation. La catégorie des systèmes matériels traite des thèmes concernant l'organisation et l'architecture matérielle des ordinateurs incluant les configurations des processeurs, de la mémoire, des périphériques.

Ralston et al. (2000) identifient trois sphères principales d'activités : théoriques, expérimentales et de design. Les activités théoriques comprennent la conception de nouvelles théories, de nouveaux modèles et de notations pour mieux comprendre les relations entre les objets d'un domaine. Les activités expérimentales permettent de tester les théories, les modèles et les notations proposés et les raffiner. Les activités de design se concentrent sur la conception de systèmes et d'applications informatiques pour répondre aux besoins d'une organisation ou d'un domaine d'application.

Selon Ralston et al. (2000), le corpus des connaissances de l'informatique se divise en douze sous domaines. Chacun de ces sous domaines sont présentés sommairement en identifiant les concepts principaux. Ces concepts seront utiles lors de l'analyse ultérieure des principes.

Les prochaines sections présentent les concepts des 12 domaines de l'informatique. Le lecteur peut aller, à sa guise, directement à la section 3.2.4.

3.2.3.1 Les algorithmes et les structures de données

Les algorithmes représentent la séquence des étapes d'une solution à un problème. L'algorithme sera, par la suite, traduit dans un langage de programmation qui donnera les instructions à être exécutées par l'ordinateur dans une séquence donnée. L'algorithme est le point de départ de l'automatisation des tâches. L'algorithmique est

l'étude systématique des algorithmes et se divise en trois sphères d'activités : le design pour la conception des algorithmes; l'analyse pour l'évaluation de l'efficacité, de la complexité et des performances et la vérification pour la preuve du bon fonctionnement de l'algorithme.

Une structure de données est une collection organisée de valeurs et un ensemble d'opérations qu'il est possible d'effectuer sur celles-ci. Les structures de données classiques comprennent entre autres, les ensembles, les piles, les files, les listes, les arbres, les graphes, les tableaux et les articles. Le concept d'abstraction est étroitement lié aux structures de données par l'entremise de l'interface des opérations (services) qui offre une abstraction sur l'implantation physique des structures afin de faciliter leur manipulation.

Tableau LXVI

Principaux concepts du domaine des algorithmes et des structures de données (Adaptation de Ralston et al. (2000))

▪ Algorithmes ▪ Structures de donnée o Listes o Arbres o Adressage aléatoire o Graphes ▪ Programmation parallèle ▪ Algorithmes probabilistiques ▪ « Pattern-matching » ▪ Combinatoire ▪ Cryptographie	▪ Diviser pour régner ▪ Programmation dynamique ▪ Interpréteur d'état et de piles ▪ Test ▪ Algorithmes parallèles ▪ Évaluation de la complexité et de la performance ▪ Algorithme aléatoire

3.2.3.2 Les langages de programmation

Le langage de programmation est un outil que le programmeur dispose pour dicter à l'ordinateur la séquence des opérations à être exécutées. Le programmeur peut traduire

un algorithme dans un des nombreux langages de programmation existants. Les langages offrent aussi des possibilités pour organiser le code d'une solution afin d'améliorer la lisibilité et la facilité d'entretien du code. Un langage de programmation est composé de mots et de règles syntaxiques dont le programmeur doit faire l'apprentissage afin de programmer une solution. Les langages évolués offrent une couche d'abstraction sur la configuration matérielle de l'ordinateur. Le compilateur, l'interpréteur ou la machine virtuelle se chargent de la traduction des programmes en un code binaire exécutable par l'ordinateur. Les langages se classent selon leur paradigme. Ainsi, les paradigmes les plus connus sont : procédural, orienté-objet et fonctionnel. En 1999, on pouvait répertorier plus de 1000 langages de programmation commerciaux, académiques ou expérimentaux (Ralston et al. (2000)).

Tableau LXVII

Principaux concepts du domaine des langages de programmation (Adaptation de Ralston et al. (2000))

▪ Modèles d'ordinateur o Machine de Turing ▪ Génération & traduction ▪ Langages formels ▪ Sémantique, syntaxe, grammaire ▪ Expression régulière ▪ Types et objets ▪ Simulation ▪ Graphiques ▪ Compilateurs, assembleurs ▪ Analyseur syntaxique ▪ Débogueur, Parseur	▪ Type statique & dynamique ▪ Fonctionnelle ▪ Objet ▪ Procédurale ▪ Flux de données ▪ Graphiques ▪ Décomposition fonctionnelle ▪ Types abstraits de données ▪ Abstraction ▪ Traitement de donnée

3.2.3.3 Architecture des ordinateurs

L'architecture des ordinateurs est le domaine de l'informatique qui traite de l'organisation physique des composants de l'ordinateur. Ce domaine a des liens étroits

avec la microélectronique ainsi qu'avec les mathématiques. L'architecture comporte deux volets. L'architecture système se préoccupe d'offrir aux programmeurs une vue fonctionnelle et abstraite sur la configuration matérielle. L'implantation de l'architecture traite des aspects de coûts et de performances des composants matériels. L'architecture décompose l'ordinateur en trois groupes de composants : les processeurs, le stockage de l'information (mémoire, disques) et les entrées-sorties incluant les communications entre les composants de l'ordinateur. Ce domaine intègre aussi la gestion des erreurs et la tolérance aux pannes.

Tableau LXVIII

Principaux concepts du domaine de l'architecture des ordinateurs (Adaptation de Ralston et al. (2000))

▪ Logique digitale ▪ Algèbre booléenne ▪ Mathématiques discrètes et statistiques ▪ «Finite state theory » ▪ Théorie des nombres ▪ Gestion des erreurs ▪ Tolérance aux pannes	▪ Structures de contrôle ▪ Optimisation ▪ Organisation des ordinateurs ▪ Simulateurs ▪ Entrée-sortie ▪ Parallélisme ▪ « Finite state machine » ▪ Architecture multiprocesseurs

3.2.3.4 Les systèmes d'exploitation et les réseaux

Les systèmes d'exploitation sont des logiciels spécialisés dont leur but est de gérer les ressources matérielles de l'ordinateur. Ces logiciels offrent une abstraction sur l'architecture complexe des différents composants physiques d'un ordinateur en offrant une gamme de services de haut niveau qui permettent d'interagir avec l'ordinateur. Les thèmes de ce domaine sont, entre autres, les aspects de sécurité, la fiabilité, les files d'attente, l'ordonnancement, la concurrence, les inter blocages, la tolérance aux pannes

et la gestion des erreurs. Au niveau des réseaux, on y retrouve les thèmes des protocoles, la gestion distribuée, la bande passante et la nomination des ressources.

Tableau LXIX

Principaux concepts du domaine des systèmes d'exploitation et des réseaux (Adaptation de Ralston et al. (2000))

▪ Gestion de la concurrence ▪ Ordonnancement ▪ Gestion de mémoire ▪ Réseaux ▪ Analyse et modèles de performance ▪ Systèmes en temps réel ▪ Protocoles réseaux	▪ l'abstraction ▪ «Information hiding » ▪ Encapsulation ▪ «Binding » ▪ Gestion de processus et de tâches ▪ Gestion de mémoire ▪ Gestion de fichiers

3.2.3.5 Le génie logiciel

Ralston et al. (2000) identifient le génie logiciel comme un domaine de l'informatique. Le génie logiciel se concentre sur la réalisation de grands systèmes logiciels qui répondent aux besoins exprimés tout en étant fiables et sécures. Comme nous avons déjà choisi SWEBOK (2004) comme référence pour l'identification des concepts du génie logiciel, nous ne ferons pas d'identification de concepts dans cette section.

3.2.3.6 Les systèmes de gestion de bases de données

Ce domaine regroupe les thèmes reliés aux bases de données et à la gestion sécuritaire de grands ensembles de données. Les SGBD sont des logiciels qui font la gestion des informations en permettant de les rechercher, de les consulter et de les mettre à jour en en préservant leur intégrité. Les SGBD intègrent aussi des gestionnaires de transactions qui permettent de mettre à jour les données d'une façon intègre et sécuritaire. Ce domaine comporte également les thèmes de sécurité des accès aux données, l'encryptage

de l'information, l'optimisation des requêtes, la sauvegarde et le recouvrement des données.

Tableau LXX

Principaux concepts du domaine des bases de données (Adaptation de Ralston et al. (2000))

▪ Algèbre relationnelle ▪ Concurrence ▪ Sérialisation des transactions ▪ Prévention des deadlocks ▪ Synchronisation ▪ Inférence statistique ▪ Inférence basée sur des règles ▪ Techniques de tris et de recherches ▪ Indexation ▪ Analyse de performance ▪ Cryptographie ▪ Authentification	▪ Modélisation de données ▪ Fichiers ▪ Accès aux données ▪ Optimisation des requêtes ▪ Contrôle de la concurrence et du recouvrement ▪ Intégrité ▪ Sécurité ▪ Machine virtuelle ▪ Intégration multimédia

3.2.3.7 L'intelligence artificielle et la robotique

Le domaine de l'intelligence artificielle traite de la modélisation des aspects cognitifs des humains dans le but de concevoir des machines et des logiciels capables de les reproduire ou de les augmenter. Ce domaine comporte trois volets. La *psychologie informatique* fait l'étude du comportement et de l'intelligence humaine afin de concevoir des programmes qui seraient en mesure d'imiter le raisonnement humain. La *philosophie informatique* fait l'étude de la constitution d'une base de connaissances pouvant être traitées par l'ordinateur. Le dernier volet, *l'intelligence de la machine*, fait l'étude de méthodes de programmation permettant aux ordinateurs et aux robots d'exécuter des tâches dont seuls les humains étaient en mesure de faire. Ce domaine comporte aussi l'étude des thèmes suivants : la reconnaissance des sons, de la parole et du langage; des images, des patterns; de l'apprentissage et du raisonnement.

Tableau LXXI

Principaux concepts du domaine de l'intelligence artificielle et de la robotique (Adaptation de Ralston et al. (2000))

▪ Logique	▪ Robotique
▪ Règles	▪ Représentation des connaissances
▪ Connaissance	▪ Résolution de problèmes
▪ Méthodes formelles	▪ Heuristiques
▪ Méthodes de recherches	▪ Apprentissage
▪ Théories d'apprentissage	▪ Compréhension du langage
▪ Reconnaissance de la parole	▪ Réseaux de neurones

3.2.3.8 Les graphiques par ordinateur

Le domaine des graphiques par ordinateur s'intéresse aux modèles pour représenter les graphiques en deux ou trois dimensions. La réalité virtuelle et la simulation (aussi sous forme de jeux) sont des applications concrètes des théories et des modèles développés dans ce domaine. Deux volets importants composent ce domaine : la représentation (modèle interne) des images et les méthodes pour afficher efficacement en considérant les aspects de performances. De plus, les animations et l'interaction des utilisateurs avec les images sont aussi des thèmes couverts par le domaine. Les fondements théoriques s'appuient, entre autres, sur la géométrie, les mathématiques et les algorithmes.

Tableau LXXII

Principaux concepts du domaine des graphiques par ordinateur (Adaptation de Ralston et al. (2000))

▪ Géométrie	▪ « Smoothing »
▪ Caos	▪ Fractales
▪ Graphique	▪ Animation
▪ Échantillonnage	▪ Réalité virtuelle
▪ Simulation	▪ Gestion des couleurs

3.2.3.9 Les interactions personne-machine

Ce domaine traite de la coordination des interactions et des informations échangées entre l'humain et l'ordinateur. Ce domaine a des liens importants avec les graphiques et les interfaces personne-machine ainsi qu'avec les sciences cognitives. Les aspects d'ergonomie, de sécurité, d'apprentissage ainsi que les périphériques tels les souris, les claviers et autres capteurs de sens sont aussi des thèmes étudiés par le domaine. La conception des interfaces utilisateurs est un volet important de ce domaine puisqu'elles comptent pour beaucoup dans la facilité d'utilisation d'un logiciel. La réalité virtuelle est une application concrète dans laquelle les interactions personne machine jouent un rôle important.

Tableau LXXIII

Principaux concepts du domaine des interactions personne machine (Adaptation de Ralston et al. (2000))

▪ Science cognitive ▪ Analyse de risque ▪ Ergonomie	▪ Structure de données ▪ Traitement des images ▪ Panneau de contrôle

3.2.3.10 Le calcul numérique

Ce domaine fait l'étude de phénomènes scientifiques inaccessibles sans une capacité de calcul élevée offerte par des ordinateurs de grande puissance de traitement numérique. Les scientifiques et les ingénieurs utilisent la puissance de calcul pour résoudre des équations complexes et valider des modèles complexes. Sans l'aide des ordinateurs, les calculs complexes demandant de grande quantité d'itérations ne pourraient se faire par les personnes durant leur vie. Ce domaine était nommé « traitement numérique » jusqu'au début des années 1980. Les fondements de ce domaine s'appuient fortement sur les mathématiques ainsi que sur l'algorithmique.

Tableau LXXIV

Principaux concepts du domaine du calcul numérique (Adaptation de Ralston et al. (2000))

▪ Mathématiques ▪ Analyse d'erreurs ▪ Géométrie ▪ Statistiques ▪ Quantum ▪ Intégration symbolique	▪ Approximations ▪ Erreurs ▪ Stabilité ▪ Élément fini ▪ Convergence ▪ Parallélisme

3.2.3.11 Les systèmes d'information

Ce domaine traite des aspects de l'automatisation des tâches et des processus d'affaires au sein d'une entreprise ou organisme. Un système d'information se définit comme une collection de personnes, de procédures et d'équipements afin de recueillir, d'enregistrer, de retrouver, de traiter et d'afficher l'information de gestion. Comme l'automatisation des tâches et des processus d'affaires est considérée névralgique, ce domaine a des liens étroits avec d'autres disciplines telles les sciences de la gestion et le management, l'ingénierie des systèmes et les interfaces personne-machine. Les fondements de ce domaine sont, entre autres, l'organisation du travail, la gestion et la modélisation des processus.

Tableau LXXV

Principaux concepts du domaine des systèmes d'information (Adaptation de Ralston et al. (2000))

▪ Langages ▪ Systèmes d'exploitation ▪ Réseaux ▪ Bases de données ▪ Intelligence artificielle	▪ Interface personne – machine ▪ Linguistique ▪ Sciences de la gestion ▪ Sciences cognitives

3.2.3.12 La bioinformatique

La bioinformatique est un domaine en pleine émergence. Ce domaine représente une collaboration étroite de l'informatique au développement des disciplines de la biologie et de la médecine. L'informatique permet, entre autres, de représenter et de valider certains modèles de ces disciplines d'une façon visuelle et rapidement. L'étude de la chaîne de l'ADN et de ces composants est un exemple où l'informatique est d'un grand support, de même pour l'étude des structures chimiques des enzymes. De plus, l'étude de composants de mémoire organique fait l'objet de collaboration entre l'informatique et la biologie, tout comme la conception d'implants permettant de restaurer l'audition et la vue chez les humains.

Tableau LXXVI

Principaux concepts du domaine de la bioinformatique (Adaptation de Ralston et al. (2000))

▪ Algorithmes ▪ Traitement des images ▪ Graphiques ▪ Architecture des ordinateurs	▪ Structure de données ▪ Calcul numérique ▪ Combinatoire ▪ Bases de données ▪ Robotique

3.2.4 Les activités du génie logiciel (ISO/IEC 12207)

Pour faire suite à l'identification des domaines de connaissances du génie logiciel et de leurs principaux concepts, il est utile pour notre recherche d'identifier les principaux processus et activités du génie logiciel. À cet effet, nous faisons référence à la norme ISO/IEC 12207 (1995). Cette norme internationale établit un cadre commun, incluant une terminologie normalisée, sur les processus, les activités et les tâches associés au cycle de vie du logiciel. Un volet intéressant de cette norme est l'attention donnée à l'acquisition de composant logiciel ou de services. Ainsi, ceci s'ajoute aux activités

traditionnelles de développement, d'opération et de maintenance du logiciel. La norme couvre les situations suivantes :

- L'acquisition d'un système comprenant un composant logiciel
- L'acquisition ou la fourniture d'un logiciel ou d'un service logiciel
- Le développement, l'opération et la maintenance du logiciel

La norme regroupe un ensemble de processus comprenant pour chacun des activités et des tâches spécifiques à réaliser tout au long du cycle de vie. Pour les besoins de cette recherche, nous ne décrirons pas en détail tout le contenu de la norme, mais notre attention portera sur l'identification des principaux processus et sur les activités principales qui y sont associées.

La norme regroupe les processus du cycle de vie du logiciel sous trois catégories : primaire, de soutien et organisationnel. Le tableau LXXVII dresse la liste des processus selon les catégories.

Tableau LXXVII

Catégories de processus (adaptation de ISO/IEC 12207)

Processus du cycle de vie ISO/IEC 12207		
Primaires	**De soutien**	**Organisationnels**
▪ Acquisition ▪ Approvisionnement ▪ Développement ▪ Exploitration ▪ Maintenance	▪ Documentation ▪ Gestion des configurations ▪ Assurance qualité ▪ Vérification ▪ Validation ▪ Revue ▪ Audit ▪ Résolution des problèmes	▪ Gestion de projet ▪ Infrastructure ▪ Amélioration ▪ Formation

3.2.4.1 Processus primaires

La catégorie « primaires » regroupe cinq processus et 35 activités principales concernant l'acquisition, la fourniture, le développement, l'exploitation et la maintenance du logiciel. Le tableau LXXVIII suivant identifie les principales activités associées à chacun des processus primaires.

Tableau LXXVIII

Principales activités associées aux processus primaires (adaptation de ISO/IEC12207)

1. Processus d'acquistion	
▪ Démarrage ▪ Appel d'offre ▪ Préparation du contrat	▪ Suivi des fournisseurs ▪ Acceptation
2. Processus d'approvisionnement	
▪ Démarrage ▪ Préparation d'appel d'offre ▪ Contrat ▪ Planification	▪ Exécution et suivi ▪ Revue et évaluation ▪ Livraison
3. Processus de développement	
▪ Démarrage ▪ Analyse des exigences système ▪ Architecture du système ▪ Analyse des exigences logicielles ▪ Conception architecturale du logiciel ▪ Conception détaillée ▪ Programmation et tests	▪ Intégration logicielle ▪ Test de qualification de logiciel ▪ Intégration système ▪ Test de qualification du système ▪ Installation du logiciel ▪ Acceptation du logiciel

Tableau LXXVIII (suite)

4. Processus d'exploitation	
▪ Démarrage ▪ Test opérationnel	▪ Opération du système ▪ Soutien aux utilisateurs
5. Processus de maintenance	
▪ Démarrage ▪ Analyse des problèmes et des modifications ▪ Mise en place des modifications	▪ Revue et acceptation ▪ Migration ▪ Retrait du logiciel

3.2.4.2 Processus de soutien

Les processus de soutien viennent compléter les processus « primaires » en contribuant à la qualité et au succès du projet. Il y a huit processus principaux dans cette catégorie englobant 25 activités principales. Le tableau LXXIX suivant identifie les principales activités associées à chacun des processus de soutien.

Tableau LXXIX

Principales activités associées au processus de soutien (adaptation de ISO/IEC12207)

1. Processus de documentation	
▪ Démarrage ▪ Conception et développement	▪ Rédaction ▪ Mise à jour
2. Processus de gestion des configurations	
▪ Démarrage ▪ Identification des éléments ▪ Contrôle de la configuration	▪ Suivi de la configuration ▪ Évaluation de la configuration ▪ Gestion des versions et des livraisons

Tableau LXXIX (suite)

3. Processus d'assurance qualité	
▪ Démarrage ▪ Conformité du produit	▪ Conformité du processus ▪ Conformité du système de qualité
4. Processus de vérification	
▪ Démarrage	▪ Contrats ▪ Des processus ▪ Exigences, du design ▪ Code, intégration et documentation
5. Processus de validation	
▪ Démarrage	▪ Cas de tests ▪ Conformité des tests avec les exigences ▪ Effectuer les tests ▪ Valider la conformité du logiciel ▪ Tester le logiciel dans son environnement
6. Processus de revue conjointe	
▪ Démarrage	▪ Du code par rapport au design ▪ De la documentation ▪ Des tests avec les exigences ▪ De l'exécution des activités ▪ Des coûts et des échéanciers
7. Processus d'audit	
▪ Démarrage ▪ Revue de suivi de projet	▪ Revues techniques
8. Processus de résolution des problèmes	
▪ Démarrage	▪ Description du problème ▪ Analyse du problème ▪ Solution au problème ▪ Mise en place

3.2.4.3 Processus organisationnels

La catégorie des processus organisationnels regroupe les processus utilisés par l'entreprise pour structurer l'organisation des projets et les ressources humaines impliquées. Cette catégorie comprend quatre processus principaux et 14 activités principales. Le tableau LXXX suivant identifie les principales activités associées à chacun des processus de soutien.

Tableau LXXX

Principales activités associées aux processus organisationnels (adaptation de ISO/IEC12207)

1. Processus de management	
▪ Démarrage ▪ Planification ▪ Déroulement et suivi	▪ Revue et évaluation ▪ Fin du projet
2. Processus d'infrastructure	
▪ Démarrage ▪ Mise en place de l'infrastructure	▪ Mise à jour de l'infrastructure
3. Processus d'amélioration	
▪ Mise en place ▪ Évaluation	▪ Amélioration
4. Processus de formation	
▪ Démarrage ▪ Conception du matériel de formation	▪ Mise en place du plan de formation

3.3 Définition des termes concept et principe

L'une des premières constatations soulignées à la lecture des travaux antérieurs sur les principes du génie logiciel, est l'absence fréquente de définition des termes utilisés. Ainsi, en plus d'être à la source d'une certaine confusion dans l'utilisation de ces termes, il en découle également l'absence de critères qui permettraient de mieux identifier ce qu'est un principe, par exemple. Dans cette section, les termes « concept », « principe » et « loi » sont définis à l'aide d'une synthèse basée sur l'analyse de plusieurs définitions données à ces termes.

3.3.1 Qu'est ce qu'un concept?

L'encyclopédie Universalis (2002) souligne que l'être humain a essentiellement deux modes de représentation des connaissances. Le premier mode est le concret qui porte sur la connaissance en lien direct avec un objet du monde réel. L'autre mode est l'abstrait qui est représenté par la connaissance par « concepts ». Les concepts se distinguent par leur caractère abstrait et universel, en plus d'avoir la propriété d'être relativement stables. L'aspect universel est repris par Wikpedia en ajoutant qu'un même concept peut s'exprimer en plusieurs langues par des termes différents, mais conserver la même signification. Le concept permet d'unifier les représentations des objets et des idées.

Les concepts peuvent être de deux types : théorique ou observationnel. (Universalis) Les concepts théoriques concernent les propriétés non-observables tandis que les concepts observationnels concernent les objets observables de la réalité physique.

On a recensé 13 définitions utiles du terme concept qui sont répertoriées à l'annexe 1. Comme plusieurs de celles-ci partagent des points communs, on retient, en premier lieu, la définition donnée par Le Robert (2002) : « *Représentation mentale et générale d'une idée, d'un objet, d'une notion générale* ».

Bunge (2003) ajoute qu'un concept est : « *Simple idea, unit of meaning, building block of a proposition. Every concept can be symbolised by a term* »

Le concept, représenté par un terme, est le composant de base de toute proposition. Une proposition est définie comme étant « *un énoncé qui exprime une relation entre deux ou plusieurs termes* » (Le Robert 2002). Le concept serait l'unité de connaissance de base avec lequel les propositions telles, entre autres, les principes et les lois sont formulées.

La définition que nous proposons pour le terme concept est la suivante :

Une représentation mentale et générale d'une idée, d'un objet ou d'une notion. Le concept, pouvant se symboliser par un terme, est l'unité de base de composition d'une proposition.

3.3.2 Qu'est ce qu'un principe?

Le terme *principe* trouve ses origines latines dans le terme *principium* signifiant le commencement ou l'origine. À cet effet, le premier sens donné au terme principe fait référence aux origines, causes premières ou élément constituant. (Le Robert (2002), Webster (1989)). Les principes sont à la base d'une discipline, d'un phénomène ou du fonctionnement d'un objet.

Un deuxième sens donné au terme principe est « *une proposition première, posée et non-déduite* » (Le Robert (2002)). Une proposition est un énoncé qui exprime une « *relation entre deux ou plusieurs concepts* ». Ainsi, un principe serait d'abord une proposition composée de concepts. De ce fait, un concept à lui seul ne pourrait être considéré comme un principe. Une proposition non-déduite signifie qu'un principe ne peut être déduit d'un autre principe. S'il est déduit d'un autre principe, il ne pourrait alors être qualifié de fondamental. Concernant le qualificatif « fondamental », nous soulignons que

si un principe est une proposition première non déduite, il ne serait plus approprié, dès lors, d'utiliser l'expression « principes fondamentaux » qui serait alors considérée comme un pléonasme.

En plus d'être une proposition, un principe représente aussi une règle, une loi ou une vérité générale et non démontrée (Le Robert (2002), Bunge (2003), Webster (1989)). Une règle est définie comme *« une prescription pour faire quelque chose »* (Le Robert (2002)). Le sens donné au terme règle comprend trois interprétations. La première concerne toute la dimension des règles sociales ou des règlements d'une communauté. Cette interprétation est moins pertinente pour notre recherche. La deuxième concerne les règles scientifiques basées sur les lois de la physique ou de la nature. Le logiciel ne se basant pas vraiment sur ce type de lois, cette interprétation ne sera probablement pas retenue. La troisième concerne les *règles empiriques* adoptées suite à des essais fructueux en pratique. Ces règles empiriques sont à la base de l'action. Cette interprétation est pertinente puisque les principes du génie logiciel seraient à la source de l'action et à la base des pratiques. De plus, les principes peuvent être à la base des processus (activités) et des normes (Moore1998). Cette interprétation sera retenue.

Nous avons noté dans certains travaux antérieurs que le terme *loi* est utilisé à l'occasion pour signifier un principe. Également, il est fréquemment identifié comme un synonyme du terme principe (Le Robert (2002), Webster (1989)). Selon ces mêmes dictionnaires, le terme loi représente un ensemble de règles obligatoires établies par l'autorité afin d'orienter les comportements des individus dans une société. Également, le terme loi peut avoir le sens de représenter un rapport constant entre des phénomènes (lois scientifiques). Ces deux sens donnés au terme loi sont moins pertinents pour l'étude du génie logiciel. Le Robert(2002) présente un troisième sens à l'effet qu'une loi représente *« une règle impérative exprimant un idéal ou une norme. »* Ce sens est directement relié avec la notion de règle présentée précédemment.

Bunge (2003) et Nadeau (1999) ajoutent qu'une loi peut être de nature empirique. Bunge souligne que *« Most if not all of the law-statements in emergent sciences are empirical generalizations. Some of these are precursors of law-statement in the strict sense, which are typical of advanced sciences."* (p.161). Nadeau présente les lois empiriques sous deux écoles de pensée. D'une part, les *inductivistes* considèrent qu'une loi empirique est une « *généralisation vraie susceptible d'être confirmée par leurs exemples positifs* » (p.382). D'autre part, les *déductivistes* considèrent qu'une loi empirique *« prend forme d'un énoncé universel dénotant une régularité empirique et dont il est possible de déduire, en conjonction avec des conditions initiales, des conséquences observables »* (p.382). Une loi empirique n'est pas une vérité absolue au même titre qu'un principe. Ainsi, une loi empirique pourrait être contredite. De plus, une loi empirique est moins générale qu'un principe, ainsi, sa portée serait plus limitée qu'un principe. Cependant, une loi peut se généraliser et être élevée au rang de principe.

Le concept de *loi empirique* sera retenu et isolé du terme principe. À cette étape, on fait l'hypothèse que certains principes recensés pourraient être plutôt des lois empiriques. Cette hypothèse est basée sur le fait que le génie logiciel est une discipline en émergence (dans le sens de Bunge) et que beaucoup de généralisations ont été faites à partir d'observation des pratiques de l'industrie. De ces observations et généralisations, des normes, des méthodes, des lois et des principes ont été suggérés. Ainsi, nous faisons l'hypothèse que le génie logiciel a des principes, mais aussi des lois empiriques. Cependant, cette recherche se concentre toujours sur les l'identification des principes, mais laisse une ouverture à catégoriser certains principes comme étant plutôt des lois empiriques. Ainsi au niveau de la définition du terme principe, on ne retient pas comme tel le terme loi au sens large, mais on conserve l'hypothèse que le génie logiciel peut comporter des lois empiriques.

Pour revenir à la composition de la définition du terme principe, une *vérité non démontrée* trouve son sens dans un synonyme : *axiome*, signifiant une proposition vraie

indémontrable, mais évidente. Le Robert (2002) ajoute qu'un principe est une proposition non-démontrée, mais *vérifiable dans ses conséquences*. Ainsi, il serait possible d'observer dans les résultats, si un principe ou un groupe de principes a été suivi ou non. Litré (2004) affirme que *l'omission d'un principe mène à l'erreur*. Ces erreurs peuvent être observées dans les résultats d'un projet logiciel. De plus, un principe peut se vérifier par l'expérience, donc dans la pratique.

Ainsi, suite à l'exposé des divers éléments des définitions recensées, nous proposons la définition suivante pour le terme principe:

Proposition fondamentale de la discipline formulée sous forme prescriptive (règle), à la source des actions, pouvant être vérifiée dans ses conséquences et par l'expérience.

Pour la suite, nous conservons donc trois entités pour la suite de la recherche représentées par la figure 22.

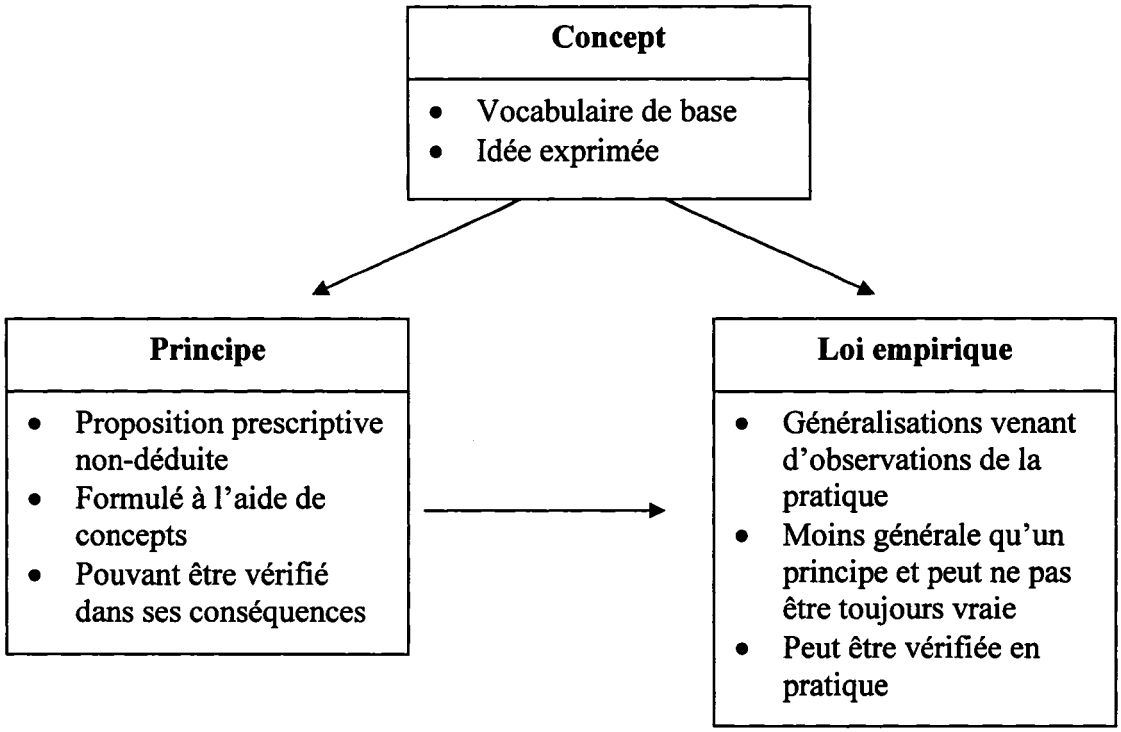

Figure 22 Relations entre les termes

3.4 Critères d'identification des principes

Les critères d'identification sont un composant majeur du cadre conceptuel de cette recherche. Les critères retenus serviront à évaluer chacun des énoncés qui ont été recensés au niveau des travaux antérieurs et de décider s'ils sont conservés ou écartés.

Parmi les travaux antérieurs, seulement un auteur et un groupe d'auteurs ont explicitement spécifié des critères d'identification d'un principe. Il est à noter que c'est seulement en 1983 qu'un auteur a suggéré les premiers critères. Ainsi, Boehm (1983) propose les deux critères suivants :

- *"They should be independent."*

- *"The entire space should be representable by combination of the basic principles."*

Le premier critère est en lien direct avec la définition proposée à l'effet qu'un principe est une proposition non-déduite. Un principe ne peut être déduit d'un autre principe, ainsi, il devrait donc être indépendant. Ce critère ne peut être appliqué lors de la première évaluation du bassin de principes, car un énoncé pris individuellement ne peut satisfaire à ce critère. Cependant, ce critère sera utilisé par la suite comme critère pour évaluer l'ensemble des principes conservés suite à la première évaluation. Cependant, ce critère aura une portée limitée, les interprétations possibles d'une proposition de principe peuvent rendre difficile d'affirmer qu'un principe est indépendant d'une façon *absolue*. Cependant, il serait envisageable de dresser une forme de hiérarchie de principes

Le deuxième critère proposé souligne que l'ensemble des principes devrait couvrir l'ensemble des activités du génie logiciel. Ce critère n'est pas un critère d'identification individuelle ou d'ensemble, mais plutôt une finalité du processus. Ainsi, ce critère ne sera pas retenu pour cette recherche. Il est possible que le groupe de principes retenus suite à l'analyse faite ne couvre pas l'ensemble des activités du génie logiciel.

Le groupe d'auteurs Bourque et al. (2002) propose huit critères d'identification des principes du génie logiciel. Nous présentons chacun des critères proposés en les commentant.

1. "*Fundamental principles are less specific than methodologies and techniques.*"
   La formulation proposée se rapproche plus d'une caractéristique (ou d'une propriété d'un principe que d'un critère d'identification. Cependant, une reformulation mineure pourrait le transformer en un critère utile. Ainsi, la formulation suivante est suggérée :
   Un principe ne doit pas être associé ou découler directement d'une technologie, ou d'une méthode.

De plus, à cette formulation, on ajoute qu'un principe ne doit pas découler d'une technique ou être une activité du génie logiciel tel que défini par la norme IEEE 12207. Cependant, une technique, une technologie, une méthode ou une activité peuvent découler d'un ou plusieurs principes.

2. "*Fundamental principles are more enduring than methodologies and techniques. Principles should be phrased in a way that will stand the test of time rather than in the context of current technology.*"
   Ce critère est lié au premier et représente également une caractéristique d'un principe ou une propriété intrinsèque d'un principe. Compte tenu de la reformulation suggérée du critère no.1, le critère no.2 est intégré à celui-ci et n'est pas retenu comme critère individuel.

3. "*Fundamental principles are typically discovered or abstracted from practice and should have some correspondence with best practices.*"
   L'objectif premier de cette recherche n'est pas de découvrir de nouveaux principes, mais d'évaluer rigoureusement plus de 300 principes déjà identifiés dans les travaux antérieurs. Ainsi, ce critère indique les sources possibles des principes et comment ceux-ci sont en relation avec les pratiques jugées les meilleures de l'industrie. Ce critère ne sera pas retenu.

4. "*Software engineering fundamental principles should not contradict more general fundamental principles.*"
   Ce critère tel que formulé n'est pas applicable. Les *« more general fundamental principles"* ne sont pas identifiés par les auteurs. Ainsi, il n'est pas possible de vérifier concrètement s'il y a contradiction entre les principes éventuellement retenus et les principes généraux non-identifiés. Le critère ne peut être retenu sous sa formulation originale. Cependant, en le reformulant, il peut être retenu comme un

critère d'évaluation de l'ensemble des principes retenus. La formulation suggérée serait : *un principe ne doit pas contredire un autre principe connu.*

5. "*A fundamental principle should not conceal a tradeoff.*"
   La formulation d'un principe ne doit pas suggérer ou imposer un dosage entre deux éléments spécifiés dans l'énoncé. Ces éléments peuvent être entre autres des attributs de qualité du logiciel. Ainsi, les compromis ne peuvent être dictés au niveau des principes, mais ils doivent plutôt être considérés au niveau du processus de développement. Ainsi, ce critère n'est pas retenu.

6. "*...but, there may be tradeoffs in the application of fundamental principles.*"
   Ce critère souligne que l'application d'un principe peut amener à faire des compromis. Ce critère ne permet pas d'identifier ou d'écarter des principes comme tel. Ainsi, il ne peut nous aider à évaluer un principe dans le cadre de la recherche. Le critère ne sera donc pas retenu.

7. "*A fundamental principle should be precise enough to be capable of support or contradiction.*"
   La formulation de ce critère est quelque peu ambiguë. Cependant, notre interprétation est à l'effet que la formulation du principe doit être assez précise pour être en mesure de tester le principe en pratique et de le vérifier dans ses conséquences pratiques. À titre d'exemple, un principe tel « abstraction » ne serait pas une formulation assez précise permettant de vérifier avec précision les conséquences du principe en pratique.

8. "*A fundamental principle should relate to one or more underlying concepts.*"
   Ce critère rejoint des éléments importants de la définition donnée au terme principe. Le principe doit être une proposition contenant un ou des concepts du génie logiciel ou du génie en général. Ainsi, c'est dans ce but que nous avons fait antérieurement

un recensement des concepts du génie logiciel tiré du guide SWEBOK. De plus, nous avons aussi identifié les principaux concepts du génie et de l'informatique. Le critère sera retenu.

Parmi les critères présentés par Boehm (1983) et Bourque et al. (2002) , nous observons deux catégories : les critères *individuels* d'identification et les critères *d'ensemble*. Le tableau LXXXI présente les critères retenus.

Tableau LXXXI

Liste des critères retenus

Critères d'identification individuels
1. Un principe est une proposition formulée de façon prescriptive • Indique quoi faire, mais sans spécifier comment le faire
2. Un principe ne doit pas être associé directement ou découler d'une technologie, d'une méthode ou d'une technique ou être une activité du génie logiciel. (adapté de Bourque et al.(2002))
3. Le principe ne doit pas énoncer un compromis (ou un dosage) entre deux actions ou concepts (Bourque et al.(2002) et de Moore (1998))
4. Un principe du génie logiciel devrait inclure des concepts reliés à la discipline ou au génie. (Bourque et al.(2002))
5. La formulation d'un principe doit permettre de le tester en pratique ou de le vérifier dans ses conséquences. (Bourque et al.(2002)) • Vérifiable dans ses conséquences et par l'expérience
Critères d'ensemble
1. Les principes devraient être indépendants (non déduit) (Boehm (1983))
2. Un principe ne doit pas contredire un autre principe connu. (Bourque et al. (2002))