UNIVERSITÉ DU QUÉBEC À MONTRÉAL

APPROCHE EXPLORATOIRE POUR LE DESIGN D'UNE MÉTHODE D'ESTIMATION EN AMONT DE LA TAILLE FONCTIONNELLE

PRÉPARATION DE L'ACTIVITÉ DE SYNTHÈSE
PRÉSENTÉ COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INFORMATIQUE DE GESTION

PAR
CHANH CHAU TRAN

MARS 1998

TABLE DES MATIÈRES

RÉSUMÉ

Explorer la faisabilité d’évaluer la taille fonctionnelle d’un produit logiciel lorsque le projet n’en est qu’aux étapes les plus en amont du développement, en d’autres termes, vérifier s’il existe un fondement qui permettrait d’évaluer la taille fonctionnelle d’un produit logiciel sur la base de documents moins volumineux que des spécifications détaillées.

INTRODUCTION

Le présent document représente la proposition d'une activité de synthèse comme exigence partielle du programme de Maîtrise en informatique de gestion à l'Université du Québec à Montréal.

La proposition porte sur une approche exploratoire pour le design d'une méthode d'estimation en amont de la taille fonctionnelle d'un produit logiciel. Elle parte de l'hypothèse qu'il existe une relation au niveau de la taille, entre l'architecture globale et les applications logicielles qui sous-tendent les activités et la mission de l'entreprise.

La structure de la proposition suit les directives du document fourni par l'Université du Québec à Montréal pour guider l'étudiant lors de l'activité de synthèse et contient les chapitres suivants :

1. La problématique
2. L'objectif de l'activité de synthèse
3. Le cadre conceptuel
4. La stratégie de recherche
5. L'échéancier

1. LA PROBLÉMATIQUE

Dans le domaine du développement des systèmes d 'information de gestion, l'évaluation «à priori » des coûts de développement des logiciels est une préoccupation majeure. Tout gestionnaire demande à savoir très tôt dans un projet « combien ça va coûter ? ».

Au début, à l'étape de l'étude d'opportunité par exemple, les nombres obtenus correspondront à des ordres de grandeur simplement; puis au fur à mesure que le problème et sa solution seront précisés le niveau de précision atteignable augmentera. La précision de l'estimation du coût de développement d'un système informatique analysé lors de l'étude d'opportunité est, dans les conditions les plus favorables, de l'ordre de +ou - 50% (Edward Yourdon 1989). Des estimations sérieuses ne sont réellement possibles qu'à partir du moment où le problème a été circonscrit par une analyse préliminaire.

1.1 Les différentes méthodes d'évaluation :

On retrouve dans la littérature une grande quantité de méthodes pour estimer le coût de développement d'un logiciel :

1. Les méthodes synthétiques sont celles dans lesquelles le projet de développement est considéré et ses coûts évalués, globalement. La plupart du temps, une personne ayant une bonne connaissance du domaine d'application, fait une estimation intuitive après une étude attentive de la description du projet.
2. Les méthodes analogiques sont celles dans lesquelles les coûts du projet considéré sont évalués par comparaison avec un ou des projets analogues déjà réalisés. Dans certains cas, notamment lorsqu'on dispose des données sur la taille, la nature et le coût d'un assez grand nombre de projets réalisés antérieurement, on peut utiliser les modèles économétriques pour évaluer a priori des coûts de nouveaux projets avec une marge d'erreur acceptable et connue.
3. Les méthodes analytiques sont celles qui sont basées sur une étude détaillée des travaux à exécuter. Après avoir identifié en détail le travail à effectuer pour réaliser le projet et l'avoir décomposé en lots de travail et en tâches, on évalue le coût d'exécution de

chaque tâche. C'est, en principe, la méthode qui donne l'évaluation la plus précise et la plus fiable des coûts du projet.

4. Les méthodes type « point de fonction » consistent dans une première phase à mesurer la taille d'un logiciel par la quantification des fonctionnalités selon le point de vue de l'utilisateur. On fait le recensement des fonctions caractéristiques (entités internes et externes, transactions d'entrée et de sortie, requêtes) du système informatique envisagé en les classant selon les niveaux de complexité. À chaque composante recensée, on impute un certain nombre de points selon son type et son niveau de complexité et on fait le cumul des points pour l'ensemble du système. La taille ou volume de l'application étant identifié, on doit dans une deuxième étape estimer combien de temps et d'effort seront requis pour développer le produit ainsi quantifié.

1.2 Les contraintes des méthodes existantes

1. Méthodes synthétiques : elles sont peu coûteuses, mais très risquées. De plus, elles ne sont pas auditables, ni vérifiables et les marges d'erreurs sont inconnues.
2. Méthodes analogiques : on doit compter sur l'existence d'un projet témoin et doit connaître non seulement le coût global de ce projet mais aussi son coût en détail. Le temps et les ressources pour procéder à son évaluation peuvent être élevés.
3. Méthodes analytiques : charge d'évaluation très importante car on doit procéder à une analyse relativement détaillée de tout le travail à effectuer. Lorsqu'elle est utilisée très tôt dans le cycle de développement, cette méthode amène nécessairement une disparité entre le nombre d'unités d'ouvrages projeté et celui qui émerge progressivement des étapes subséquentes.
4. Méthode « points de fonction » : elle n'est pas encore très utilisée dans la plupart des entreprises et nécessite un important référentiel des ratios de charge sur des projets réels, ce qui demande un investissement non négligeable en temps et ressources humaines.

Ces méthodes d'estimation utilisent toutes un modèle à deux niveaux (Heemstra, 1989). Le premier niveau consiste à estimer la taille et la complexité du logiciel tandis qu'au deuxième niveau, on estime combien de temps et d'effort qu'il faudra allouer pour développer le

logiciel en question. Au niveau de l'estimation de la taille, on est actuellement confronté au dilemme suivant :

- On gagne en temps en faisant une estimation en amont du développement de la taille du logiciel; cette estimation par contre ne donne seulement qu'un ordre de grandeur.
- On augmente la précision de l'estimation de la taille après avoir préparé les spécifications détaillées, ce qui implique un investissement important en temps et ressources humaines. Il est d'ailleurs reconnu que les spécifications fonctionnelles représentent entre 15 à 40% de l'effort total consacré à un projet de développement.

2. OBJECTIFS DE LA RECHERCHE

La motivation principale de ce travail de recherche est d'explorer la faisabilité d'estimer la taille fonctionnelle d'une application tôt dans le cycle de développement, en amont de l'analyse préliminaire, en se basant sur une composante importante du processus de planification informatique soit l'architecture des systèmes d'information .

Il a comme premier objectif de développer les règles et procédures à suivre pour faire la jonction entre les macro-élements (fonctions et entités) de l'architecture des systèmes d'information et les systèmes cibles dont la taille est mesurée selon la technique des points de fonction. Selon Zachman, 1987, à cause de la complexité croissante de l'implantation des systèmes d'information, on doit construire une architecture pour définir et contrôler les interfaces et l'intégration de toutes les composantes du système. L'architecture des systèmes d'information apparaît comme l'élément unificateur de l'ensemble des systèmes d'information de l'entreprise et est composé d'un ensemble de modèles conceptuels qui décrivent les fonctions, les entités ainsi que les relations entre les fonctions et entités. La matrice représentant ces relations permet d'identifier les systèmes d'information requis par l'entreprise pour accomplir sa mission.

L'objectif secondaire serait d'établir le type de relation qui pourrait exister entre les macro-éléments (fonctions et entités) de l'architecture des systèmes d'information et les différentes composantes des systèmes informatiques telles que mesurés par la technique des points de fonction.

C'est le degré de qualité de cette relation qui nous indiquera s'il est faisable d'estimer la taille fonctionnelle d'une application logicielle sur la base de documents moins volumineux que des spécifications détaillées.

2.1 Les acteurs intéressés

- Le client ou maître d ’ouvrage, qui souhaite établir le cadre budgétaire d’un projet de développement, en relation des objectifs et des fonctionnalités pré-établis.
- Les gestionnaires qui ont à prendre des décisions sur le portefeuille de développement d’applications
- Le fournisseur ou maître d’œuvre, appelé à prendre des engagements forfaitaires
- Les chercheurs

3. LE CADRE CONCEPTUEL

La problématique posée par cette activité de synthèse demande de passer en revue la littérature de deux domaines différents :

1. La planification informatique et le modèle d'entreprise
2. Les méthodes d'estimation de la taille d'un produit logiciel.

3.1 La planification informatique et la construction de l'architecture des systèmes d'information

On retrouve dans la littérature une grande quantité de méthodes pour aider à mieux planifier et gérer les ressources et le développement des systèmes d'information. Mintzberg, 1994, définit la planification comme « une procédure formelle pour produire un résultat articulé, sous la forme d'un système intégré de décisions ».

Selon Lederer (1988) et Earl (1993) la planification informatique est le processus de gestion par lequel l'entreprise définit les systèmes informatiques à développer, les range par priorité, définit les ressources qui seront allouées à la réalisation du plan et l'infrastructure technologique et organisationnelle à mettre place pour permettre cette réalisation.

Un des objectifs de la planification informatique, selon Pyburn, 1983, et reconnu par de nombreux auteurs est d'exploiter les possibilités des technologies de l'information et de définir le portefeuille d'applications.

McLean et Soden (1977) ont proposé quatre types de planification de 1'informatique:

la planification stratégique,
la planification à long terme (conceptuelle),
la planification à moyen terme (de gestion),
la planification à court terme (opérationnelle).

Em-Dor et Segev (1978), ont identifié six variables de planification:

la stratégie de développement,

la mission de 1'informatique,
les priorités,
les fonctions d’application,
les buts des fonctions,
les spécifications de besoin.

Pour Lientz et Chen (1980), il y a trois niveaux de planification:
la planification à long terme,
la planification stratégique,
la planification d'action.

Selon King et Srinivasan (1981), il existe deux phases principales de planification de 1'informatique:
la planification stratégique,
la planification de système.

Bowman, Davis et Wetherbe (1981; 1983), ont développé un modèle de planification à trois niveaux:
la planification stratégique,
1'analyse des besoins organisationnels en information,
l’affectation des ressources.

Thistlethwaite (1985), a énuméré trois tâches d'une planification intégrée:
la planification stratégique de 1'information,
la planification stratégique de la technologie,
la planification de projets à long terme.

Hufnagel (1987), a identifié trois approches de planification:
la planification stratégique,
le contrôle de gestion,
la planification opérationnelle.

Enfin, pour Henderson et Sifonis (1988), il y a trois phases de planification:

la planification stratégique des affaires,

la planification stratégique de l'informatique,

le plan d'action et 1'affectation des ressources.

Ce bref survol de la littérature permet de distinguer divers niveaux de planification de l'informatique. Au sommet de la pyramide il y a une planification à un niveau stratégique tandis qu'à l'autre extrême, on reconnaît une planification à un niveau plutôt opérationnel.

Earl, 1989, a étudié le changement d'emphase de la planification informatique et identifié cinq étapes dans le processus, telles que présentées dans le tableau 1 ci-dessous (page 23, thèse de Doctorat en Sciences de Gestion de M. Éric Lefebvre)

Étape	1	2	3	4	5
Tâche Principale	Identification des applications informatiques	Définition des besoins d'affaires	Planification détaillée des systèmes d'information	Stratégie par avantage compétitif	Relation à la stratégie d'affaires
Objectif clé	Compréhension de la direction	Accord sur les priorités	Équilibrage du portefeuille	Poursuite des opportunités	Intégration des stratégies des systèmes d'information et des affaires
Direction	Dirigé par l'informatique	Initiative des gestionnaires principaux	Informaticiens et utilisateurs ensemble	Direction générale, gestionnaires principaux, utilisateurs	Coalition d'utilisateurs, de gestionnaires et d'informaticiens
Approche principale	Développement ascendant	Analyse descendante	Descendante et ascendante	Innovation des utilisateurs	Méthode multiple simultanée

Tableau 1 : les 5 étapes de l'évolution de la planification.

La plupart des entreprises ont compris dès les années 80, le besoin d'adopter une approche stratégique pour gérer les investissements au niveau des systèmes informatiques et de la technologie de l'information.

Les méthodologies d'architecture des systèmes d'information ont été élaborées pour faire la jonction entre la planification stratégique et les méthodologies de développement

d'applications. C'est un ensemble de modèles conceptuels qui décrivent les fonctions, les entités ainsi que les relations entre les fonctions et les entités. Ces dossiers d'architecture ne donnent pas par contre une vue de l'ensemble des besoins des différents projets de l'entreprise, ce qui a souvent amené un dédoublement des ressources et leur incohérence entre elles.

Pour remédier à cette situation, plusieurs firmes dont IBM , 1981, proposent une méthode d'ingénierie de l'information qui permet de concevoir de façon cohérente l'ensemble des systèmes d'information de l'entreprise. La méthode BSP « Business Systems Planning » est selon Hackatorn et Karimi, 1988, une des principales méthodes couvrant la planification. Elle est reprise dans ses grandes lignes par la plupart des autres méthodes de planification informatique, comme « Information Engineering » de James Martin, 1990, et les méthodes des grandes firmes de consultants en Amérique du Nord.

BSP, Information Engineering et les autres méthodes analogues proposent toutes de construire en premier lieu une architecture globale qui identifie les applications-cible à partir du modèle d'entreprise.

3.2 Le bilan de la planification informatique

Dans une étude conduite auprès de 80 entreprises aux États-Unis en 1988, et mise à jour en 1992, Lederer et Sethi ont trouvé que la satisfaction des gestionnaires, quant aux résultats de la planification informatique est faible :

- seulement 24% des projets planifiés ont été entrepris,
- 38% des projets démarrés n'ont pas été planifiés,
- moins de 50% des changements recommandés ont été effectués.

De plus, la recherche de Lederer et Sethi démontre que finalement les coûts et la durée apparaissent comme le facteur-clé de la planification. Des projets de planification longs et coûteux courent le risque de perdre l'implication de la Direction Générale pour implanter les plans proposés.

3.3 Le modèle générique d'entreprise

Dans son rapport de thèse de doctorat, monsieur Éric Lefebvre propose que pour avoir les meilleures chances de réussir la planification informatique, « on doit améliorer l'approche informatique par l'utilisation d'un modèle générique des systèmes d'information de l'entreprise et l'approche politique par le développement du partenariat entre le service informatique et ses services utilisateurs. ». L'approche informatique concerne les activités formelles de la planification informatique, soit :

1. la construction d'une architecture des systèmes d'information, à partir des fonctions d'affaires, processus, activités et données de l'entreprise,
2. le positionnement des systèmes existants par rapport à cette architecture,
3. l'identification des besoins des gestionnaires en systèmes d'information,
4. l'élaboration du plan

Selon ce chercheur, une des difficultés dans l'élaboration du plan informatique et dans son implantation est l'évaluation du coût des projets, compte tenu des éléments d'analyse dont on dispose alors et constate qu'une des composantes indispensables du plan informatique, soit la construction de l'architecture, couvre la moitié des activités du processus de planification.

Pour réduire le temps de construction de l'architecture globale, ce chercheur propose d'utiliser un modèle générique d'entreprise pour déduire rapidement une première maquette de l'architecture des systèmes. C'est un modèle général que l'on peut adapter facilement à différentes types d'entreprises et qui comprend les mêmes objets constituant l'architecture cible requise par les méthodes usuelles, soit les fonctions et entités.

Monsieur Lefebvre s'est basé sur la méthode BICS (Business Information Characterization Study) développée initialement par IBM pour des besoins internes pour améliorer le modèle générique en incluant 76 fonctions et 46 entités regroupés en cinq domaines d'affaires. Ce chercheur industriel a fait l'évaluation de ce modèle générique auprès de quatre entreprises québécoises (Télébec, la STCUM, la Ville de Montréal, Le Curateur public du Québec) et a pu dégager cinq propositions dont trois sont pertinentes dans le cadre de cette activité de synthèse :

1. Le modèle générique permet de construire l'architecture des systèmes d'information « beaucoup plus rapidement » que ne le permettent les méthodes usuelles.
2. La qualité de l'architecture finale est dans tous les cas équivalente à celle obtenue par les méthodes usuelles.
3. Le modèle générique permet de créer rapidement une première maquette d'architecture de systèmes d'information, facile à faire évoluer.

L'architecture des systèmes d'information pouvant être construit rapidement par le modèle générique, il reste dans un deuxième temps de déterminer s'il existe un fondement qui permettrait d'estimer la taille fonctionnelle d'un produit logiciel à partir de cette architecture.

3.4 Les méthodes d'estimation de la taille d'un produit logiciel.

On retrouve dans la littérature quelques types de mesure pour déterminer la taille d'un logiciel, les plus fréquemment cités sont les lignes de code et les points de fonction.

3.4.1 Lignes de code

Il existe plusieurs modèles d'estimation de coût basés sur les lignes de code. Ces modèles n'offrent pas une bonne fiabilité à cause d'un certain nombre de problèmes associés avec l'utilisation des lignes de code comme unité de mesure de la taille d'un logiciel :

- Il n'y a pas de définition universellement acceptée ce qu'est une ligne de code
- Il n'est pas possible de comparer directement les projets de développement qui utilisent différents langages de programmation car les lignes de code sont dépendants du langage de programmation.
- Il est difficile d'estimer les lignes de code requises pour développer un système à partir du dossier des spécifications et exigences.

3.4.2 Les points de fonction

Dans la fin des années 70, Allan Albrecht d'IBM définissait les concepts qui permettaient de mesurer la productivité des projets de développement de logiciel. C'est une technique normalisée qui mesure la taille d'un logiciel par la quantification des fonctionnalités selon

le point de vue de l'utilisateur. Les points de fonction sont basés sur une vue logique et non pas technique des fonctionnalités du logiciel.

En 1983, Symons et Zwanzig proposent d'utiliser les PF dans les modèles de production et d'estimation et en 1984 a eu lieu la première rencontre à Toronto de l'IFPUG (International Function Point Group).

L'emploi des points de fonction pour mesurer la taille fonctionnelle d'un système d'information a considérablement augmenté dans la décennie passée parce qu'ils semblent répondre à trois besoins fondamentaux exprimés par les entreprises :

- mesurer un logiciel selon les fonctionnalités demandés par l'utilisateur
- obtenir les mesures de façon la plus indépendante possible du contexte technologique dans lequel les applications seront développées et utilisées
- utiliser les mêmes mesures pour faire une estimation de l'effort et de la durée d'un projet de développement ou de maintenance.

Initialement les règles de comptage des points de fonction étaient vagues (qu'est-ce qu'une entrée, une sortie? etc.) mais l'IFPUG a produit en 1992 un ensemble de règles pour clarifier et standardiser le comptage et les démarches ont été faites depuis 1993 pour que les points de fonction deviennent une norme ISO dans les années à venir.

Au Québec, le Laboratoire de recherche en gestion de logiciel de l'UQAM a développé une expertise reconnue dans cette technique avec les travaux de monsieur Alain Abran (Thèse de doctorat : Analyse du processus de mesure des points de fonction, 1994) et de monsieur Jean-Marc Desharnais (Analyse statistique de la productivité des projets de développement en informatique à partir de la technique des points de fonction, Rapport d'activité de synthèse, UQAM, Décembre 1988).

Au niveau des techniques de mesure pour estimer la taille d'un logiciel, Heemstra et Kusters ont testé et confirmé l'hypothèse que les lignes de code ne donnent pas une

indication fiable quant à la taille d'un produit logiciel par rapport à la méthode des points de fonction (Journal of Information Systems, vol.1, No 4, pp 229-237, 1991) .
De plus, il apparaît que les points de fonction répondent mieux aux besoins des gestionnaires parce qu'ils peuvent être utilisés relativement tôt dans la phase de développement pour faire l'estimation de la taille d'une application.

Récemment, lors de la conférence de l'IFPUG en Septembre 1997 à Scottsdale, Arizona, un des conférenciers, Roberto Meli propose une nouvelle technique (Early and Extended Function Point) qui permettrait d'estimer la taille d'une application en points de fonction encore plus tôt dans le cycle de développement, c'est-à-dire tout de suite après l'étude de faisabilité.

La méthode consiste à identifier les éléments du logiciel à développer tels que les fonctionnalités et les données à différents niveaux de détails. Une fonctionnalité d'un logiciel est définie comme un ensemble d'actions exécutées par le système informatique pour permettre à l'utilisateur de réaliser un objectif d'affaire et peut être divisée en quatre catégories :

- La primitive fonctionnelle qui permet à l'utilisateur d'atteindre un objectif d'affaire unitaire au niveau opérationnel. Selon le point de vue de l'utilisateur, il n'y a pas lieu de décomposer plus loin cette primitive qui est de 3 types : primitive d'entrée, de sortie et d'interrogation.
- La micro-fonction qui est constituée de 4 primitives fonctionnelles : création, lecture, mise-à-jour et suppression d'une donnée élémentaire dans un ou plusieurs fichiers logiques.
- La fonction qui est un ensemble de primitives fonctionnelles pouvant être assimilé à un sous-système opérationnel de l'application et qui permet à l'utilisateur d'atteindre un objectif d'affaire de niveau supérieur.
- La Macrofonction qui est un regroupement de fonctions pouvant constituer un sous-système du système d'information de l'entreprise (par exemple, système comptable, système de paie, etc.)

Petite	Jusqu'à 11 primitives fonctionnelles
Moyenne	De 12 à 18 primitives fonctionnelles
Grande	De 19 à 25 primitives fonctionnelles

Classification de la taille d'une fonction

Petite	Jusqu'à 3 fonctions
Moyenne	De 4 à 7 fonctions
Grande	De 8 à 12 fonctions

Classification de la taille d'une Macrofonction

Chaque objet ainsi identifié est associé à trois valeurs en termes de fonction point (minimum, moyen, maximum) selon des observations empiriques et en tenant compte de ce qu'on appelle des « bonnes règles » en génie logiciel.

Les données doivent être aussi classées sur une échelle à cinq niveaux de complexité générale, dont les trois premiers niveaux correspondent à ceux définis par les règles de comptage de l'IFPUG.

Une fois que la typologie et la complexité des fonctions et données identifiées, l'auteur utilise les tables de conversion pour déterminer le nombre de UEFP (Unadjusted Early Function Points).

4. STRATÉGIE DE RECHERCHE

Rappelons que la motivation principale de ce travail de recherche est d'explorer la faisabilité d'estimer la taille fonctionnelle d'une application tôt dans le cycle de développement, en amont de l'analyse préliminaire, en se basant sur une composante importante du processus de planification informatique soit l'architecture des systèmes d'information .

La stratégie consiste à développer dans la première phase les règles et procédures à suivre pour faire la jonction entre les macro-éléments (fonctions et entités) identifiés dans l'analyse fonctionnelle et les systèmes cibles dont la taille est mesurée selon la technique des points de fonction.

Dans la deuxième phase, nous établirons le type de relation qui pourrait exister entre ces macro-éléments (fonctions et entités) et les différentes composantes des systèmes informatiques telles que mesurés par la technique des points de fonction.

Dépendemment de la qualité des relations établies dans la phase antérieure, nous pourrons dans la troisième phase, utiliser le même processus mais avec les macro-éléments (fonctions, entités) identifiés lors de l'analyse préliminaire, ce qui se situe à un niveau antérieur dans le cycle de développement.

Finalement, à la dernière phase, il y a lieu d'établir le type de relation qui pourrait exister entre les macro-éléments (fonctions et entités) identifiés à partir d'un modèle générique des systèmes d'information de l'entreprise et les différentes composantes des systèmes informatiques telles que mesurés par la technique des points de fonction.

Dans le cadre de cette activité de synthèse, l'étude exploratoire portera sur les trois premières phases.

4.1 La démarche de l'activité de synthèse

Elle consiste à choisir au sein d'une même entreprise entre 6 à 12 applications en informatique de gestion dont les points de fonction sont comptabilisés et dont tous les niveaux de détails de mesure sont disponibles. À partir de fonctions d'affaires, processus, activités, données et architecture globale d'entreprise, nous établirons la correspondance entre les blocs fonctions/entités formant les grands systèmes d'information et les systèmes dont les points de fonction ont été comptabilisés.

La deuxième étape consiste à déterminer les règles et procédures pour attribuer un pointage à chaque bloc fonctions/entités formant les grands systèmes d'information de l'entreprise.

Par la suite nous utiliserons le test chi-carré pour identifier la qualité de relation entre les pointages attribués aux blocs fonctions/entités formant les grands systèmes d'information de l'entreprise et les points de fonction déjà comptabilisés pour ces mêmes systèmes.

4.2 Méthode de cueillette de données

4.2.1 Sélection du site

La Régie des rentes du Québec a été sélectionnée pour les raisons suivantes :

- Selon M. Jean-Marc Desharnais, consultant de la firme SELAM qui a récemment effectué un comptage exhaustif des points de fonction du système opérationnel de la RRQ, cet organisme possède une documentation très complète.
- Les points de fonction sont disponibles pour les treize applications qui permettent d'opérer la Régie des Rentes du Québec.
- L'organisme accueille favorablement l'entente de collaboration (Annexe)

4.2.2 Sélection des applications

- La Régie est responsable de l'administration de trois lois : la loi sur le Régime de rentes, la loi sur les Allocations d'aide aux familles et la loi sur les Régimes complémentaires de retraite. Le système opérationnel de la RRQ comprend en tout seize sous-systèmes qui représentent la population qui nous intéresse. La taille fonctionnelle de ces sous-systèmes est la suivante : (Annexe)

• Pensions		2,541 points de fonction
- Traitement	615	
- Suivi	720	
- Calcul	527	
- Paiement	654	
- Support	195	
• Registre des cotisants		1,461
• Fichier d'inscription de la clientèle		397
• Allocation d'aide aux familles		917
• Régimes complémentaires de retraite		406
• Assignation des dossiers		237
• Agenda journal		145
• Gestion des dossiers		229
• Lettre modèle		190
• Historique des interventions		113
• Description des employés et autorisations		249
• Programme de service		108
Total		6,693

Dans le cadre de l'activité de synthèse, nous choisirons huit sous-systèmes dont la taille fonctionnelle peut être catégorisée de cette façon :

- Petite : entre 100 à 200 points de fonction (sous-système « Support »)
- Moyenne : entre 200 à 400 points de fonction (sous-systèmes « Fichier d'inscription de la clientèle » et « Description des employés et autorisations »)
- Grande : plus de 400 points de fonction : sous-systèmes « Traitement, Suivi, Calcul, Paiement »

Ces huit sous-systèmes totalisent 3416 points de fonction soit 51% du total en taille fonctionnelle de la population.

- Collecte des documents qui serviront à faire la jonction entre le modèle fonctionnel de l'entreprise et les six applications sélectionnées dans l'architecture des systèmes (architecture fonctionnelle d'entreprise, processus opérationnel RRQ, architecture des traitements, architecture des données etc.)

4.2.3 Analyse des données

- Construire la matrice fonctions/entités en regroupant les fonctions et les blocs d'entités selon leur affinité logique. Les fonctions constituent les rangées de la matrice et les entités les colonnes.

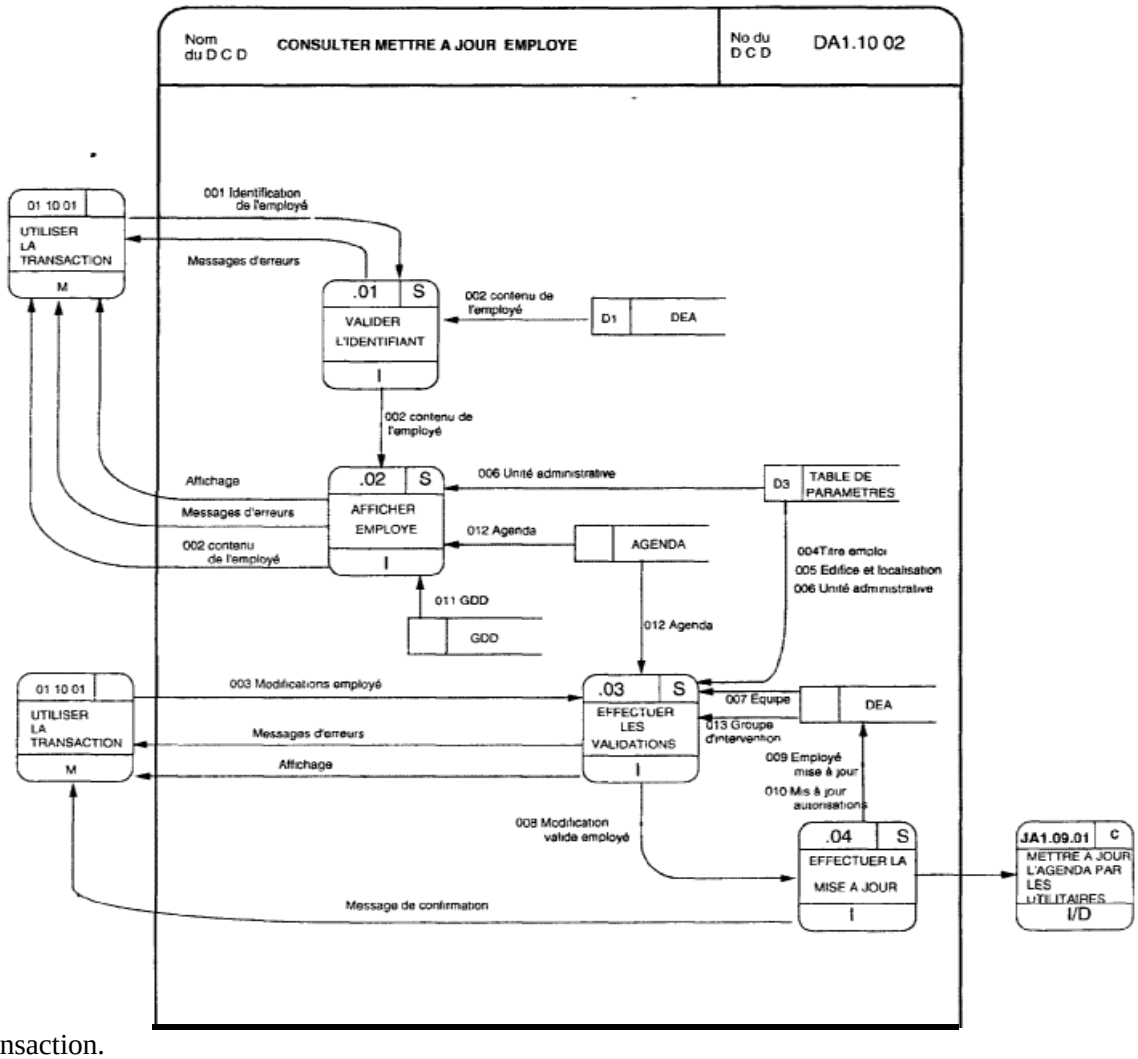
		Entités	Employé RRQ	Emploi RRQ	Suspension assignation	Complexité Assignation	Autorisation Accès agenda journal	Équipe
Fonctions								
DA1.01.10	Consulter et mettre à jour les données d'un employé							
DA1.01.15	Épurer la banque des données descriptives							
DA1.01.20	Consulter et mettre à jour les autorisations d'accès à l'agenda							
DA1.01.30	Produire les rapports							
Da1.01.40	Consulter et mettre à jour les équipes							
DA1.01.50	Gérer les groupes d'intervention							
DA1.01.60	Consulter les employés d'un groupe d'intervention							

Tableau 2 : Exemple de construction de la matrice fonctions/entités du sous-système Description des employés et autorisations

- Indiquer par un « u » l'élément de la matrice pour lequel la fonction utilise l'entité
  - Dans le cas d'une lecture seulement on peut avoir une interrogation ou une sortie. On peut alors prendre une hypothèse de sortie et interrogation moyenne, ce qui équivaut à 3-4 points à 6-7 points selon les tables PF.
- Indiquer par un « c » l'élément de la matrice pour lequel la fonction crée l'entité correspondante.

- Dans ce cas, il y a un CRIM c'est-à-dire changer, retirer, interroger et modifier ce qui fait entre 12 et 24 points selon les tables PF

- Cette opération se fait à partir du diagramme du cheminement des données du dossier fonctionnel pour chaque fonction.
- Totaliser le nombre de points pour chaque bloc fonctions/entités formant les sous-systèmes d'information de l'entreprise. Ce pointage représente la composante



transaction.

Figure 4.1 Diagramme de cheminement des données de la fonction « Consulter et mettre à jour les données d'un employé »

- Utiliser le test chi-carré (Rickmers et Todd, 1967 p. 108) pour identifier la qualité de relation entre les pointages attribués aux blocs fonctions/entités formant les sous-systèmes d’information de l’entreprise et les points de fonction déjà comptabilisés pour ces mêmes systèmes.

5. ÉCHÉANCIER

5.2.1 Phase 1 (Octobre 1997)

- Choisir l'entreprise avec les six à douze applications dont les points de fonction sont comptabilisés. La Régie des rentes du Québec a été sélectionnée.

5.2.2 Phase 2 (Novembre, Décembre 97)

- Contacter la RRQ et obtenir une entente de collaboration. (Voir Annexe 1)
- Collecte des documents pertinents à l'activité de synthèse (architecture fonctionnelle d'entreprise, processus opérationnel RRQ, architecture des traitements, architecture des données etc.)

5.2.3 Phase 3 (Janvier 98, Mars 98)

- Codification, analyse et traitement des données

5.2.4 Phase 4 (Avril, Juillet 98)

- Rédaction

RÉFÉRENCES BIBLIOGRAPHIQUES

Abran, A., *Analyse comparative de la fiabilité des points de fonction comme modèle de productivité*, Revue de Liaison de la recherche en informatique cognitive des organisations [ICO], vol. 4, pp. 16-24, 1993.

Abran, Alain, « *Analyse du processus de mesure de points de fonction* », Thèse de doctorat, École Polytechnique de Montréal, Mars 1994, 405 pages

Abran, A., Paton, K., *A Formal Notation for the Rules of Function Points Analysis*, Submitted to ACM-Transactions on Software Engineering and Methodology, 1996.

Abran, A., Robillard, P.N., *Function Points: A Study of their Measurement Processes and Scale Transformations*, Journal of Systems and Software, vol. 25, pp. 171-84, 1994.

Abran, A, Robillard, P.N., *Function Points Analysis : An Empirical Study of its Measurement Processes*, Accepted for publication in IEEE Transactions on Software Engineering, 1996.

Albrecht, A.J., *Measuring Application Developement Productivity*, presented at IBM Applications Development Symposium, Monterey, CA, 1979.

Albrecht, A.J., *Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation*, IEEE Transactions on Software Engineering, vol. SE-9, pp. 639-648, 1983.

Boehm, B.W, *Software engineering economics*, Édition : Englewood Cliffs, N.J. : Prentice-Hall, c1981.

Conte, S.D., Dunsmore, H.E., Shen, V.Y, *Software engineering metrics and models*, Menlo Park, Calif. ; Don Mills, Ont. : Benjamin/Cummings, c1986.

Desharnais J-M, *Analyse statistique de la productivité des projets de développement en informatique à partir de la technique des points de fonction*, Rapport d'activité de synthèse, UQAM, Décembre 1988.

Desharnais, J.-M., Abran, A., *Implanter un programme de mesures en génie logiciel : des étapes pour éviter les risques d'échec*, L'expertise informatique, vol. 1, pp. 2-6, 1995.

Desjardins, D., *L'architecture globale, pourquoi et comment?*, mémoire d'activité de synthèse présenté en exigence partielle de la maîtrise en informatique de gestion, Montréal : Université du Québec à Montréal, 1992.

Heemstra, F.J., Kusters, R.J., *Function Point Analysis: Evaluation of a Software Cost Estimation Model*, European Journal of Information Systems., vol. 1, pp. 229-237, 1991.

Hendren, G., *La planification stratégique de l'informatique : survol, synthèse et réflexion personnelle*, Montréal : Université du Québec à Montréal, 1990.

Jacquet, J-P., Abran, A., *Conception d'un guide de validation des méthodes de mesure de taille fonctionnelle*, presented at Neuvième journées internationales « Le génie logiciel et ses applications » (GL96), Paris, 1996.

Jenkins, A.M., *MIS design variables and decision making performance : a simulation experiment*,Édition : Ann Arbor, Mich., UMI Research Press, c1983.

Jones, C., *The role of function point metrics in the 21st century*, The Voice 1996 - A publication of the International Function Point Users Group, vol. 1, pp. 32-34, 1996.

Kemerer, C.F., *Measurement for Improved Software Development*, in Conference at the Computer Research Institute of Montreal (CRIM). Montréal, Canada, 1992, pp. 14.

Kemerer, C.F., *An empirical validation of software cost estimation models*, Communications of the ACM, vol. 30, pp. 406-29, 1987.

Kitchenham, B., *Using Function Points for Software Cost Estimation - Some Empirical Results,* presented at Tenth Annual Conference of Software Metrics and Quality Assurance in Industry, Amsterdam, 1993.

Kitchenham, B., *Cost estimation myths and Facts*, National Computing Center (NCC-UK), 1992, pp. 1-12.

Lefebvre, É., *Évaluation d'un modèle générique pour la construction de l'architecture des systèmes d'information de l'entreprise,* mémoire présenté comme exigence partielle de la maîtrise en administration des affaires, Montréal : Université du Québec à Montréal, 1993.

Londeix, B., *Cost estimation for software development***,** Wokingham, Angleterre ; Don Mills, Ont. : Addison-Wesley, 1987.

Matson, J.E., Barrett, B.E., Mellichamp, J.M., *Software development cost estimation using function points*, IEEE Transactions on Software Engineering, vol. 20, pp. 275-87, 1994.

Parker, M.M, Benson, R.J., *Information economics : linking business performance to information technology*, Marilyn M. Parker and Robert J. Benson ; Englewood Cliffs, N.J. : Prentice-Hall, 1988.

Rickmers & Todd, *Statistics, An introduction*, McGraw-Hill, 1967.

St-Pierre, D., Desharnais, J.-M., Abran, A., and Gardner, B., *Definition of When Requirements Should be Used to Count Function Points in a Project*, in The Voice 1996 - A publication of the International Function Point Users Group, vol. 1, pp. 6-7, 22, 1996

Symons, C.R., *Mark 2 Function Points for Productivity Measurement & Estimating*, in International Conference on Project Management, Planning & Estimation. London: Nolan, Norton & Co., 1990, pp. 15.

Ward, J., Griffiths, P.M., Whitmore, P., *Strategic planning for information systems*, Chichester ; Toronto : J. Wiley, c1990.