APPLICATION DE LA MÉTHODE FFP À PARTIR D'UNE SPÉCIFICATION SELON LA NOTATION UML : COMPTE RENDU DES PREMIERS ESSAIS D'APPLICATION ET QUESTIONS

VALÉRY BÉVO, GHISLAIN LÉVESQUE ET ALAIN ABRAN
Université du Québec à Montréal
Département d'informatique

RÉSUMÉ

La mesure apparaît aujourd'hui comme un outil nécessaire pour les analyses de qualité et de productivité associées au développement et à la maintenance de logiciel. Plusieurs méthodes de mesure ont vu le jour : F.P.A. (Function Points Analysis), MarkII, F.F.P. (Full Function Points), et bien d'autres.

La méthode F.F.P. s'avère efficace pour la mesure de la taille fonctionnelle des logiciels aussi bien temps-réel qu'embarqués, de gestion ou système.

Sur un tout autre plan, UML(Unified Modelling Language) est une notation de plus en plus utilisée dans le domaine de la spécification et de la conception de logiciel. Elle fournit un vocabulaire et des règles pour représenter les différents modèles permettant de comprendre (de visualiser) un système. Elle permet de représenter un système à toutes les étapes de sa réalisation.

Quel lien faisons-nous entre la notation UML et la méthode FFP ?

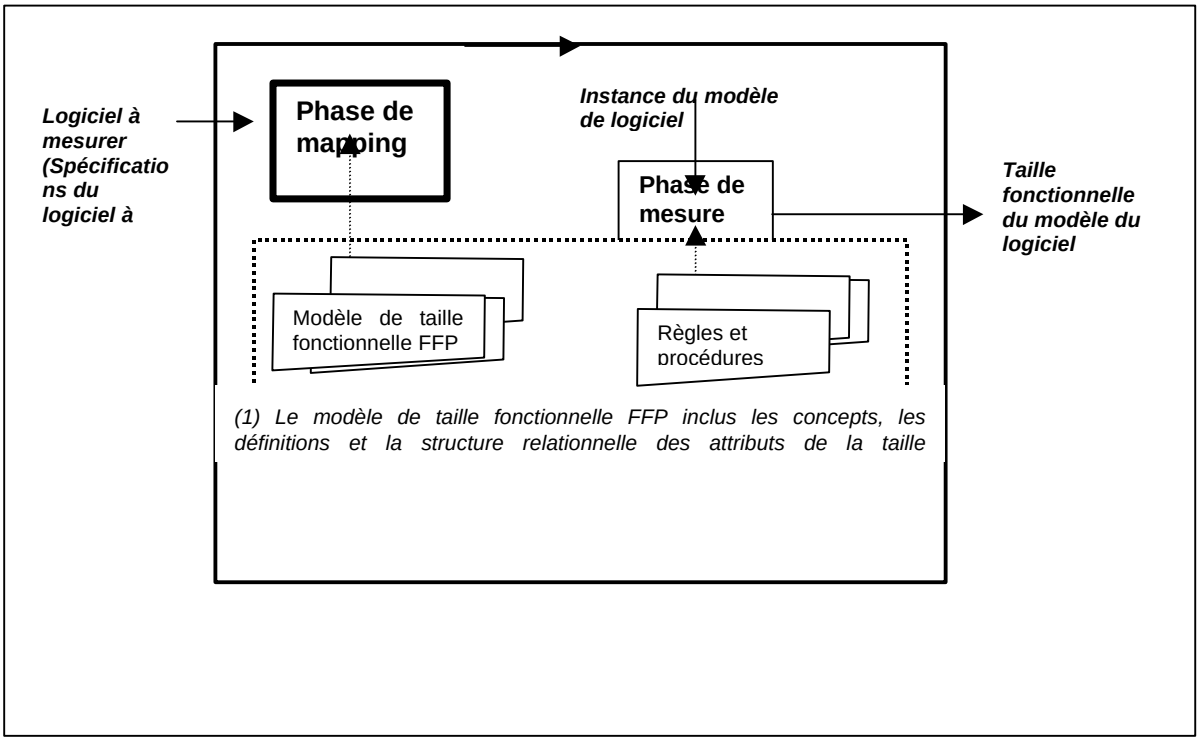
La phase de « mapping » du processus de mesure FFP apparaît comme une phase critique. En effet, la seconde phase du processus (la phase de mesure) en dépend : Plus le modèle FFP du logiciel (obtenu à l'issue de la phase de « mapping ») est complet, plus la mesure de la taille fonctionnelle du logiciel (obtenue à l'issue de la phase de mesure) est significative.

Le succès de la phase de « mapping » dépend pour beaucoup de la qualité des documents de spécification du logiciel fournis en entrée de la phase. UML s'imposant peu à peu comme une notation standard pour les méthodes d'analyse et de conception orientées-objet, de plus en plus de documents de spécifications de logiciel seront basés sur cette notation. Par conséquent, il nous a semblé opportun d'examiner la possibilité, à partir d'un document de spécification en UML d'un logiciel, de déterminer la taille fonctionnelle du logiciel à l'aide de la méthode de mesure FFP.

A défaut de pouvoir déjà proposer une solution à ce problème, nous identifions dans le présent document les questions de fond qui se posent pour sa résolution. Nous commençons par établir un rapprochement entre certains concepts FFP et UML. Sur la base de ce rapprochement, nous identifions à partir d'une étude de cas (mesure de la taille fonctionnelle d'un logiciel de contrôle d'accès à un bâtiment[1] à partir de son document de spécification en UML), les principales difficultés à surmonter pour aboutir à une solution acceptable, sans nécessairement préciser comment les surmonter. Notre proposition vise à susciter le débat autour de la question, lequel débat devrait permettre de s'orienter vers une solution acceptable.

[1] Modélisation objet avec UML, Pierre-Alain Muller, Chapitre V

1. INTRODUCTION : QUE FAIRE ?



Modèle du processus de mesure FFP

En ce début de projet, l'objectif principal est de réaliser le **« mapping »** (première phase du processus de mesure avec la méthode FFP) à partir d'une spécification en UML d'un logiciel dont on veut mesurer la taille fonctionnelle. Plus concrètement, comment identifier dans une spécification faite selon la notation UML d'un logiciel à mesurer :

1. Les différentes couches du logiciel (software layers)
2. La frontière du logiciel (software boundary)
3. Les groupes de données (data groups)
4. Les attributs de données (data attributes)
5. Les processus fonctionnels (functional processes)
6. Les sous-processus fonctionnels des différents processus fonctionnels (sub-processes)

Il s'agit donc de l'établissement d'un pont entre la méthode FFP (qui s'avère efficace pour la mesure de la taille fonctionnelle de diverses catégories de systèmes logiciels) et la modélisation en UML des systèmes.

Un premier rapprochement entre des concepts FFP-2.0 et des concepts U.M.L. est présenté dans la prochaine section. Un exemple de cas est ensuite proposé.

2. RAPPROCHEMENTS ENTRE LES CONCEPTS FFP (2.0) ET DES CONCEPTS UML

- **cas d'utilisation & processus fonctionnel (functional process)**

définition de concepts FFP-2.0 [2]

Processus fonctionnel (Functional process): Un processus fonctionnel est un ensemble unique et ordonné de mouvements de données (entry, read, write, exit) implémentant un ensemble cohésif de

[2] From "Full Function Points Measurement Manual, version 2.0, Field tests version", July 1999, section 0.

requis fonctionnels d'utilisateurs. *(un ensemble **cohésif** de processus fonctionnels réalise un tache unique et nécessite peu d'interactions avec les autres ensembles de processus fonctionnels)*

Requis fonctionnels d'utilisateurs (Functional user requirements (FUR)): Il s'agit d'une expression ISO désignant un sous ensemble des requis de l'utilisateur. Ce sont les fonctionnalités qui doivent être offertes par le système logiciel pour satisfaire les besoins de l'utilisateur.

définition de concepts UML *(UML Semantics Appendix M1-UML Glossary, version 1.0, 13 january 1997, p 16)*

Cas d'utilisation (instance et non classe): Il s'agit d'une séquence d'actions effectuées par un système, qui produit pour un acteur donné un résultat de valeur observable. *(généralement, les scénarios illustrent les cas d'utilisation)*

rapprochement

De ce qui précède, il apparaît qu'un processus fonctionnel FFP se rapproche d'un cas d'utilisation UML. En effet, qu'il s'agisse d'un cas d'utilisation UML ou d'un processus fonctionnel FFP, tous les deux représentent une suite d'actions (considérées comme mouvements de données dans FFP) effectuées par le système, et dont la finalité est la satisfaction d'un des requis fonctionnels des utilisateurs (considérés comme acteurs dans UML) du système.

- **scénarios & séquence de sous processus FFP (sub-processes (FFP))**

définition de concepts FFP-2.0[3]

Sous processus FFP (Sub-process (FFP)): Un sous processus FFP est un mouvement élémentaire de données pendant l'exécution d'un processus fonctionnel. *(Il y a quatre classes de sous processus FFP : entrée (entry), sortie (exit), lecture (read), écriture (write)).*

Entrée (Entry): Une entrée déplace les valeurs d'attributs d'un groupe de données de la partie usager de la frontière du logiciel vers l'intérieur de la frontière du logiciel. *(une entrée ne met pas à jour la donnée qu'elle déplace*.

Sortie (Exit): Une sortie déplace les valeurs d'attributs d'un groupe de données de l'intérieur de la frontière du logiciel vers la partie usager de la frontière du logiciel. *(une sortie ne lit pas la donnée qu'elle déplace*.

Lecture (Read): Une lecture fait référence aux attributs de données d'un groupe de données sans en modifier les valeurs. Sur le plan fonctionnel, une lecture prends des données se trouvant *à* l'extérieur de la frontière du logiciel, dans la partie stockage, et les met à la disposition du processus fonctionnel auquel elle appartient.

Ecriture (Write): Une écriture fait référence aux attributs de données d'un groupe de données et en modifie les valeurs. Sur le plan fonctionnel, une écriture envoie des données manipulées par le processus fonctionnel auquel elle appartient vers l'extérieur de la frontière du logiciel, dans la partie stockage.

définition de concepts UML *(UML Semantics Appendix M1-UML Glossary, version 1.0, 13 january 1997, p 12)*

Scénario : Il s'agit d'une séquence spécifique d'actions illustrant des comportements (effets observables d'une opération ou d'un événement). Un scénario peut être utilisé pour illustrer une interaction *(spécification comportementale comprenant un ensemble de messages échangés entre des objets, dans un contexte particulier pour atteindre un but spécifique) (une interaction peut être illustrée par un ou plusieurs scénarios)* .

N.B. : Nous considérons dans notre analyse qu'un scénario est **une** (car il peut y en avoir plusieurs pour le même cas d'utilisation *(cf étude de cas)*) séquence d'interactions représentatives d'un cas d'utilisation.

rapprochement

De ce qui précède, il apparaît qu'un scénario UML se rapproche d'une séquence de sous processus FFP. En effet, les actions mentionnées dans un scénario UML peuvent être exprimés en termes de mouvements élémentaires de données (entrées, sorties, lectures ou écritures FFP) et donc de sous processus FFP.

[3] From "Full Function Points Measurement Manual, version 2.0, Field tests version", July 1999, section 0.

- **déclencheur (triggering event)**

définition de concepts FFP-2.0[4]

Déclencheur (triggering event) : Un déclencheur est un événement qui se produit à l'extérieur des frontières du logiciel à mesurer et qui initie un ou plusieurs processus fonctionnel(s). *(Il y a quatre classes de sous processus FFP : entrée (entry), sortie (exit), lecture (read), écriture (write)).*

définition de concepts UML

Nous n'avons pas trouvé d'équivalent strict en UML du concept de déclencheur (un déclencheur FFP se rapproche beaucoup plus d'un cas particulier d'interaction acteur/système UML)

rapprochement

Nous faisons le rapprochement entre un déclencheur FFP et une interaction acteur/système UML pour laquelle il n'y a pas d'échange de données (celles qui apparaissent sur le diagramme de séquence) entre l'acteur et le système (dans le sens acteur vers système).

- **Classe & groupe de données (data group)**

définition de concepts FFP-2.0[4]

Groupe de données (Data group): Un groupe de données est un ensemble (non ordonné) non vide et non redondant d'attributs de données, ou chaque attribut décrit un aspect complémentaire du même objet d'intérêt. *(un objet d'intérêt est identifié du point de vue des requis fonctionnels de l'utilisateur et pourrait représenter un objet (ou une partie d'un objet) que l'on trouve dans le monde réel, ou encore un objet conceptuel (ou une partie d'un objet conceptuel), nécessaire pour supporter les opérations propres d'un logiciel)*

définition de concepts UML *(UML Semantics Appendix M1-UML Glossary, version 1.0, 13 january 1997, p 4)*

Classe : Il s'agit d'une description d'un ensemble d'objets partageant les mêmes attributs, méthodes, relations et sémantiques.

rapprochement

Si nous nous limitons uniquement aux attributs (sans considérer les méthodes, ni les relations), il apparaît qu'une classe UML devient tout simplement un ensemble (non ordonné) non vide et non redondant d'attributs de données, et donc un groupe de données FFP.

- **attributs d'un objet & attributs de données (data attributes)**

définition de concepts FFP-2.0[5]

Attribut de données (Data attribute): Un attribut de données est la plus petite parcelle d'information codée et significative du point de vue des requis fonctionnels des utilisateurs.

définition de concepts UML *(UML Semantics Appendix M1-UML Glossary, version 1.0, 13 january 1997, p 3)*

Attribut d'un objet : Il s'agit d'une propriété nommée de l'objet.

rapprochement

De ce qui précède il apparaît qu'un attribut de données FFP se rapproche bien d'un attribut d'objet UML. En effet, les attributs d'objets en UML ne sont que de l'information codée et significative du point de vue des requis fonctionnels des utilisateurs *(ils sont d'ailleurs obtenus à partir des requis fonctionnels des utilisateurs)*.

- **Diagramme des cas d'utilisation & frontières de l'application (boundaries)**

définition de concepts FFP-2.0[5]

Environnement d'opération du logiciel (Operating environment (software)): L'environnement d'opération du logiciel est l'ensemble des logiciels opérant de manière concurrente sur un système informatique spécifié.
Frontière (Boundary): La frontière d'un morceau de logiciel est la limite conceptuelle

[4] From "Full Function Points Measurement Manual, version 2.0, Field tests version", July 1999, section 0.

[5] From "Full Function Points Measurement Manual, version 2.0, Field tests version", July 1999, section 0.

séparant ce morceau de l'environnement dans lequel il opère, tel qu'elle est perçue à l'extérieur, du point de vue des utilisateurs.

définition de concepts UML *(UML Notation Guide, version 1.0, 13 january 1997, p 62)*

Diagramme des cas d'utilisation : Il s'agit d'un graphe d'acteurs, un ensemble de cas d'utilisation a l'intérieur de la frontière du système, d'associations de communication (participation) entre acteurs et cas d'utilisation, et de généralisations parmi les cas d'utilisation.

rapprochement

De ce qui précède, il apparaît que le diagramme des cas d'utilisation définit de manière intrinsèque la frontière du système.

- **Acteur UML & utilisateur (ou usager) FFP**

définition de concepts FFP-2.0[6]

Utilisateur FFP (FFP Users): Il s'agit d'un être humain, d'un autre logiciel ou d'un dispositif, qui interagit avec le logiciel mesuré.

définition de concepts UML *(UML Semantics Appendix M1-UML Glossary, version 1.0, 13 january 1997, p 2)*

Acteur (UML) : Il s'agit d'un stéréotype prédéfini de type décrivant une entité hors du système, qui interagit avec les cas d'utilisation *(et donc avec le système)*

rapprochement

De ce qui précède, il apparaît que la notion d'utilisateur FFP est très proche de la notion d'acteur UML.

- **couches de l'application (layers)**

définition de concepts FFP-2.0[6]

Couche (Layer): Une couche est une partition fonctionnelle de l'environnement du logiciel dans laquelle tous les processus fonctionnels inclus exhibent un degré élevé de cohésion .

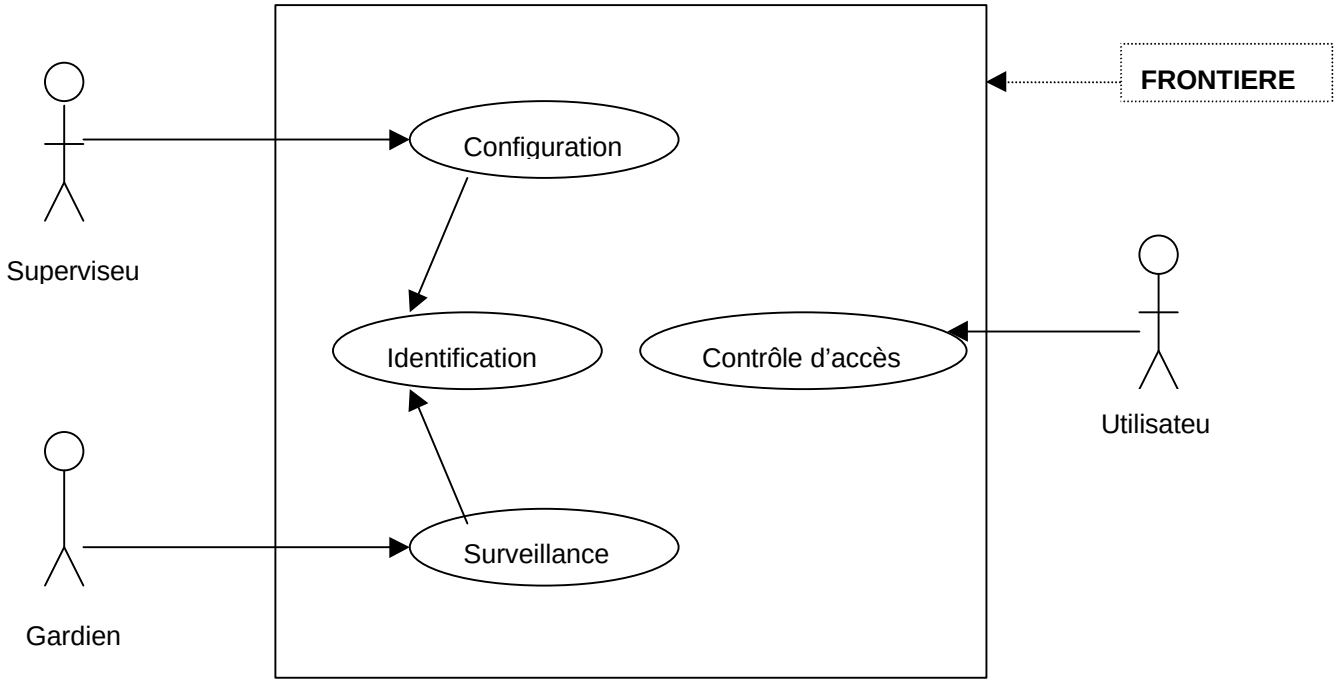
rapprochement

Le seul rapprochement que nous pouvons faire c'est de considérer une couche comme un système a part entière. Il est bien entendu qu'un système est susceptible d'interagir avec d'autres systèmes, tout comme une couche peut bel et bien interagir avec un autre couche (tel que mentionné dans les spécifications FFP).

[6] 3 From "Full Function Points Measurement Manual, version 2.0, Field tests version", July 1999, section 0.

3. ÉTUDE DE CAS : APPLICATION DE CONTRÔLE D'ACCÈS A UN BÂTIMENT[7]

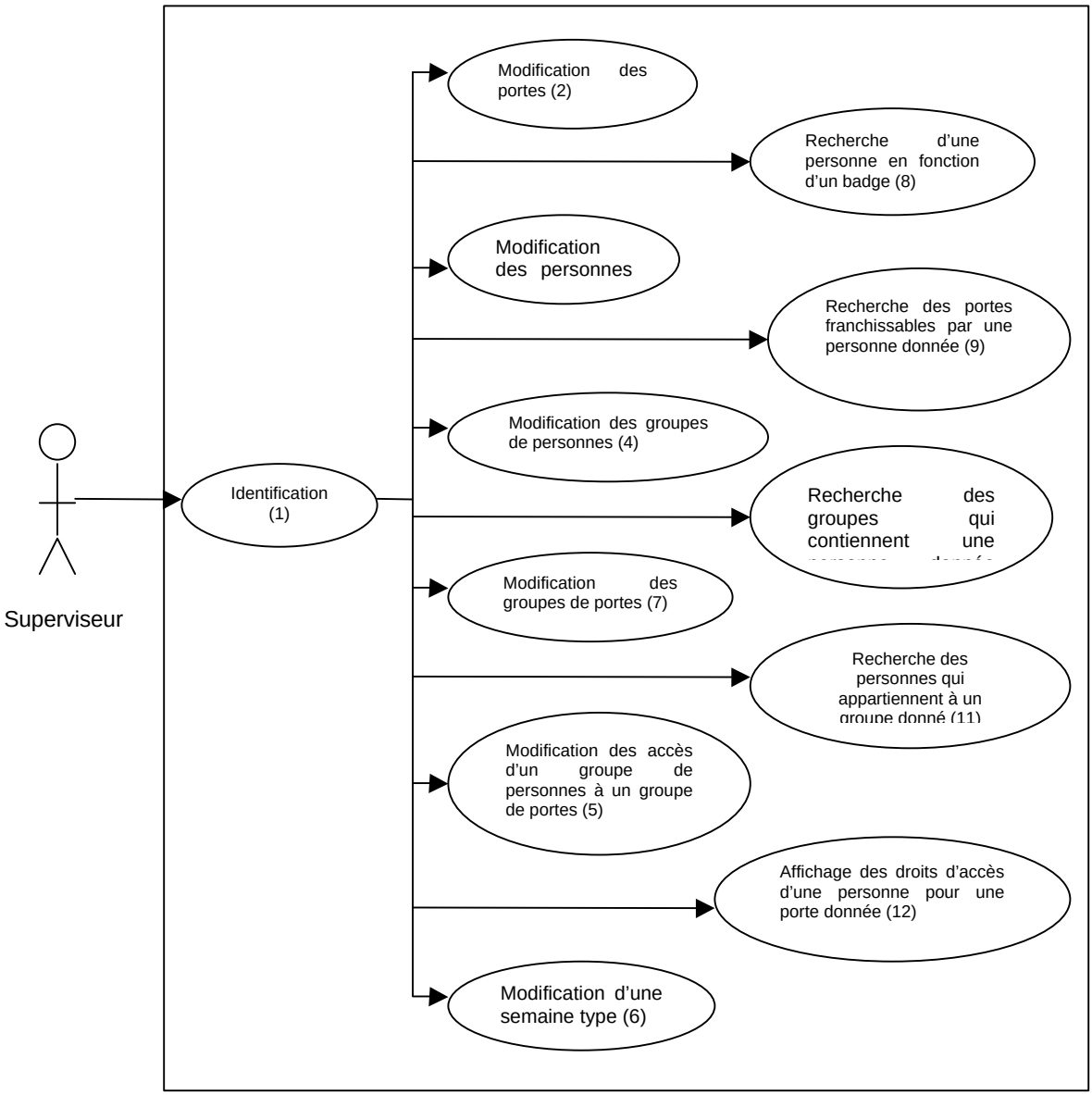
- **Frontière**



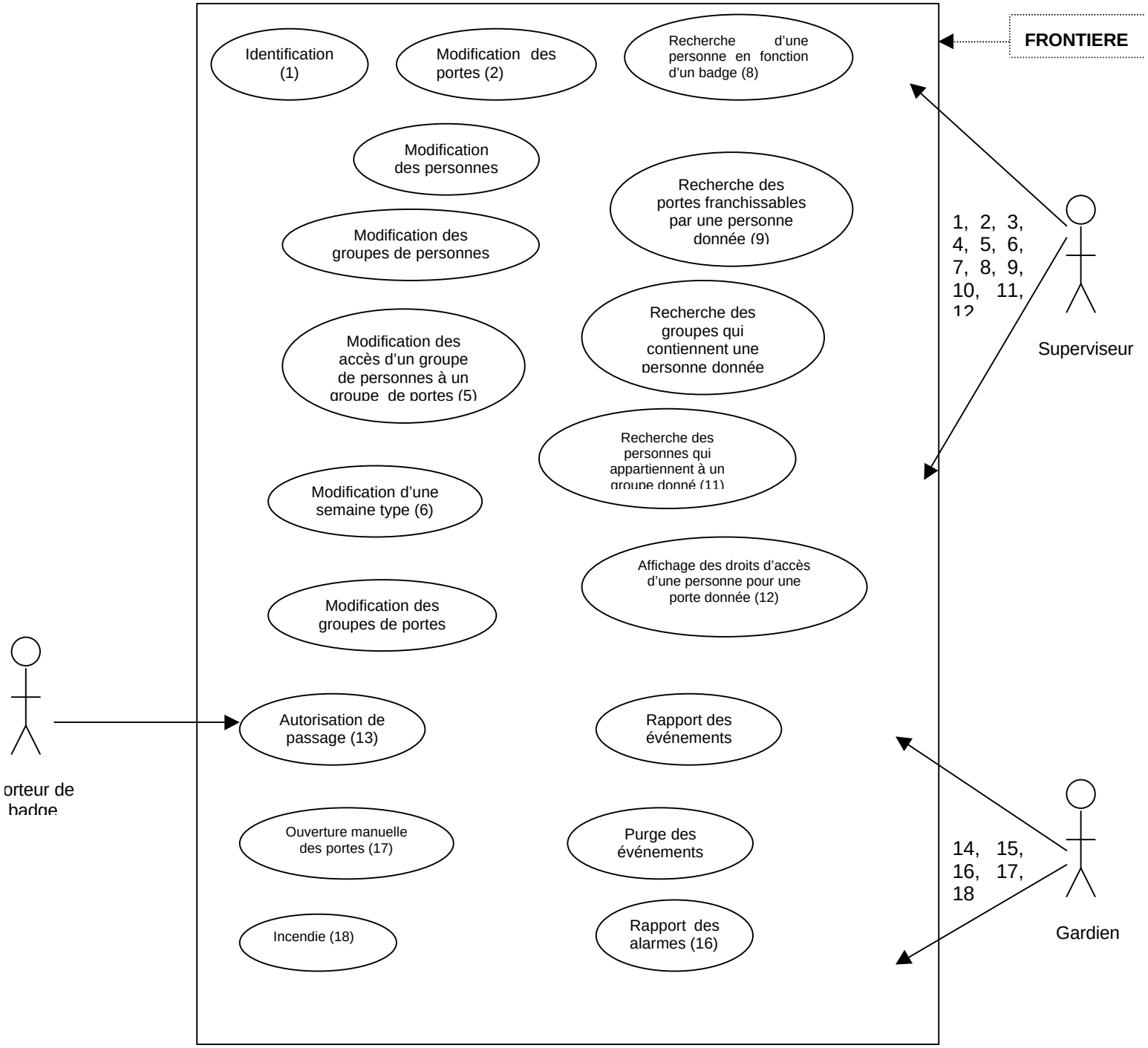
Question : Dans la mesure où un cas d'utilisation est susceptible de comporter plusieurs scénarios et que les scénarios sont relativement indépendants les uns par rapport aux autres ⇒ Ne serait-il pas plus judicieux de considérer chacun des scénarios comme un processus fonctionnel ? *(auquel cas, il faudrait faire un rapprochement entre processus fonctionnel FFP et scénario UML)*

[7] Modélisation objet avec UML, Pierre-Alain Muller, Chapitre V

Examinons un peu plus en détail le cas d'utilisation **Configuration**. Il comporte plusieurs scénarios, lesquels scénarios sont relativement indépendant les uns par rapport aux autres.



En faisant donc le rapprochement entre scénario UML et processus fonctionnel FFP nous obtenons le diagramme suivant :



- **Couches**

Le système ne comporte qu'une seule couche.

- **Groupes de données**

Ils sont obtenus à partir du diagramme des classes du domaine (lui-même obtenu à partir des diagrammes de collaboration).

- ➔ Lecteur de badges
- ➔ Badge
- ➔ Porte
- ➔ Personne
- ➔ Groupe de portes
- ➔ Groupe de personnes
- ➔ Semaine type
- ➔ Calendrier

- **Attributs de données**

- ➔ Lecteur de badges (adresse, anti-retour, site, temporisation, type d'événements, veille)
- ➔ Badge (validité, numéro de site, numéro de badge)
- ➔ Porte (numéro de porte, numéro de salle)
- ➔ Personne (prénom, nom)
- ➔ Groupe de portes (nom)
- ➔ Groupe de personnes (nom)
- ➔ Semaine type ()

- **Processus fonctionnels**

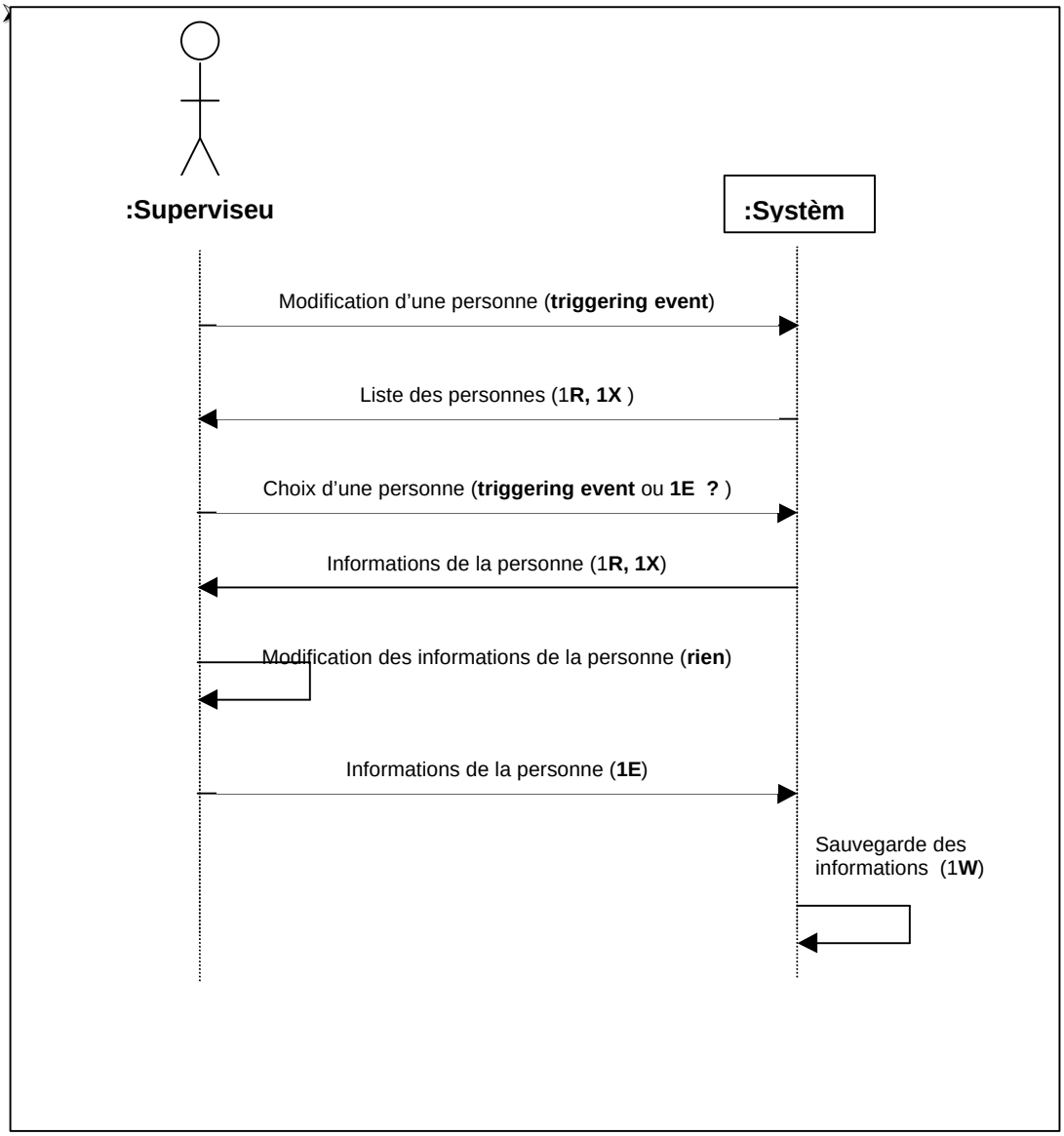
(en supposant qu'ils correspondent aux différents cas d'utilisation recensés).
Configuration
Identification
Contrôle d'accès
Surveillance

- **Utilisateurs**

Ils correspondent aux différents acteurs recensés.
Superviseur
Gardien
Utilisateur (porteur de badge)

- **Exemple de Scénario : Modification d'une personne**

IDENTIFICATION DES SOUS PROCESSUS FONCTIONNELS DANS UN SCÉNARIO



- **Modèle FFP du système (en faisant le rapprochement cas d'utilisation/processus fonctionnel)**

LAYERS	FUNCTIONAL PROCESSES	DATA GROUPS								SUB PROCESSES			
LAYER 1		***Badge***	***Porte***	***Personne***	***Groupe de portes***	***Groupe de personnes***	***Lecteur de badge***	***Semaine type***	***Calendrier***	**ENTRY (E)**	**EXIT (X)**	**READ (R)**	**WRITE (W)**
	CONFIGURATION	E	R, X, E, W	R, X, E, W	R, X, E, W	R, X, E, W		R, X, E, W		6	5	4	5
	Identification			R, X, E						1	1	1	
	Contrôle d'accès	R, E, & (X)[8]								1	1	1	
	Surveillance		R, X					R, X, E, W		1	2	2	1
TOTAL – layer 1										9	9	8	6
GRAND TOTAL										**32 FFPs**			

Question :

1- Peut-on appliquer ici sans heurt la règle de non duplication de sous processus (pour une même entrée [ENTRY]) à l'intérieur d'un même processus fonctionnel ? *(Nous avons le sentiment que non, dans la mesure où le processus fonctionnel apparaît ici comme de très haut niveau par rapport à ses sous-processus)*

[8] pour le message d'autorisation

- **Modèle FFP du système**

(en faisant le rapprochement scénario UML/processus fonctionnel FFP)

LAYERS	FUNCTIONAL PROCESSES	DATA GROUPS								SUB PROCESSES			
LAYER 1		***Badge***	***Porte***	***Personne***	***Groupe de portes***	***Groupe de personnes***	***Lecteur de badge***	***Semaine type***	***Calendrier***	**ENTRY (E)**	**EXIT (X)**	**READ (R)**	**WRITE (W)**
	(1)			E, R, X						1	1	1	
	(2)		R, X, E,W							1	1	1	1
	(3)			R, X, E, W						1	1	1	1
	(4)					R, X, E, W				1	1	1	1
	(5)				R, X	R, X		R, X, E, W		1	3	3	1
	(6)							R, X, E, W		1	1	1	1
	(7)				R, X, E, W					1	1	1	1
	(8)	E		R, X						1	1	1	
	(9)		R, X	R, X							2	2	
	(10)			R, X		R, X					2	2	
	(11)			R, X		R, X					2	2	
	(12)		R, X	R, X				R, X			3	3	
	(13)	E, R	E, X							2	1	1	
	(14)							E, R, X		1	1	1	
	(15)							E		1			
	(16)							E, R, X		1	1	1	
	(17)		R, X					W			1	1	1
	(18)		R, X								1	1	
TOTAL – layer 1										13	24	24	7
GRAND TOTAL										**68 FFPs**			

4. SYNTHÈSE DES QUESTIONS SOULEVÉES

A quel concept UML doit-on associer le concept de processus fonctionnel FFP ? (à ce niveau il se pose un problème de granularité de la mesure. Doit-on considérer comme processus fonctionnel uniquement les grandes fonctionnalités du système [cas d'utilisation], ou alors le détail de ces fonctionnalités [scénarios] ?)
Les cas d'utilisation et les scénarios associés sont-ils suffisants pour recenser tous les mouvements de données (sous processus FFP) dans un système dont on veut déterminer la taille fonctionnelle ? *(ne doit-on pas faire appel également aux diagrammes de collaboration ou encore aux diagrammes d'états/transitions ?)*
Le diagramme des classes du domaine est-il suffisant pour identifier tous les groupes de données du système ?
Quel est l'impact du niveau de détail du formalisme utilisé dans les spécifications sur les résultats ?
Comment changer d'échelle ? (Règles de passage si l'on veut mesurer à un niveau macro, à partir d'un niveau micro par exemple)

5. PROCHAINES ÉTAPES

Validation analytique (stabilisation des arrimages entre concepts)

Construction d'études de cas (certaines disposant déjà d'une spécification en UML et d'autres pas encore)

Fit-back par rapport à la méthode FFP (éventuelles évolutions)

Tests à une plus large échelle (projets d'entreprise)

REMERCIEMENTS

Nous remercions Bell Canada ainsi que le Conseil National de Recherches en Sciences Naturelles et en Génie du Canada, pour leur soutien financier. Les opinions exprimées dans ce document sont uniquement celles de leurs auteurs.

RÉFÉRENCES

[1] L.R.G.L. & S.E.L.A.M., *Full Function Points Measurement Manual*, version 2.0, Field tests version, July 1999

[2] Ghislain Lévesque, Analyse de système orientée-objet et génie logiciel, Chenelière/McGraw-Hill, 1998

[3] Rational Software Corporation, *UML Semantics Appendix M1-UML Glossary*, version 1.0 , 13 January 1997

[4] Rational Software Corporation, *UML Notation Guide*, version 1.0, 13 January 1997

[5] Pierre-Alain Muller, Modélisation objet avec UML, 1997

[6] [ESCOM'96] Frank Armour, Bill Catherwood and Sean Furey*, Experiences measuring object oriented system size with use cases*, American Management Systems, 1996