

À propos de dossier

Alain VAILLY
Département Informatique - UFR de Sciences et Techniques
Université de Nantes

I) Introduction

Après avoir corrigé pendant de nombreuses années (plus de trente !) des dossiers remis par des étudiants, il est clair que la fourniture d'un plan est un acte essentiel. Il leur fixe un cadre de travail dans lequel ils vont œuvrer. Le plan détermine la liste des questions posées auxquelles ils vont devoir répondre.

Par delà l'exercice universitaire, ces travaux visent à former à des pratiques. Le plan doit donc refléter ce qui est attendu à « l'extérieur », dans les entreprises. Il est donc important qu'il ne soit pas totalement déconnecté des usages professionnels.

Nous fournissons, dans ce document, et avec beaucoup de précautions, UN plan type, tout en sachant qu'il y en a bien d'autres, tous aussi valables. Nous complétons notre message en donnant un exemple de « dossier » qui respecte l'enchaînement suggéré par ce plan. Cet exemple a été emprunté à un cas réel. Nous ne fournissons que des éléments de solution, le dossier complet étant trop volumineux.

Pour compenser ceci, nous avons mis à disposition, sur le même site, trois exemples complets de dossiers, un bon, un moyen et un mauvais. De la lecture des trois surgira, nous en sommes convaincu, la « lumière ».

II) Plan-type d'un dossier de spécification « à la UML »

Un dossier de spécification, en UML ou dans un autre langage, doit avant tout être rédigé en français, de façon à être lu par des informaticiens et par des non-informaticiens (clients, utilisateurs...). C'est la raison pour laquelle nous tentons de rejeter dans les annexes tous les « grands » schémas que l'on trouve habituellement dans ces dossiers : diagrammes de classes, dictionnaires...

Un dossier de spécification, en UML, va refléter l'approche par étapes du processus UP. La première étape correspond à l'analyse des besoins, avec les besoins fonctionnels (ce que veut l'utilisateur en matière de fonctionnalités offertes) et les besoins non fonctionnels (exigences de qualité, critères de performance...). Elle se termine le plus souvent par la production d'un premier diagramme de classes métier simplifié. En termes de diagrammes UML, on trouvera dans cette partie des diagrammes de cas d'utilisation –avec leurs descriptions textuelles– et des scénarios.

La deuxième étape est celle de l'analyse de la solution. C'est durant cette étape que s'élabore la solution. Chaque fonctionnalité demandée va être minutieusement décrite, confrontée avec la structure de données, celle-ci s'enrichissant au fur et à mesure que l'étude des fonctions progresse. On retrouvera dans cette partie les diagrammes de séquences, les diagrammes de collaboration, un diagramme de classes plus précis et, si le besoin s'en fait sentir, des diagrammes états-transitions.

Nous proposons donc un plan en deux parties, encadrées globalement par une introduction et une conclusion :

- 1. Introduction générale
- 2. Analyse des besoins
 - Introduction
 - Besoins fonctionnels
 - Besoins non fonctionnels
 - Ebauche de modèle métier
 - Conclusion
- 3. Analyse de la solution
 - Introduction
 - Objectifs assignés
 - Fonctionnalités mis en œuvre
 - Structure de données proposée
 - Conclusion
- 5. Conclusion générale

Ce plan est une réponse posée à la question fréquemment posée par les étudiants (« Qu'est-ce qu'il faut mettre dans le dossier ? »). Il y en a d'autres, toutes aussi valables. Nous prenons, simplement, appui sur plus de trente années d'expérience dans ce domaine et sur la rédaction et la lecture de plusieurs centaines de dossiers pour proposer celui-ci. C'est un bon plan. Ce n'est pas le seul.

III) Exemple

L'exemple que nous avons choisi de prendre est extrait d'un ouvrage destiné à la formation des responsables de services informatiques. Il est à vocation pédagogique puisqu'il vient illustrer un cours sur le langage UML et le processus unifié.

Nous avons toutefois rédigé ce texte à partir d'un cas réel, celui d'une gestion d'une association professionnelle... de responsables de services informatiques. Cette association existe (nous avons toutefois changé son nom) ; ses besoins aussi.

Nous avons donc pris le texte d'origine, en réduisant certaines parties, en anonymant certaines autres, de façon à respecter la confidentialité et aussi à obtenir un document d'une taille adéquate. Trop petit, l'illustration ne serait qu'incomplète. Trop long, la lassitude risquerait de s'installer.

III.1) Exemple de partie consacrée à l'analyse des besoins

L'analyse des besoins sert à bien comprendre ce qui se passe dans « l'espace du problème ». Il s'agit de décrire précisément, et dans un langage compréhensible par lui, ce que fait et veut l'utilisateur.

Dans le cas qui nous « préoccupe », il y a trois cas d'utilisation principaux, qui vont se diviser en dix-sept sous-cas, eux-même décrits par autant de scénarios. Il est donc bien clair que nous ne décrirons pas tout. Cette partie se termine par la présentation d'une première version du diagramme de classes métier, sans aucune opération.

L'analyse des besoins met en évidence les exigences fonctionnelles et non-fonctionnelles du système étudié.

Une première synthèse des informations collectées auprès des utilisateurs d'ARGOSI et des documents qui nous ont été remis permet de dégager trois grands secteurs d'activité : administration, séminaires et pôles. Le premier traite de tout ce qui se rapporte aux assemblées

générales, à la gestion des membres et de leurs situations. Le deuxième comprend les réunions au niveau du Grand Ouest (i.e. les séminaires), celles qui sont régionales (i.e. les pôles) étant regroupées dans le troisième. Appliquant le principe « diviser pour régner », nous découpons le système en trois paquetages que nous pouvons étudier séparément.

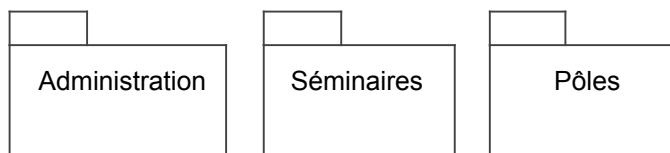


Fig. 01 – Architecture générale des paquetages

Le découpage en paquetage correspond à une structuration verticale de l'application. Ainsi, chaque modèle produit (besoins, analyse, conception, etc) peut être structuré selon ce découpage.

Dans ce qui suit, on se restreint à la partie **Administration**, car il s'agit du secteur principal de l'association. Cela ne change pas l'organisation du travail car les deux modes d'organisation restent envisageables pour la suite.

L'administration de l'association consiste à gérer ses membres (et notamment, mais pas seulement, son conseil d'administration et son bureau) et leurs situations. Essayons d'y voir un peu plus clair en modélisant son fonctionnement grâce à des cas d'utilisation. Deux parties se distinguent nettement, celle qui permet de gérer les membres et celle qui permet de gérer les situations. Une première description en cas d'utilisation peut donc être obtenue :

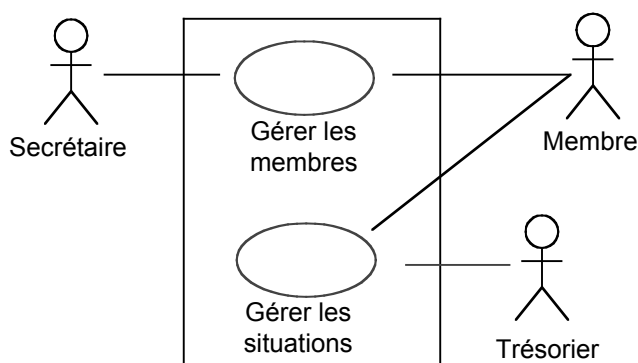


Fig. 02 – Cas d'utilisation général

I- GESTION DES MEMBRES

La gestion des membres comprend une partie Gestion des réunions du conseil, une partie Gestion des assemblées générales et une partie Gestion des membres eux-mêmes.

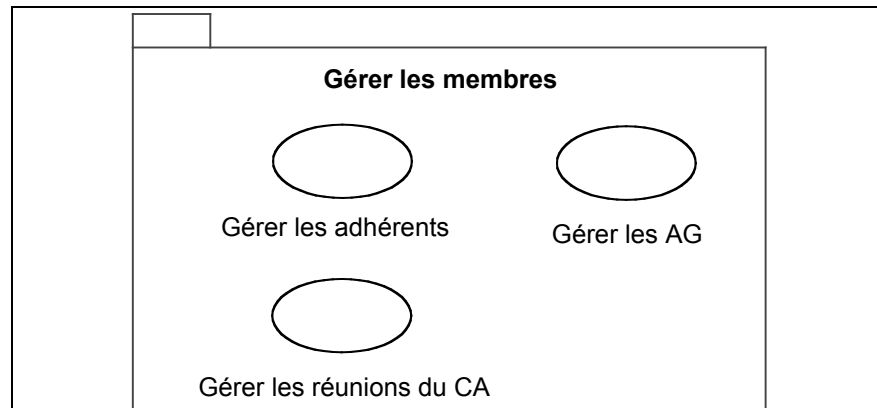


Fig. 03 - Paquetage Gérer les membres

Détaillons maintenant chacun des cas d'utilisation.

1.1- Gérer les adhérents

La définition des besoins comprend une description du cas d'utilisation et des scénarios pour l'illustrer.

1.1.1- Cas d'utilisation

La gestion des adhérents est la partie la plus classique (enregistrer un nouvel adhérent, prendre en compte un changement d'adresse, radier un adhérent... en constituent les composants) et la moins sujette à interactions importantes entre le système et son environnement. A titre d'illustration, et sans que cela épuise le sujet, nous pouvons décrire brièvement le cas comme suit :

Cas d'utilisation : Gérer les adhérents

Acteurs primaires : Secrétaire, Membre, Membre actif

Description

La gestion des adhérents comprend l'ajout d'un membre, la définition du bureau de l'association, la démission d'un membre, son départ, l'édition de listes diverses.

Cas : Ajout d'un membre

Pré-condition : Le membre à ajouter n'existe pas

Saisir les informations sur le membre (intitulé postal de l'entreprise, nom de l'entreprise en clair, adresse de l'entreprise, nom du contact, numéros de téléphone et de fax, adresse électronique, date de naissance). Vérifier qu'il n'existe pas et, si tel est le cas, enregistrer les informations sur ce membre.

Cas : Définir le bureau

Pré-condition : Les membres du bureau sont des membres actifs du CA

A la suite de l'AG, la composition du bureau est fournie au "système". Pour chaque fonction, la personne choisie est mentionnée. Elle doit obligatoirement être membre du CA. Le choix est effectué pendant une AG. La date d'entrée en fonction est la date de l'assemblée générale durant laquelle le choix du membre a eu lieu.

Cas : Démettre un membre du bureau

La démission d'un membre du bureau passe d'abord par l'enregistrement d'une demande de démission émanant d'un membre actif de l'association. Une fois cette demande mémorisée, le nombre de demandes similaires est comptabilisé. S'il y a plus de la moitié de ce bureau qui réclame la démission de cette personne, le secrétariat est prévenu : le conseil doit se réunir en urgence. Il (i.e. le secrétariat) choisit une date et le système programme une réunion (état *prévue*, nature *provoquée*, ordre du jour *démission membre*). Les personnes concernées sont les membres du CA. Le délai de convocation est de 8 jours.

Appel UC Programmer une réunion

Cas : Editer les listes

L'édition de listes correspond à la production d'une liste (!) de personnes appartenant à un groupe donné : liste des membres de l'association, liste des membres du bureau, liste des membres du CA, liste des membres sortants...

Cas : Départ d'un membre

Il y a plusieurs cas de départ : un membre peut quitter l'association ou bien seulement quitter la fonction qu'il occupait. Un membre qui quitte la fonction qu'il exerce se signale au secrétariat. Celui-ci fournit l'information au système qui enregistre le départ. La date de signalement est notée comme la date de sortie de la fonction du membre. Le membre qui part "perd" toutes les demandes de démission déposées contre lui. Celles-ci sont détruites. Un membre qui quitte l'association est supprimé (il s'agit d'un marquage, son état prenant la valeur *parti*), les fonctions qu'il occupait deviennent vacantes, les demandes de démission qu'il avait éventuellement sont supprimées (elles, réellement).

Exceptions

Cas : Proposer un inconnu

La proposition de membres du bureau concerne quelqu'un qui n'est pas membre de l'association, ni du conseil d'administration. La proposition est refusée.

Cas : Ajouter un membre connu

Le membre qui doit être ajouté existe déjà. L'ajout est refusé.

Dans cette description, nous avons pris en compte certaines hypothèses, qu'il est bon de rappeler ici :

[fonctions concernées par une demande de démission] Une demande concerne un membre du bureau. Si cette personne est également responsable, par exemple, d'un pôle ou d'un séminaire, cette deuxième responsabilité n'est pas en jeu.

[portée d'une demande de démission] Les demandes sont conservées jusqu'au départ du bureau de la personne. A ce moment-là, elles sont supprimées.

I.1.2- Scénarios

La description textuelle recense cinq cas normaux et deux cas exceptionnels. Les interactions entre le système et les utilisateurs doivent donc être décrites sous la forme de sept scénarios.

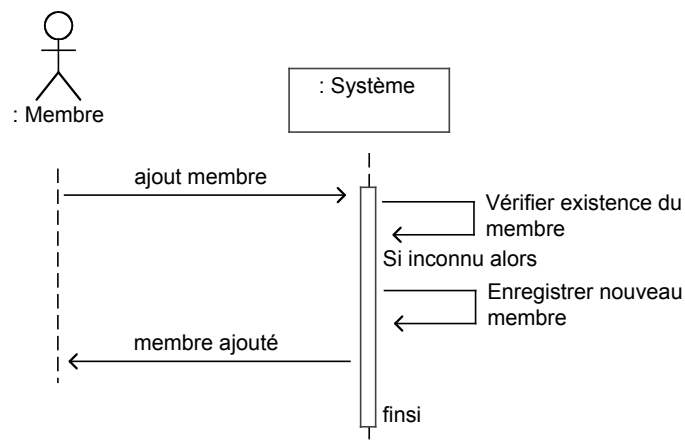


Fig. 04 - Scénario Ajout d'un membre

Pour faire partie de l'association, le membre fait une demande d'ajout. Après vérification, le membre inexistant est enregistré.

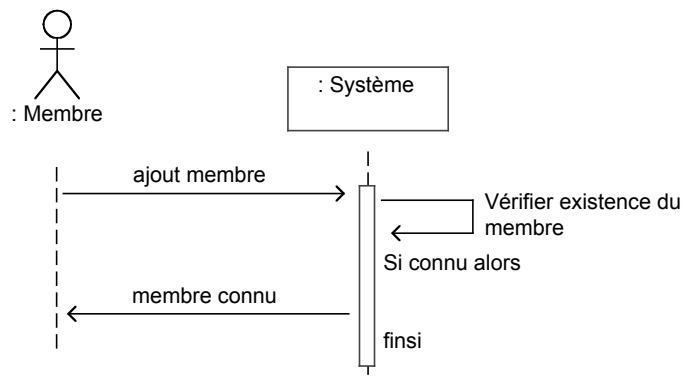


Fig. 05 - Scénario Ajouter un membre connu

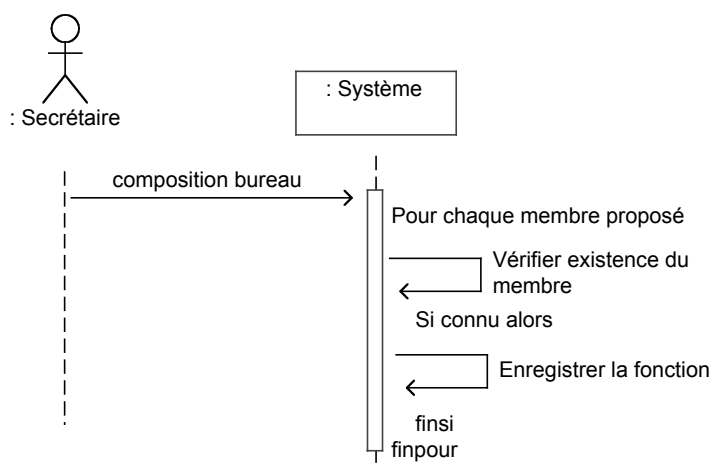


Fig. 06 - Scénario Définir le bureau

xxxxxx description de tous les scénarios composant le cas xxxxxx

Nous pourrions ajouter des cas permettant à un membre de se porter candidat à une élection, de retirer sa candidature... Nous ne

développerons pas davantage. Par contre, nous allons détailler les deux autres UC, gestion des assemblées générales et gestion des réunions du conseil d'administration.

I.2- Gérer les AG

xxxxxx introduction à la description du cas Gérer les AG xxxxxxx

I.2.1- Cas d'utilisation

La gestion des assemblées générales comprend la convocation à l'assemblée générale ordinaire -AG- (le cas "standard") ou extraordinaire -AGE- (le cas "anormal") et la gestion des élections. Celles-ci se déroulent en deux temps, vote puis dépouillement. Celui-ci se fait en deux étapes, au cours desquelles sont comptabilisés les votants puis les voix. Nous devons faire, ici, quelques hypothèses supplémentaires :

[informatisation des élections] Les élections proprement dites seront informatisées, ce qui signifie que les votes seront saisis, les voix attribuées aux membres enregistrés.

[organisation de l'AG] Une AG (ou une AGE) est préparée un mois avant la date fixée pour sa tenue. Elle est programmée en début d'année.

[remplacement des membres lors d'une AGE] Lors d'une AGE, le remplacement se fait membre par membre : un membre élu prend la place (dans le tiers d'affectation) du membre remplacé.

[procuration] Il n'y a pas de vote par procuration. Les votes se font à bulletins secrets.

[membres électeurs] Seuls les membres à jour de leur cotisation peuvent voter.

[égalité] En cas d'égalité de voix, les candidats sont classés par ordre d'âge croissant.

La structuration de cette gestion en cas comprend une partie "évidente", avec les deux cas *Programmer une AG* et *Convoquer une AG*, et une partie, plus "floue", correspondant au vote. Nous pouvons structurer ce vote en trois cas normaux :

- *Voter*, de pré-condition "bureau ouvert" ;
- *Comptabiliser les votants*, de pré-condition "bureau ouvert" et de post-condition "bureau fermé" ;
- *Comptabiliser les voix*, de pré-condition "bureau fermé" ;

ou bien en un seul (*Voter*), qui englobe les deux autres. Nous choisissons la seconde solution, plus cohérente et indépendante.

Nous compléterons la description par deux cas exceptionnels (*Convoquer une AGE* et *Re-convoquer une assemblée*).

Cas d'utilisation : Gérer les AG

Acteurs primaires : Secrétaire

Acteurs secondaires : Membre

Invariant : L'ensemble des membres de l'association ne change pas.

Description

La gestion des assemblées générales comprend la programmation de ces assemblées, la convocation à l'assemblée générale (le cas "standard") ou extraordinaire (le cas "anormal") et la gestion des élections. Cette dernière comprend la gestion des votes, la comptabilisation des votants et des voix, regroupées en un seul cas, *Voter*.

Cas : Programmer une AG

Tous les ans, en début d'année, la date de l'assemblée générale est choisie. La programmation de cette AG consiste à créer une réunion ayant pour nature la valeur *AG*, pour code la valeur *prévue*. Son ordre du jour peut être, au moment de la programmation, encore indéfini. Il est fourni par le secrétariat. Tous les membres de l'association sont concernés par cette réunion. Le délai de convocation est d'un mois.

Cas : Convoquer une AG

Tous les ans, un mois avant la date choisie, une convocation est envoyée à tous les membres de l'association. Cette convocation comprend l'ordre du jour de l'AG (défini par le président et fourni par le secrétariat), la date de l'AG (fournie par le secrétariat en début d'année) et la liste des membres actifs à remplacer. Cette liste recense les membres élus trois années auparavant (le "tiers sortant"). Il s'agit d'une AG ordinaire.

Cas : Voter

Les votes ne peuvent avoir lieu que si le bureau de vote est ouvert. Cette ouverture faite, quand un membre de l'association se présente pour voter, il y a vérification de sa situation. Seuls, en effet, les membres à jour de leur cotisation reçoivent les bulletins de vote. Si ce membre n'est pas en règle, il peut payer immédiatement sa cotisation. Il recevra ensuite les bulletins lui permettant de s'exprimer. A la clôture du bureau de vote, le nombre de votants est calculé. S'il y a eu au moins deux tiers de membres à jour de leur cotisation qui ont voté, le vote est déclaré licite. Le nombre de voix attribuées à chaque candidat est comptabilisé. Ceux qui ont recueilli le plus grand nombre de suffrages (majorité relative) sont déclarés élus membres actifs de l'association dans la limite du nombre de sièges à pourvoir. En cas d'égalité de voix, les candidats sont classés par ordre d'âge croissant.

Exceptions

Cas : Convoquer une AGE

Pré-condition : Plus du quart des membres actifs à remplacer

Si plus d'un quart des membres actifs fait défaut (par suite de décès, démission ou radiation dans l'année écoulée), alors le président devra convoquer une assemblée générale extraordinaire (AGE) afin de pourvoir à leur remplacement. Le traitement est similaire à celui d'une AG, mais les membres à remplacer sont ceux qui font défaut et non le tiers sortant. Lors d'une AGE, le remplacement se fait membre par membre : un membre élu prend la place (dans le tiers d'affection) du membre remplacé.

Cas : Re-convoquer une assemblée

Pré-condition : Moins de 2/3 des membres à jour ont voté

Le vote est annulé. Une nouvelle AG est programmée, à une nouvelle date, communiquée par le secrétariat. L'ordre du jour, la liste des membres actifs à remplacer sont inchangés. Dans la "foulée", la convocation est envoyée.

Nous avons, dans cette description textuelle, traité différemment les cas "anormaux" : exception dans certaines situations, alternative dans d'autres. C'est volontaire et ne relève pas seulement de la simple volonté, pédagogique, de mettre en évidence toutes les possibilités offertes par UML à l'analyste.

Nous avons également pris en compte, dans notre choix, la "rareté" d'apparition de chaque situation. Le cas où plus du quart des membres actifs fait défaut, celui dans lequel moins des deux tiers des membres s'expriment sont moins fréquents que celui d'un membre non à jour de sa cotisation ou bien encore celui d'une égalité entre deux candidats élus. Ceci est, bien évidemment, discutable.

I.2.2- Scénarios

xxxxxx suite de la présentation des différents cas xxxxxx

II- MODÈLE DU DOMAINE

De ces modèles et des informations contenues dans l'énoncé, nous pouvons extraire un modèle du domaine. La figure 15 est une première version du diagramme de classes.

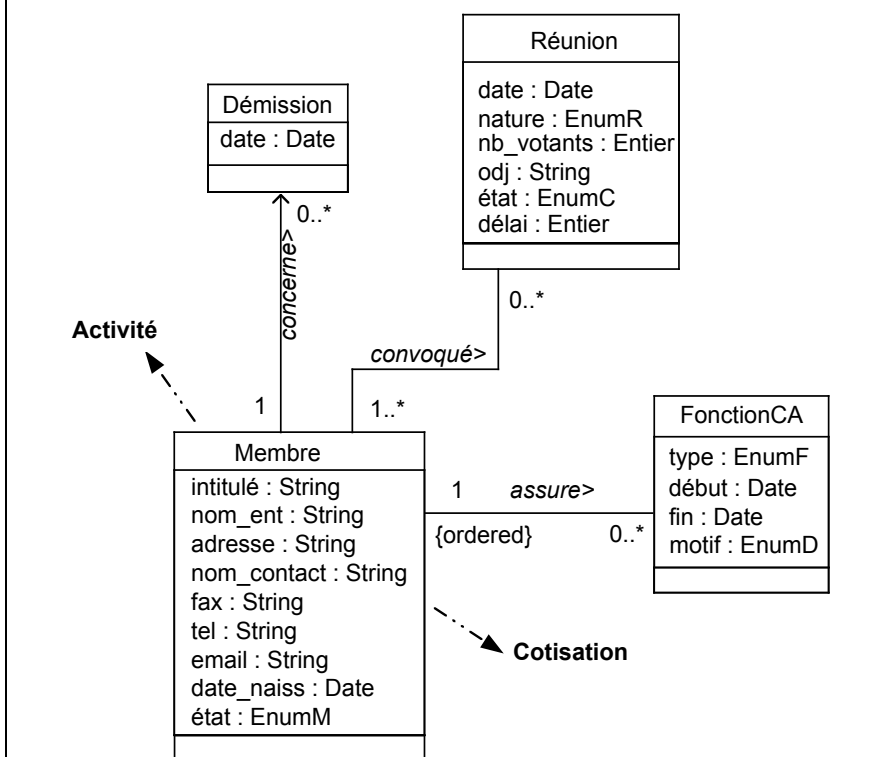


Fig. 15 - Diagramme de classes métier V1

Un membre est décrit par l'intitulé de l'entreprise, le nom en clair, l'adresse de celle-ci, le nom du contact, ses numéros de téléphone et de fax, son adresse électronique et sa date de naissance. Il lui est attribué un statut (voir les diagrammes états-transitions pour connaître les valeurs prises par cet attribut). Un membre peut avoir plusieurs fonctions, mais il ne peut cumuler celles qu'il occupe au sein du bureau. Ceci milite en faveur d'une énumération de ces fonctions (celle-ci pourra sans doute être complétée lors de l'étude des autres paquetages et notamment des responsabilités de pôles et/ou de séminaires) et d'une mémorisation des dates d'entrée et de sortie d'une fonction. Nous avons également enregistré le motif de départ d'une fonction, une fonction actuellement occupée par un membre ayant la valeur *encours* pour l'attribut *motif* et une date de sortie spéciale. Nous avons arbitrairement posé une date infinie pour simplifier les opérations de comparaison entre dates et ne pas distinguer la fonction en cours des autres fonctions.

De façon à accélérer un peu les traitements, les membres ayant des fonctions (ou ayant eu des fonctions) sont classés (en ordre croissant ?) par rapport à leur date d'entrée en fonction.

Nous pouvons faire plusieurs remarques concernant les fonctions :

- Les fonctions assurées par les membres peuvent être modélisées sous forme d'une classe *Type_fonction*, ayant autant d'instances qu'il y a de valeurs dans *EnumF*, reliée à la classe *Membre* par des associations. Celles-ci expriment l'implication des membres. Un membre relié à un type de fonction assure actuellement la fonction (ou a assuré un jour la fonction, si nous mettons en place un historique). Ces fonctions peuvent aussi être modélisées sous la forme d'une classe *Fonction*, correspondant à une fonction assurée actuellement par un membre (ou ayant été assurée un jour par un membre, si nous mettons en place un historique). Nous avons choisi la seconde façon de faire.
- Pour éviter toute ambiguïté, et dans la mesure où nous ne traitons que la partie *Administration*, nous avons renommé la classe *Fonction* en *FonctionCA*. D'autres fonctions pourront émerger lors de l'intégration des divers paquetages.
- Nous ne mettons pas en place un historique des fonctions assurées par les membres, bien qu'il soit rendu possible par la structure des données mise en place. En d'autres termes, nous pouvons le faire, mais ne le faisons pas pour l'instant.

Les règles de non-cumul sont multiples :

- une personne ne peut occuper deux fonctions simultanément ;
- elle ne peut pas non plus occuper la même fonction pendant deux intervalles de temps (l'intervalle étant défini comme le temps qui s'écoule entre la date d'entrée en fonction et la date de sortie de celle-ci) se chevauchant ;
- deux personnes, enfin, ne peuvent occuper la même fonction simultanément, sauf s'il s'agit de vice-présidents (auquel cas il ne peut y en avoir que deux).

Un certain nombre de traitements "sur" les membres sont mentionnés : liste des membres, liste des activités d'un membre, liste des noms des membres assurant actuellement une fonction, état des cotisations... Ils suggèrent une liaison entre la classe *Membre* et les classes *Activité* (qui appartient sans doute aux paquetages *Pôles* ou *Séminaires*) et *Cotisation*.

Le système à développer devant, ceci est expressément demandé, automatiser les convocations aux réunions mensuelles du CA, une classe *Réunion* a été ajoutée. Chaque réunion est datée ; elle est d'une nature donnée, pouvant prendre une valeur parmi celles de l'ensemble {mensuelle, AG, AGE, provoquée}, cette dernière valeur correspondant à une convocation faite pour remplacer un membre du bureau. Un état lui est associé. Il peut prendre sa valeur dans l'ensemble {passée, prévues}.

Un délai de convocation, exprimé en nombre de jours, est associé à chaque réunion. Enfin, à toute réunion est associée une liste de personnes convoquées. Des votes ayant lieu lors des assemblées générales, un attribut *nb_votants* complète la description de cette classe. Cet attribut a une valeur nulle pour toutes les réunions qui ne sont pas des AG ou des AGE.

La prise en compte, par le logiciel, de la démission d'un membre du bureau à la demande d'au moins la moitié des membres actifs nécessite la création d'une classe *Démission*. Celle-ci correspond à la demande déposée par un (quelconque) membre actif. Son auteur n'est pas mémorisé. Seule la date de dépôt de la demande est conservée. La relation *concerne* qui lie *Démission* à *Membre* ne sert que dans le sens *Membre - Démission* pour compter le nombre de demandes déposées. Elle est donc unidirectionnelle.

Nous n'avons, pour l'instant, pas distingué les membres entre eux. Il n'est pas impossible que nous y soyons "contraints" et que nous ayons recours à une spécialisation des membres entre membres ordinaires, membres candidats, membres du CA et membres du bureau. L'avancement de l'étude va nous permettre de prendre position sur ce recours. De la même façon, les réunions sont probablement multiples, certaines nécessitant des votes, d'autres non. Une spécialisation est donc probable, permettant de mettre en évidence des réunions avec votes et d'autres sans.

III.2) Exemple de partie consacrée à l'analyse de la solution

L'analyse de la solution se présente après avoir terminé celle des besoins. Nous entrons dans « l'espace de la solution ». Il nous faut maintenant reprendre chaque scénario et se poser la question « comment faire pour mettre en œuvre cette interaction ? »

Ceci va passer par des échanges de messages, entre les acteurs extérieurs et le système (représenté, ici, par une fenêtre de contrôle) et entre les classes elles-même. Ces dernières sont « membres » du diagramme de classes. Il doit donc y avoir une cohérence extrême entre tous ces diagrammes.

Nous présentons dans ce qui suit deux formes de diagrammes d'interaction, les diagrammes de séquences et les diagrammes de collaboration. Nous terminons le document par la deuxième version du diagramme de classes, plus riche que la première et cohérente avec tous les schémas.

L'analyse est une activité qui produit une réponse aux exigences mises en évidence dans l'activité précédente. Cette réponse est une architecture logique, indépendante de l'environnement cible (logiciel et matériel), tout en conservant ses principes généraux (interfaces, environnement...). L'analyse (vue interne du système) « réalise » les cas d'utilisation (vue externe du système).

Nous procédons en deux étapes : esquisse de la solution et affinement de la solution.

I- ESQUISSE DE LA SOLUTION

Analyser la solution développée durant l'étape précédente revient d'abord à reprendre les scénarios et à préciser le rôle du système. L'objectif essentiel est de mettre en évidence les classes qui sont impliquées dans les traitements. Nous allons également faire apparaître plusieurs classes, certaines chargées de gérer l'interface *F. Gestion des Membres* ou les groupes d'instances (*Ens. Membres*). Nous obtiendrons des diagrammes de séquences.

I.1- Diagrammes de séquences

La convocation des membres de l'association à l'assemblée générale (scénario *Convoquer*) est définie par les scénarios des figures 11 et 12. Le diagramme de séquences est illustré par la figure 16.

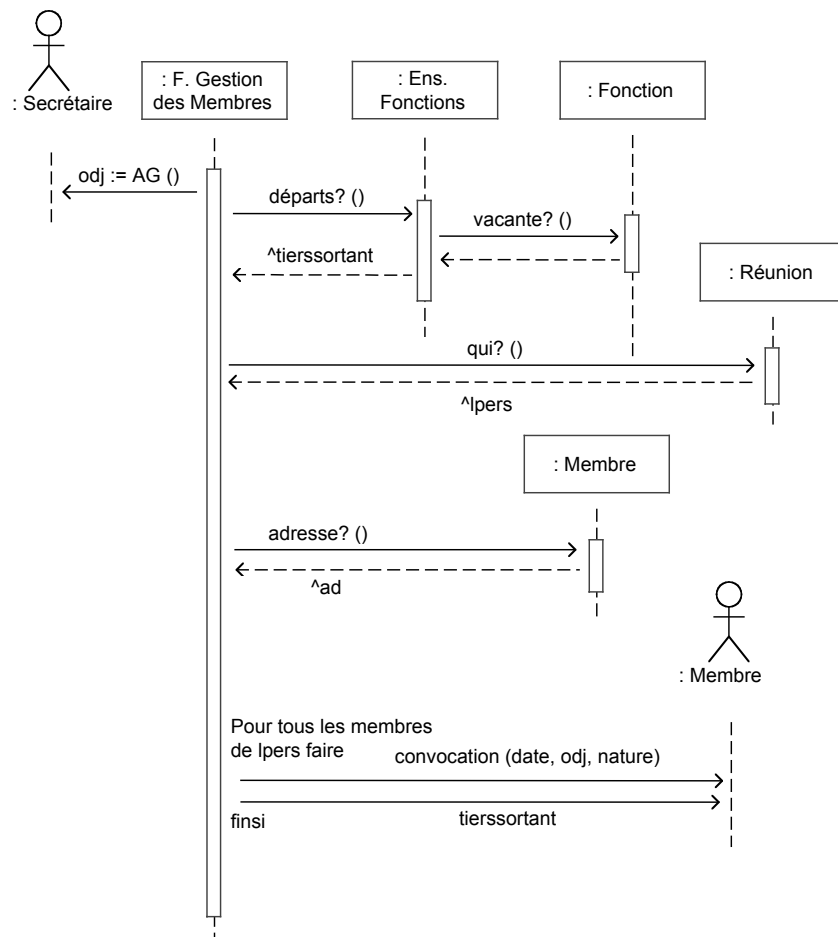


Fig. 16 - Diagramme de séquences Convoquer une AG

L'ordre du jour est fourni par le secrétariat ; la convocation est adressée à chaque membre de l'association, à jour ou non de sa cotisation ; la liste des membres à remplacer y est jointe. Une occurrence de la classe *Réunion* a été créée en début d'année.

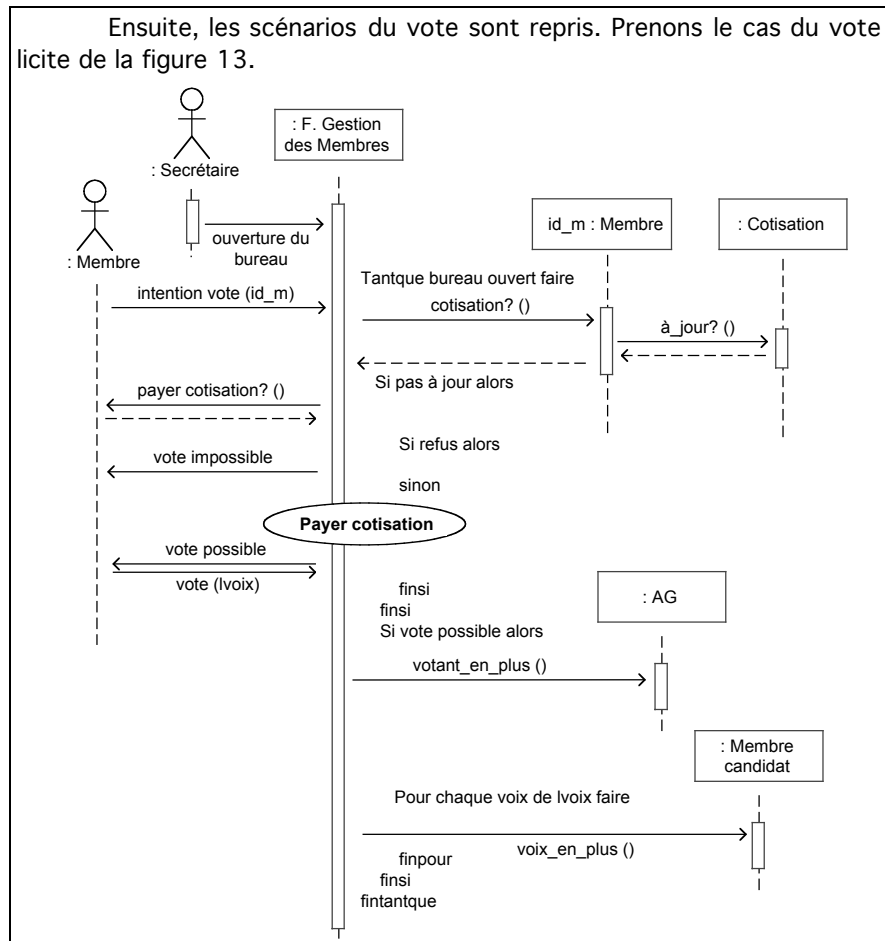


Fig. 18 - Diagramme de séquences Voter, consultation licite

Pour faire comptabiliser, par le système, les votants (ceci ne se fait qu'après la fermeture du bureau de vote), il faut avoir, auparavant, mémorisé les votes. Ceci n'était pas prévu, ni dans le scénario de la figure 13, ni dans le diagramme de classes de la figure 15. Un attribut *nb_voix* est ajouté dans la classe *Membre* (ou mieux, dans la classe *Membre candidat* spécialisant *Membre*), qui sert à enregistrer le nombre de voix obtenues. Cet attribut n'est utile que pendant les élections se déroulant lors d'une assemblée générale (AG). Il est d'une portée très restreinte et a une durée de vie (très) limitée.

Lors du vote, il y a vérification de la situation du membre votant. S'il a payé sa cotisation, son vote peut être comptabilisé. Il y a un votant de plus durant cette AG. Chaque voix est enregistrée. S'il ne l'a pas fait, il est invité à le faire. S'il paye sa cotisation, il peut voter, son vote sera comptabilisé. S'il ne le veut pas, il ne pourra voter. La première partie du scénario *Voter* devient donc :

Le décompte des votants est alors assez simple, puisque déjà réalisé dans la classe *Réunion*.

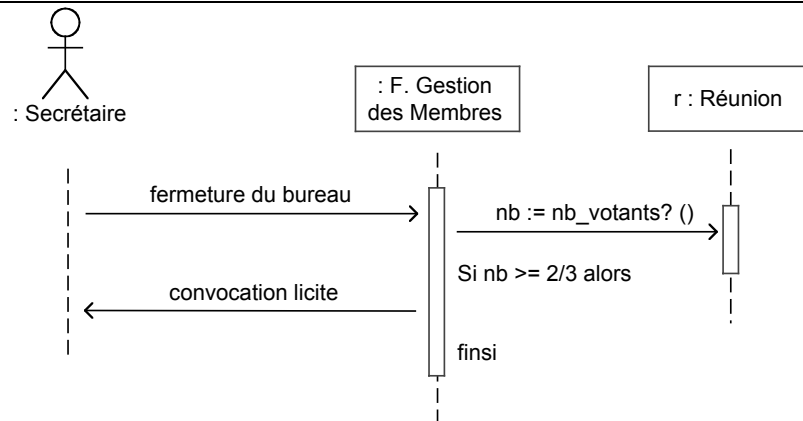


Fig. 17 - Diagramme de séquences Décompte

Nous venons de redéfinir deux des scénarios précédemment présentés. Ce travail de raffinement, nous allons le continuer, mais en changeant de formalisme.

1.2- Diagrammes de collaborations

La comptabilisation des voix est une partie du scénario *Voter* de la figure 13. Elle est développée dans la collaboration de la figure 19.

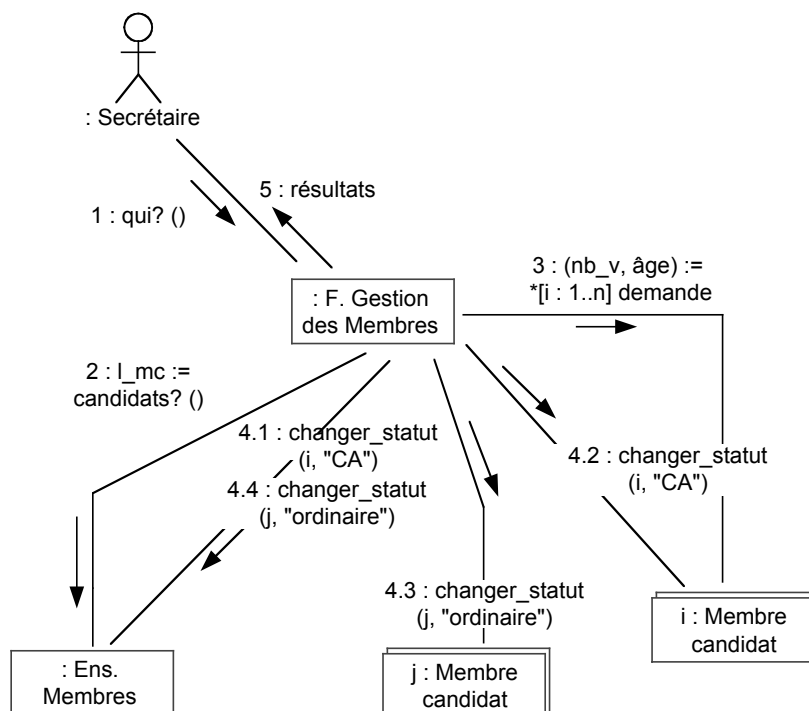


Fig. 19 - Diagramme de collaborations Voter, Comptabiliser les voix

xxxxxx suite de la description des diagrammes de collaboration xxxxxx

II- AFFINEMENT DE LA SOLUTION

Par affinement de la solution, nous entendons ici la reprise et l'enrichissement du modèle du domaine de la figure 15 par les nouveaux éléments obtenus par l'exploration systématique des interactions mises en évidence dans l'esquisse de la solution.

II.1- Diagramme de classes

Le diagramme de classes métier de la première version (voir la figure 15) s'est enrichi au cours de l'étude pour donner celui de la figure 24.

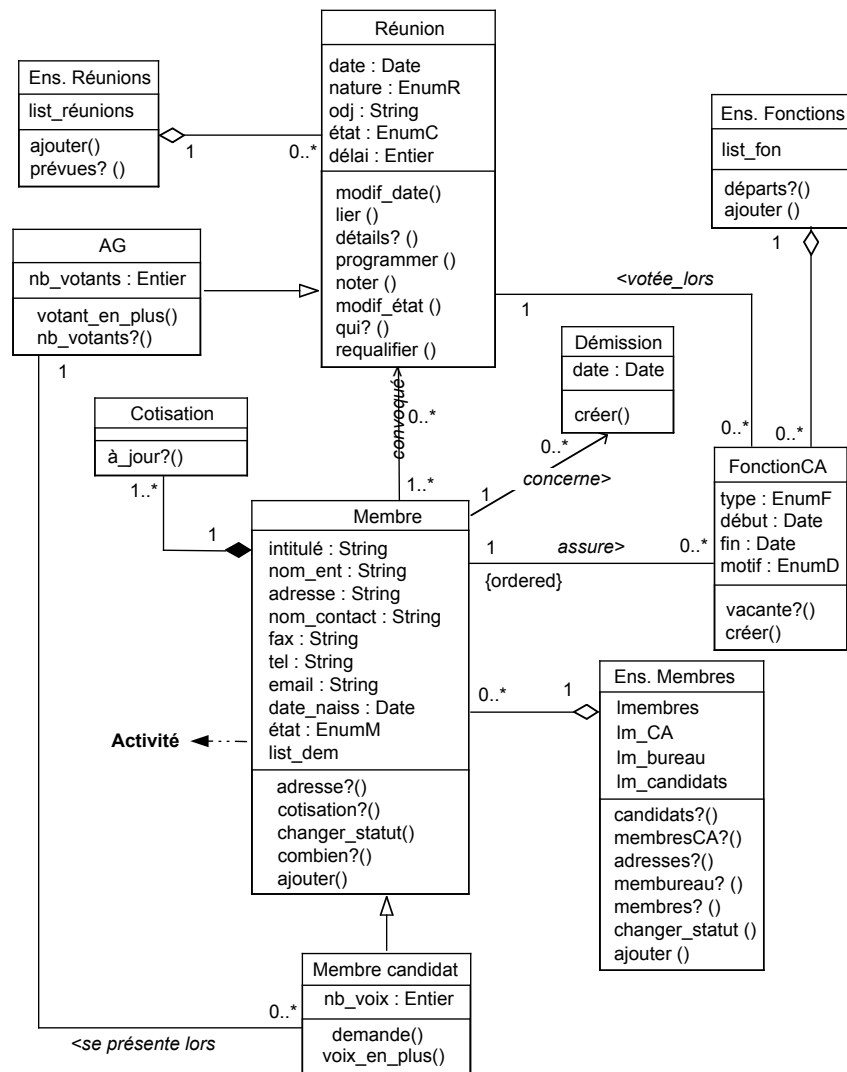


Fig. 24 - Diagramme de classes métier V2

Nous avons ajouté les spécialisations de *Membre* et de *Réunion*, les classes *Ens. xxx*, la relation entre *FonctionCA* et *Réunion*, celle entre *AG* et *Membre candidat*, les diverses opérations. Pour ces dernières, nous avons appliqué la "règle" qui veut que les messages soient pris en compte par le receveur et que cette prise en charge se traduise par une méthode, une opération. Chaque classe *Ens. xxx* est traitée de façon uniforme, avec un attribut correspondant à une liste de *xxx* et une méthode *ajouter ()* qui permet d'enregistrer une nouvelle occurrence à cette liste. Il

conviendrait d'y adjoindre aussi une opération de retrait. Nous laissons le lecteur le faire.

Un certain nombre de propriétés sont non persistantes. D'autres sont conservées plus longtemps. Il serait bon de mettre en évidence cet état de fait, en introduisant une notation particulière pour un des deux types de données.

La classe *F: Gestion des Membres* peut, elle aussi, être décrite, dans sa partie méthodes du moins :

F. Gestion des Membres
liste_dates () dat_réunion_démission () qui? () composition_bureau () vote () AG () fermer_bureau () ouvrir_bureau () demande_démission () intention_vote ()

Fig. 25 - Diagramme de classes F.Gestion des Membres

II.2- Diagrammes états-transitions

De toutes ces classes, celles dont le comportement dynamique est le plus intéressant sont sûrement *Réunion* et *Membre*. Modélisons leurs réactions par un (ou plusieurs) diagramme(s) états-transitions.

La classe *Réunion* dispose d'une variable *état* qui peut prendre deux valeurs, *prévue* et *passée*. Il est possible de décrire son comportement par un diagramme états-transitions :

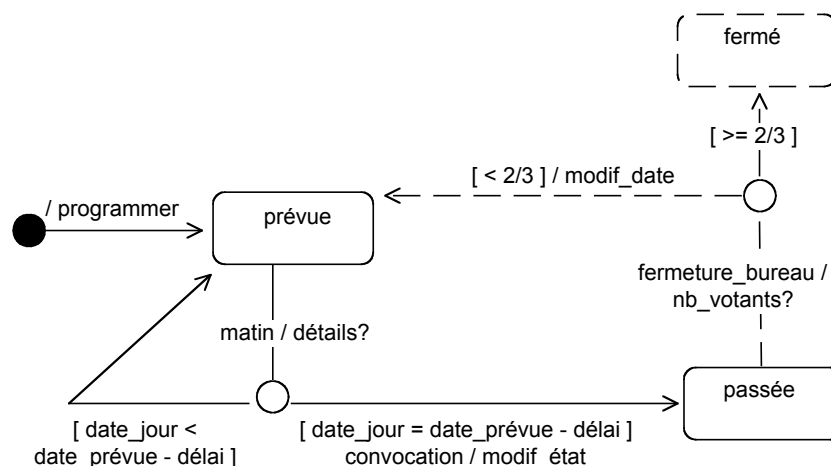


Fig. 26 - Diagramme états-transitions de la classe Réunion

La partie en pointillés correspond à la sous-classe AG. Nous avons là un exemple de spécialisation d'automates, cette classe AG ayant un automate composé d'une partie propre et d'une partie héritée.

La classe *Membre* "oscille", au premier niveau, entre les états *à jour* et *pas à jour* :

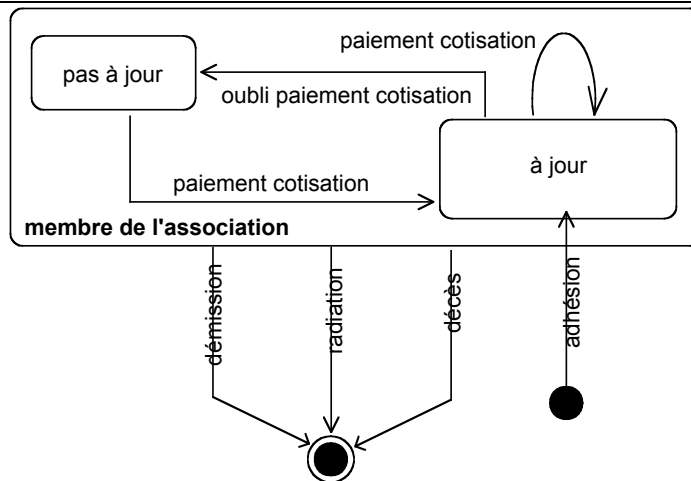


Fig. 27 - Diagramme états-transitions de la classe Membre

Les différents événements évoqués dans ce diagramme rythment la vie des membres. Il appartient à la personne qui reprendra cette modélisation en l'approfondissant d'associer, le cas échéant, des actions, des opérations, à ces transitions et de penser à modifier le diagramme des classes en conséquence.

L'état *à jour* peut être défini plus précisément : un membre à jour peut être ordinaire, candidat à une fonction (dans le cadre d'une AG), actif -il assure une fonction au sein du CA- ou membre du bureau. La figure 28 décrit l'intérieur de cet état hiérarchique séquentiel.

Comme pour le diagramme états-transitions de la classe *Réunion* (voir la figure 26) la partie en pointillés est spécifique de *Membre candidat*, celle-ci héritant de la super classe *Membre* tout le reste. Les gardes sont exprimées en clair. Il s'agit davantage de commentaires que de véritables conditions.

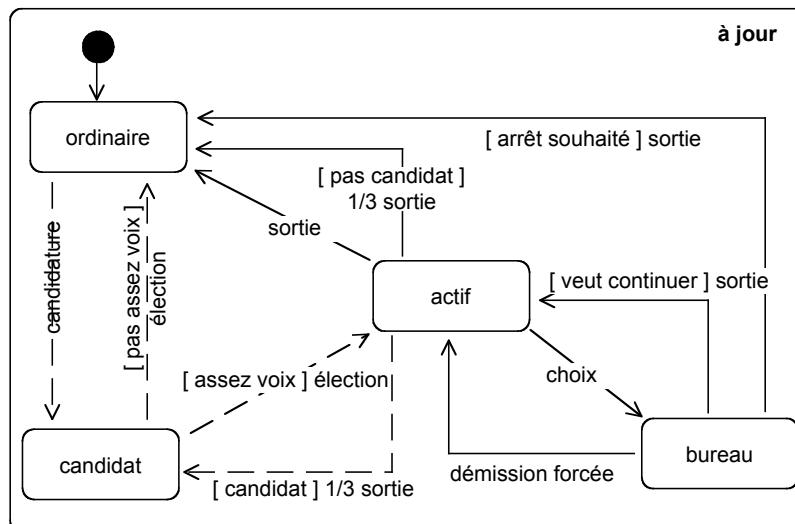


Fig. 28 - Diagramme états-transitions de la classe Membre, état à jour

Les événements du diagramme sont, le plus souvent, associés à des actions, et en particulier à des opérations de la classe. Nous ne les avons pas fait figurer sur le diagramme dans un souci de lisibilité.

EVENEMENT	OPERATION ASSOCIÉE
candidature	dépôt_candidature
élection	changer_statut
1/3 sortie	changer_statut
sortie	cessation_fonction
choix	changer_statut
démission forcée	radier_fonction

Sur ces quatre opérations, seule *changer_statut* est dans la classe. Les trois autres doivent y être ajoutées.

II.3- Diagrammes d'activités

Le diagramme d'activités sert à préciser l'ordre dans lequel les activités sont réalisées au sein d'un état (diagramme d'activité d'un état) ou entre les diverses classes (diagramme d'activité d'un processus). Nous l'avons associé (mais cela n'est pas une pratique conventionnelle) à la classe *F. Gestion des Membres* :

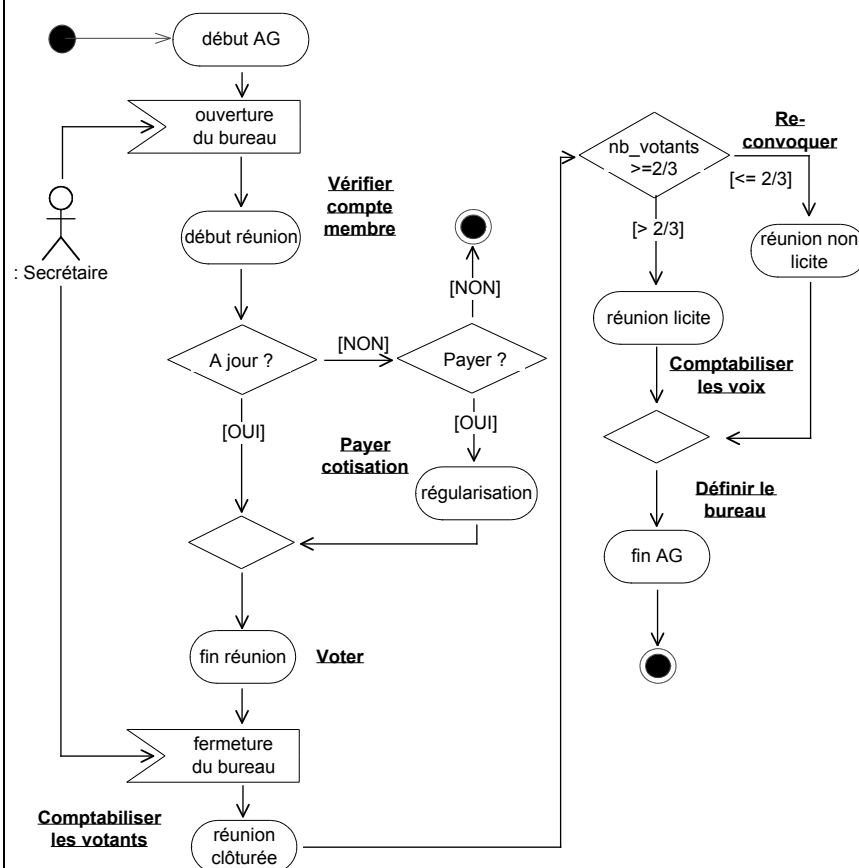


Fig. 29 - Diagramme d'activité explicitant le vote

On peut ainsi préciser le "fonctionnement" général de cette classe et associer à chaque activité un scénario ou une partie de scénario (ils apparaissent en gras, à côté des activités).

L'exemple que nous venons de présenter est partiel. Il y manque de nombreux diagrammes. Cela est volontaire. Il ne sert en effet à rien d'accumuler les schémas de même nature. Cet exemple souffre, en outre, d'un autre défaut. Il manque de recul (par rapport à notre pratique d'analyse).

L'idée qui nous a guidé était de faire le tour de ce que l'on peut mettre dans un dossier (et de montrer comme on pouvait le mettre) ; pas de faire le tour d'une application. Parce que, toutefois, nous pensons que cette exhaustivité peut intéresser quelques personnes et parce que nous avons demandé une prise de recul, nous complétons ce document, cette illustration, par la fourniture non pas d'un mais de trois dossiers traitant du même cas.

Ces dossiers n'ont pas été choisis au hasard. Un est bon, un autre moyen ; le dernier est mauvais. Les lecteurs pourront comparer.

IV) Que retenir de ce qui précède ?

Nous avons, dans ce document, fourni aux étudiants un plan-type des dossiers qu'ils doivent nous remettre. Pour compléter leur information (et, par delà, leur formation), nous avons également fourni un embryon d'exemple. Même si cet exemple est extrait d'un cas réel, il est suffisamment générique pour pouvoir être « resservi » à chaque nouvelle application. Il est constitué de deux parties, consacrées respectivement à une présentation des besoins et à une analyse de la solution proposée.

Le point essentiel à retenir de cette « présentation » est qu'un dossier est avant tout destiné à être lu. Il doit donc être lisible, c'est-à-dire composé de phrases correctement rédigées, sans faute d'orthographe ni de lourdeur. Les dessins (qu'il s'agisse d'un diagramme de classes, d'un diagramme d'activités...) ne sont là que pour illustrer un texte. Ils ne sauraient le remplacer.

« Alors, il va falloir écrire ? » nous demandait un jour un étudiant. Eh bien oui, il va falloir écrire...
