MASTER EN SCIENCES INFORMATIQUES

Gestion des documents et du flux de travail dans un système de workflow inter-organisationnel

Taes, Frédéric

Award date:
1996

Awarding institution:
Universite de Namur

Facultés universitaires Notre-Dame de la Paix
Institut d'informatique
Rue Grandgagnage, 21 - B-5000 Namur

Gestion des documents et du flux de travail dans un système de workflow inter-organisationnel

Promoteur:
M^r François Bodart

Mémoire présenté par Frédéric Taes
en vue de l'obtention du grade de
Licencié et Maître en Informatique

Remerciements

Nous remercions Messieurs F. Bodart et W. Poos
qui nous ont dirigé tout au long de la réalisation de ce travail.

Nous remercions également nos parents
qui nous ont permis de faire nos études.

Résumé

Le but de ce mémoire est de contribuer à la conception d'un système de workflow inter-organisationnel. Tout d'abord, nous présenterons ce qu'est un système de workflow, et quels sont les aspects spécifiques des contextes intra- et inter-organisationnels, prédéfini et ad-hoc, multimédia, etc. Nous présenterons alors le projet W3Flow et une application particulière : le «permis de bâtir».

Plus précisément, nous élaborerons ensuite une proposition pour la gestion des documents, et pour la gestion du flux de travail. Notre contexte sera celui de l'application de permis de bâtir : inter-organisationnel et assez largement prédéfini. L'implémentation devrait être basée sur la Suite Internet et sur le concept d'Intranet.

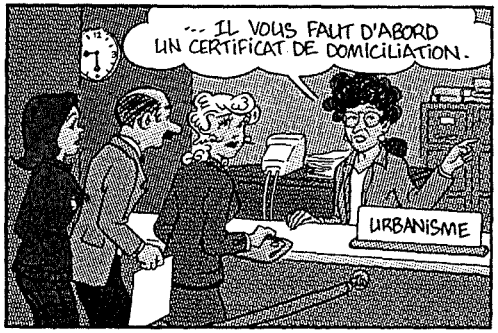
Mots-clés : workflow, inter-organisationnel, CSCW, coordination, collaboration, multimédia, Intranet, Internet Suite, groupware.

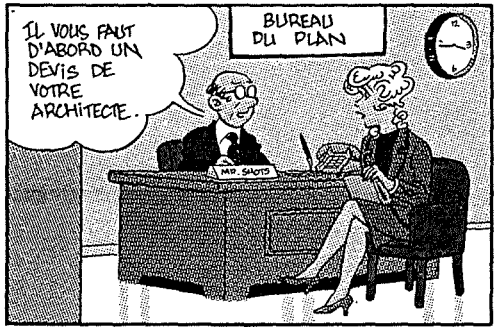
Abstract

The goal of this thesis is to contribue to the design of an inter-organizational workflow system. First, we'll present what's a workflow system, and what are the specific aspects of multimedia, predefined or ad-hoc, intra- and inter-organisational contexts. Then, we'll present the W3Flow project and a particular application : the «licence to build».

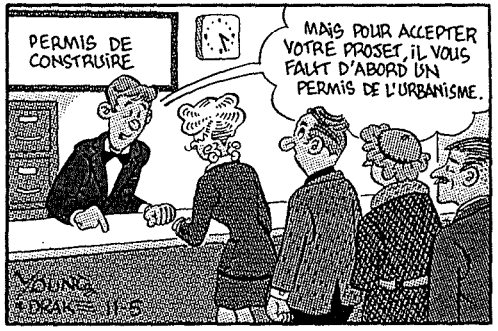
More precisely, we'll make a proposal for the management of documents, and for the management of the flow of work. Our context will be this one of the licence-to-build application : inter-organizational and widely predefined. The implementation would be based on the Internet Suite and the Intranet concept.

Keywords : workflow, inter-organizational, CSCW, coordination, collaboration, multimedia, Intranet, Internet Suite, groupware.





TOP 228



Chapitre I - Les Systèmes de Workflow

Le but de ce premier chapitre est d'introduire les différents concepts de base du *workflow*, ou *flux de travail*. Nous commencerons par signaler les principaux problèmes posés en matière de travail collaboratif [I.1.]. Ensuite, nous définirons des termes usuels [I.2.] et plus particulièrement celui de workflow [I.3.]. Nous verrons alors quelles sont les composantes d'un flux de travail [I.4.] et quelles sont les fonctionnalités des systèmes de workflow [I.5.].

I.1. Constat de départ

La division du travail, appliquée au domaine administratif, a cloisonné entre elles des tâches formant un tout. C'est ainsi que pour le traitement de dossiers, comme une demande de permis de bâtir par exemple, quantité de personnes et de tâches interviennent. Le problème est que, pour assurer la cohérence de l'ensemble de la procédure, des mécanismes de coordination deviennent nécessaire. Ces mécanismes sont d'autant plus difficile à mettre en place si l'on se situe dans un contexte inter-organisationnel.

Citons quelques problèmes posés [TAES96, page 4] :

- *l'hétérogénéité des systèmes* : cette hétérogénéité, tant matérielle que logicielle, demande dans certains cas de multiples ré-encodages d'une même information;
- *la circulation des documents* : on estime que dans la majorité des procédures administratives, le travail effectif ne représente que 15% du temps total nécessaire au traitement d'un dossier;
- *l'éparpillement, la duplication des informations* : cette duplication est généralement nécessaire, afin que chaque service intervenant détienne une copie de chaque document; si l'on évite la duplication, on est alors confronté au problème de recherche de l'information au sein de différents services;
- *la synchronisation* : il est des cas où une tâche ne peut démarrer que lorsque plusieurs conditions sont réunies. Ces conditions sont, souvent, l'achèvement d'exécution d'autres tâches, préliminaires;
- *la communication* : diverses personnes impliquées dans l'exécution de tâches demandent des mécanismes de support de communications structurées, supports qu'elles ne trouvent pas.

I.2. Workgroup, Groupware et Coopération

I.2.1. Workgroup

La première idée pour résoudre les problèmes posés est le partage de l'information et de certaines ressources, par groupes de travail. Ces groupes de travail sont appelés "*workgroup*". Un exemple est «Windows for workgroups» de Microsoft, qui permet depuis une machine d'accéder aux informations et ressources (imprimantes, disques,...) présentes sur l'autre machine. Des fonctionnalités supplémentaires relatives à la sécurité sont en outre fournies, par exemple pour identifier les utilisateurs et restreindre certains accès.

Ce concept de workgroup contribue donc à régler les problèmes d'échange de documents, au sein d'un même groupe de travail.

Mais cela ne suffit pas toujours : les tâches peuvent être très diverses, le nombre d'acteurs importants, et les documents très divers. Ainsi, la notion de workgroup n'inclut pas la répartition des tâches, ni le suivi d'avancement (voir où en est chaque procédure), etc. En fait, les systèmes de workgroup ne gère que les accès aux documents et ressources, mais d'un point de vue statique. L'envoi de documents par exemple, appelé aussi routage, reste effectué manuellement.

I.2.2. CSCW

Le terme CSCW est l'acronyme de *«Computer-Supported Collaborative Work»*; c'est le nom générique donné aux systèmes informatisés gérant le travail collaboratif.

I.2.3. Coopération

Le concept de coopération peut être approché par deux modèles classiques.

Le premier modèle envisagé dans un contexte de coopération est celui de "l'*entreprise étendue*" inter-administrations : l'ensemble des acteurs participants à des tâches administratives sont vu comme faisant partie d'un même ensemble, d'une seule organisation, même si en réalité ces acteurs font partie de plusieurs organisations [TWI, pg.18; en réalité, repris de Konsynski93].

Le second modèle envisagé dans un contexte de coopération est celui de la chaîne de valeurs de Michael Porter [TWI, pg.20] : la chaîne de valeur est un système d'activités interdépendantes reliées entre elles. Ces liens entre activités se manifestent lorsque la performance et l'exécution d'une activité influe sur les autres.

Il existe différentes typologies de la coopération[1]; nous allons synthétiser ici celle de De Michelis [DEMI94-95]. Sa taxonomie s'inscrit dans le cadre d'un modèle conversationnel étendu, dont nous rappelons ici les éléments essentiels.

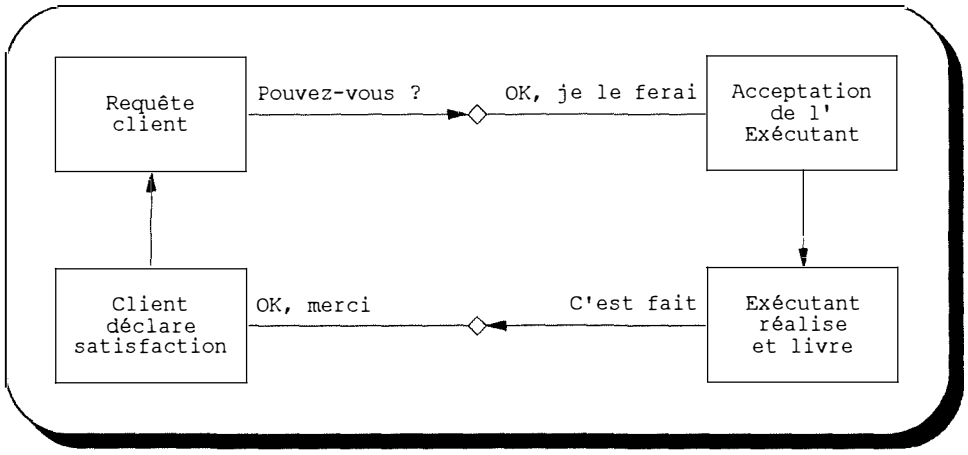


Figure I-1 - The Action Workflow Model

Un **work process** est vu comme des relations de communications - incluant si nécessaire les transformations Entrée/Sortie - entre celui qui requiert un service (le client) et celui qui l'exécute (l'exécutant, fournisseur).

Plus généralement, un work process peut être vu comme une relation impliquant plusieurs clients et plusieurs exécutants. Le type de relation entre les acteurs d'un work process peut être caractérisé du point de vue de la position qu'ils occupent au sein de ce work process. En effet, toute forme de relation de peut être caractérisée par une forme spécifique de coopération entre participants :

- **Collaboration** : forme de coopération entre clients et exécutants d'un work process. C'est la forme de base de coopération au sein de work processes. Les personnes collaborants interagissent en faisant quelque chose ensemble, créant une certaine valeur. Du succès de la collaboration naît un langage commun et une compréhension commune entre les clients et les exécutants.

- **Co-décision avec rôles égaux** (=coopération collégiale): forme de coopération entre clients d'un work process. Elle précèdent logiquement l'action : si elle est menée à bien, elle fixe les objectifs communs. La co-décision avec rôles identiques est un processus de décision au sein duquel tous les participants partagent pleinement leurs responsabilités pour la décision qui sera prise.

- **Coordination** : forme de coopération entre acteurs de différents sous-processus d'un work process. Elle permet l'intégration de différentes sous-tâches faisant partie d'une tâche commune plus générale. Le succès de la coordination dépend du degré de prise de conscience du contexte (inter)-organisationnel des participants.

[1] POOS94 : coopération répartie, coopération collégiale.

- **Co-décision avec rôles distincts** (= coopération répartie) : forme de coopération entre exécutants d'un work process. C'est un processus de décision où chaque participant doit reconnaître les actes des autres participants; il doit assumer pleinement sa part de décision, compte tenu des contraintes définies par les autres participants.

I.3. Le concept de workflow

Cette section introduit la notion de workflow (I.3.1.), en insistant sur la différence entre types et instances de workflow (I.3.2.). Les différentes typologies de systèmes de workflow sont alors brièvement exposées (I.3.3.), ainsi que la distinction entre workflow intra- et inter-organisationnel (I.3.4.).

I.3.1. Définition générale de «workflow»

Un workflow est un flux de travail, c'est-à-dire un séquencement de tâches relatif à une même instance d'une procédure administrative (que l'on appelle aussi «dossier»). Les objectifs des systèmes de workflow sont [AGOS95, pg.15]: *« d'aider les organisations à spécifier, exécuter, suivre (to monitor), et coordonner les éléments de flux de travail dans un environnement bureautique distribué. »*

Le Workflow s'inscrit aussi dans un contexte de coopération. Il prend également avantage de l'infrastructure en réseau (Internet), des environnements intégrés (Windows, World-Wide-Web), de la gestion de groupe, et de la communication (e-mail, vidéo conférences, etc.).

I.3.2. Types et instances de workflow

Avant de poursuivre, distinguons clairement un type de flux d'une instance de flux.

Un type de flux, c'est la définition d'un flux indépendamment d'un flux particulier, indépendamment de données particulières, indépendamment d'une exécution particulière. Un type de flux s'entend donc a priori et se veut plus ou moins générique. Par exemple, un type de flux dans notre système pourra être «permis de bâtir».

Une instance de flux par contre, comprend non seulement la définition du flux, mais aussi les données et l'historique du flux; bref, une instance de flux s'entend à l'exécution. Une instance d'un flux est particulière, unique. Par exemple, une instance de flux dans notre système pourra être «permis de bâtir N°123 de Mr. Dupond.»

I.3.3. Typologies de systèmes de workflow

Différents types de workflow existent, selon diverses typologies [TWI, pg.125-135]. Une première typologie propose de classer les produits actuellement disponibles en trois grandes catégories:

- **process centric** : la majorité des applications workflow actuelles utilisent des produits de ce type. Ces produits dépendent souvent d'un SGBD particulier -comme Oracle, Sybase, ou Informix- pour stocker les données et définitions du workflow. Ils ressemblent beaucoup à des langages de programmation de haut niveau, mais n'imposent pas à l'utilisateur de nouveaux outils pour le développement d'applications workflow. Ce sont des outils sous-jacents pour la gestion de workflow (inter-)organisationnels.

- **mail centric** : les produits de cette catégorie insistent sur les bénéfices de l'utilisation de modes de communication électroniques existants -principalement l'*e-mail*- comme service de messagerie et de distribution pour un système de workflow. Beaucoup de ces produits incluent des outils élaborés de *gestion de formulaires*, voire des *agents intelligents* pour accomplir certaines tâches de workflow.

- **document centric** : ces produits insistent sur le *document en tant qu'unificateur* d'un processus de workflow. Les documents sont associés à leurs propriétaires, applications, et règles qui gouvernent leurs routage et traitements.
    * Beaucoup de ces produits proposent des fonctions pour la gestion des documents : check-in (réception), check-out (envoi),...
    * D'autres étendent la métaphore traditionnelle du document dans un classeur ou dossier (=file in a folder), en associant aux classeurs des règles de traitement.

Une autre taxonomie met l'accent sur le degré de structuration des tâches :

♦ *pour les tâches peu ou pas structurées* :
- **ad hoc** : la répartition des tâches se fait au cas par cas; ceci convient uniquement pour des groupes de travail dynamique ou presque chaque dossier doit être traité différemment;
- **heuristique** : un moteur d'intelligence artificielle s'occupe d'établir la répartition des tâches, le routage, etc.; ce type de workflow semble encore actuellement assez théorique;

- *pour les tâches plus structurées* :
  - **orienté objet** : basé sur la métaphore de l'orienté-objet ("Tout est objet"); un document contient à la fois de l'information et des règles (= méthodes) qui lui sont propres. Chaque document est une instance d'un type objet de base; ce type objet de base contient un ensemble de règles génériques, communes aux documents ou à un type de documents donnés. En résumé, on définit un workflow général, qui sera instancié pour chaque document particulier. Les tâches doivent être assez structurées pour permettre de définir des types d'objet.

- *pour les tâches fortement structurées* :
  - **transactionnel** : l'environnement est statique mais complexe, et les règles qui définissent un tel workflow peuvent être définies de manière précise et exécutées selon un modèle transactionnel. Comme les documents suivent toujours le même chemin, ce type de workflow aide simplement à assurer l'intégrité du processus, et le rendement est la préoccupation principale.

I.3.4. Workflow intra- et inter-organisationnels

Les systèmes de workflow actuels, y compris les groupwares comme Lotus Notes, supportent essentiellement des workflow intra-organisationnels, c'est-à-dire où toutes les tâches de la procédure sont exécutées au sein de la même organisation.

Les systèmes de workflow inter-organisationnels au contraire, supportent des situations de collaboration entre plusieurs organisations. Le contexte inter-organisationnel est un contexte spécifique, qui introduit de nouveaux problèmes, de par sa nature.

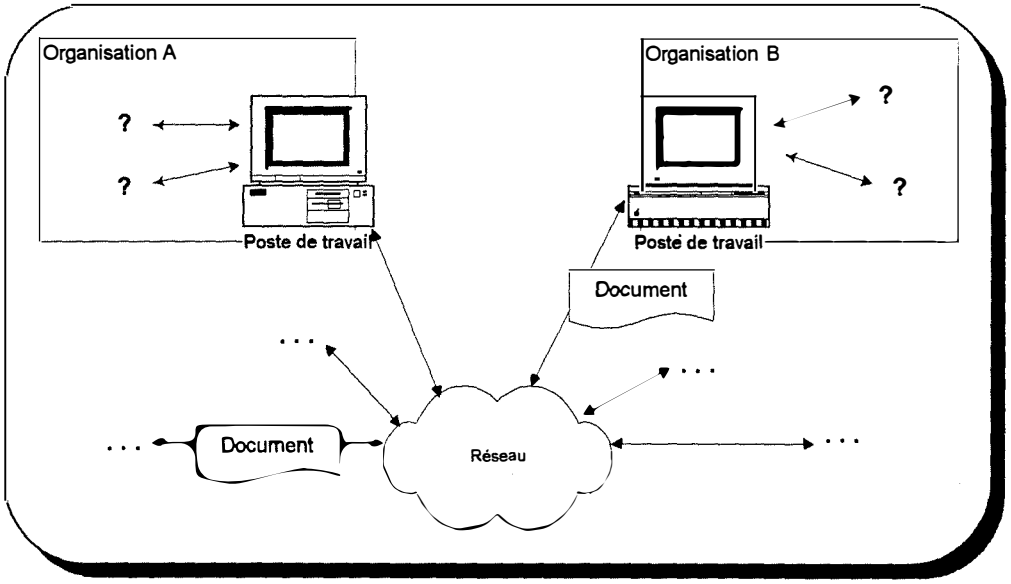


Figure I-3 : L'espace inter-organisationnel

Les problèmes intra-organisationnels sont amplifiés dans un contexte inter-organisationnel, par sa taille:

- la multiplicité des partenaires;
- l'hétérogénéité des environnements;
- l'éparpillement, la duplication des informations;
- ...

Certains problèmes sont nouveau et propres au contexte inter-organisationnel, car les entreprises y sont toutes **autonomes** :

- il n'y a généralement pas de «hiérarchie» entre les organisations, ce qui a des répercussions tant sur la circulation des informations, que la gestion des conflits (qui peut «trancher»?);
- dans le contexte de collaboration, aucune organisation n'est «neutre»: chacune tend à développer son «pouvoir» sur les autres; dès lors, pour être bien accepté par tous, tout système de gestion de la coopération devra être indépendant de toute organisation utilisant le système;
- les organisations suivent leur propre évolution technique, indépendamment des autres; il n'est pas possible d'imposer à une organisation un environnement informatique particulier, ni un mode d'organisation du travail;
- les organisations sont des boîtes noires dont on ne voit que les entrées et sorties (les documents), et les postes de travail et acteurs qui interagissent avec les autres organisations. Le reste nous est caché.
- le problème de l'interfaçage devient une pierre d'achoppement : non seulement les organisations utilisent leur systèmes particuliers, mais elles acceptent généralement mal un système externe qui interviendrait directement sur leur propre système (en cas de problème, qui serait responsable?).

I.4. Composition d'un flux de travail

Un flux de travail peut être vu comme un enchaînement de tâches relatives à un même dossier administratif, dans le cadre d'une procédure administrative. Un exemple de procédure administrative est l'attribution d'un permis de bâtir; un dossier serait par exemple la demande n°123 introduite par Mr. Dupond.

Le schéma I-4 ci-après illustre diverses composantes d'un flux de travail.

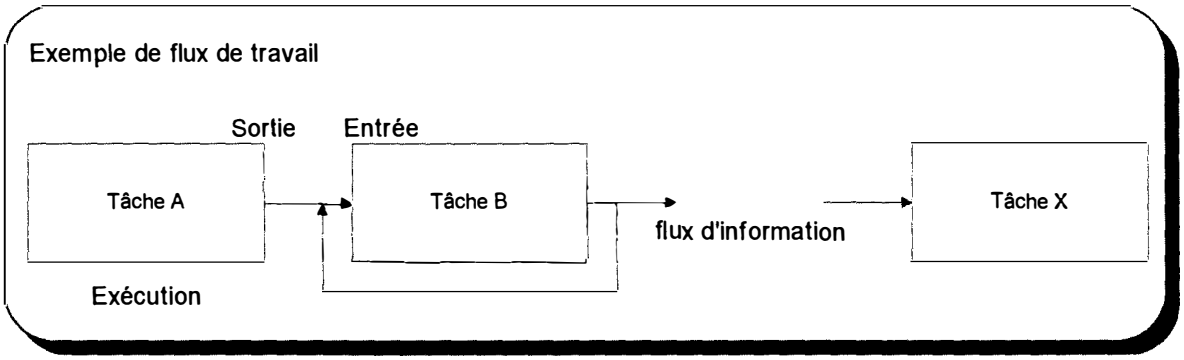


Figure I-4 : Exemple de flux de travail

Nous pouvons y distinguer plusieurs éléments-clés, donnés ici pour un environnement intra-organisationnel:

- **les tâches**, qui sont les unités élémentaires de décomposition du flux de travail;
- **les postes de travail**, qui sont les lieux d'exécution des tâches, et leurs ressources nécessaires;
- **les acteurs**, qui sont les personnes qui exécutent les tâches, et qui assument certaines fonctions appelées **rôles**;
- **les Entrées/Sorties** (ou E/S), qui sont respectivement les informations reçues en entrée pour l'exécution d'une tâche, et les informations générées en sortie par l'exécution d'une tâche;
- **les enchaînements de tâches**, qui ne sont pas toujours simples, séquentiels; des tâches peuvent par exemple s'exécuter en parallèle;
- **les conversations**, qui sont des suites d'échanges de messages entre deux acteurs, dans un but précis; ces conversations peuvent concerner des exécutions de tâches, ou une obtention d'information.

Il est important de remarquer les relations étroites entre ces différents éléments. Ainsi, les E/S modélisent la circulation d'informations entre les différents postes de travail. Les E/S sont aussi déterminantes dans l'enchaînement des tâches : pour qu'il y ait circulation d'information entre deux tâches, il faut aussi que les sorties informationnelles d'une tâche correspondent à l'entrée de la tâche suivante[2].

Dans un contexte inter-organisationnel, certains concepts diffèrent :

- **les postes de travail** ne sont plus nécessairement les lieux d'exécution des tâches, mais bien des passerelles entre l'environnement inter-organisationnel et un environnement intra-organisationnel particulier;
- **les acteurs** tels que nous les voyons ne sont plus nécessairement les personnes qui exécutent les tâches : l'assignation des tâches au sein d'une organisation ne nous concerne pas. En outre, de nouveaux concepts s'ajoutent : **groupe** de personnes, **organisation**, etc.

[2]= c'est la fonctionnalité d'interfaçage que nous présenterons dans le chapitre 2

- **les conversations**, n'ont plus nécessairement lieu entre deux acteurs physiques, mais davantage entre groupe, ou entre personnes assumant des rôles au sein de ces groupes (ex. de rôle: chef de projet).

I.5. Fonctionnalités des systèmes de workflow

Les fonctionnalités offertes par les systèmes de workflow dépendent de leur type. Par exemple, les fonctionnalités offertes par un système de workflow intra-organisationnel et *mail-centric*, gérant des tâches fortement structurées, seront différentes des fonctionnalités offertes par un système de workflow inter-organisationnel ad-hoc et *document-centric*.

Néanmoins, il est possible de classer ces fonctionnalités à partir de la définition générale des objectifs des systèmes de workflow [I.3.1.] : *« aider les organisations à spécifier, exécuter, suivre (to monitor), et coordonner les éléments de flux de travail dans un environnement bureautique distribué. »*

- **Spécification du flux de travail** : le système de workflow doit fournir un moyen de spécification relatif à l'agencement, à l'affectation ainsi qu'à la réalisation des tâches. Il doit également fournir un moyen de spécification des objets qu'il manipule : dossiers, documents, acteurs, groupes de travail, etc. y compris la spécification des droits d'accès, responsabilités, et assignations concernant ces objets. En outre, des fonctionnalités de validation des spécifications, aux niveaux syntaxique et sémantique, sont souhaitables.
- **Exécution du flux de travail** : le système de workflow doit supporter l'exécution du flux de travail. Il doit pouvoir gérer l'initialisation d'un flux de travail, l'affectation des tâches, le routage des documents, etc. A cela s'ajoutent des fonctionnalités d'aide à la gestion du travail, incluant la gestion des droits d'accès aux informations, la recherche d'informations (documents, acteurs,...), d'identification des partenaires, etc.
- **Suivi de l'exécution** : le système de workflow doit aussi pouvoir gérer le suivi de l'exécution des tâches, en permettant de rendre compte de l'état d'avancement notamment grâce à des «to-do-lists» (liste des tâches à effectuer). Il doit aussi pouvoir gérer les délais et retards, et mémoriser les événements relatifs au flux (date de début et fin de tâches, documents, etc.). Il doit également fournir un moyen de consulter et de retrouver ces informations.
- **Coordination** : le système de workflow doit assurer la coordination sur tous les plans: coordination entre tâches (ex.: synchronisation), entre documents (dont interfaçage), entre acteurs (ex.: grâce à des agendas partagés), entre bases de données des différents partenaires (acteurs ou organisations), etc.

Dans la suite de ce document, nous nous concentrerons davantage sur les systèmes de workflow inter-organisationnels. Il est donc intéressant de noter dès à présent quelques différences essentielles entre les fonctionnalités offertes par les systèmes intra-organisationnels et les systèmes inter-organisationnels.

	WF intra-organisationnel	WF inter-organisationnel
Environnement typique	distribué, avec réplication des bases de données	Client/Serveur centralisé
Gestion de l'organisation	= gestion des acteurs et rôles	= gestion des acteurs, rôles, groupes et organisations
Gestion des tâches	assignation des tâches aux acteurs	assignation par rôle/groupe
Gestion du travail	agenda partagé, gestion de rendez-vous,...	pas d'agenda partagé
Gestion des informations	assuré par un système propriétaire	système ouvert, avec fonctionnalité d'interfaçage

Note: la réplication est le mécanisme qui permet d'assurer l'homogénéité d'une base de données distribuée sur différents postes de travail.

Chapitre II - W3flow et l'application permis de bâtir

Dans ce chapitre, nous allons tout d'abord présenter W3Flow, un système inter-organisationnel de gestion de la coopération [II. 1]; nous présenterons notamment ses fonctionnalités. Puis, nous présenterons l'application «permis de bâtir», une application pilote que notre système de workflow devra être capable de gérer [II.2]. Ensuite, nous présenterons et préciseront les modèles de W3Flow dans notre optique «permis de bâtir» [II.3]. Nous présenterons alors quelques aspects techniques du système que nous allons développer [II.4]. Enfin, nous ne manquerons pas de le comparer avec d'autres systèmes existants, tels que Lotus Notes et The Milano System [II.5].

II.1. Présentation générale de W3Flow

II.1.1. Introduction à W3Flow

W3Flow est le nom d'un projet industriel en cours de recherche et développement, axé sur la conception d'un système de workflow inter-organisationnel [POOS96]. Le contexte inter-organisationnel est en effet peu supporté, et des standards tels que EDI[3] sont actuellement trop limités que pour répondre aux besoins inter-organisationnels. Nous avons dressé un petit tableau comparatif entre les systèmes actuels habituels, et W3Flow [figure II-1], montrant ainsi les objectifs et le champ d'application de W3Flow.

Autres systèmes	W3Flow
Intra-organisationnel	**Inter-organisationnel** (système de workflow étendu)
Données purement textuelles (ex.: EDI), informations fortement structurées	**Formulaires multimédia** (avec outils auteur)
Situations fortement prédéfinies	**Coopération** (répartie) **flexible**
Systèmes propriétaires, souvent chers	**Internet Suite** (gratuit) & **Intranet** (= groupware & travail collaboratif utilisant les technologies Internet)
Incompatibilités entre systèmes	**Multi-plateformes** (Internet Suite) + fonctionnalités d'**interfaçage entre systèmes intra-organisationnels**

Figure II-1 : Comparaison entre les systèmes traditionnels et W3Flow

Par rapport à nos typologies présentées dans le chapitre précédents, W3Flow est donc un système de workflow *inter-organisationnel*, *document centric*, et *orienté objet*.

[3] EDI = Electronic Data Interchange [LOBET95]

II.1.2. Les fonctionnalités de W3Flow

Les fonctionnalités proposées par W3Flow ont trait à [POOS96, pg. 16-19] :

- la gestion des documents;
- la gestion des flux et dossiers;
- la gestion des conversations;
- la gestion de l'audit et de la consultation de l'historique;
- la gestion de la sécurité;
- la gestion de l'interfaçage entre systèmes inter- et intra-organisationnels.

1.2.1. Gestion des documents

Les fonctionnalités de base relatives à la gestion des documents sont :

- la création de types de formulaires ou de document-types, à l'aide d'un langage auteur graphique;
- l'utilisation de la gestion des partenaires pour permettre l'accès à des documents à des partenaires;
- l'impression, la copie et la visualisation du contenu de documents;
- l'envoi d'un document à un partenaire.

1.2.2. Gestion du flux et des dossiers

Les fonctionnalités propres à la gestion du flux reposent sur un ordonnanceur de tâches (scheduler) capable de :

- gérer les déclenchements d'événements en fonction du temps;
- mettre à jour les états des tâches et dossiers conformément à leur cycle de vie;
- évaluer les conditions de séquencement de tâches;
- exécuter un enchaînement prédéfini de tâches;
- exécuter un séquencement ad-hoc de tâches résultant de décisions des acteurs sur la tâche suivante la plus appropriée;
- permettre la définition de tâches ad-hoc par les acteurs;
- exécuter le routage des messages et documents depuis le système W3Flow vers et depuis les systèmes d'informations intra-organisationnels;
- utiliser le gestionnaire de conversations pour la gestion des messages relatifs aux séquencements de tâches.

Les autres fonctionnalités de gestion du flux et des dossiers supportent la gestion du travail, c'est-à-dire de :

- mettre à jour les listes des travaux à faire (to-do lists) par acteur et par dossier, et permettre aux acteurs de les visualiser;
- permettre aux acteurs de visualiser l'état d'avancement des tâches, par dossier.

1.2.3. Gestion des conversations

Les fonctionnalités proposées par W3Flow pour gérer les conversations sont de :

- lancer les conversations pour information;
- lancer les conversations pour actions, lesquelles peuvent résulter de :
  - l'exécution d'un séquencement prédéfini de tâches;
  - l'exécution d'un séquencement ad-hoc de tâches décidé à l'exécution par un acteur;
- détecter les messages envoyés, d'envoyer des messages, et de mette à jour les états des conversations (à l'aide d'un «conversation scheduler»).

1.2.4. Gestion de l'historique

Ces fonctionnalités permettent de :

- spécifier quels objets doivent être mémorisés (ex.: document, tâche, acteur, etc.), ainsi que les attributs à mémoriser (ex.: date de création pour un document), et les conditions de mémorisation (ex.: garder l'info. pendant combien de temps);
- retrouver des objets mémorisés (archivés).

1.2.5. Gestion de la sécurité

Les fonctionnalités relatives à la gestion de la sécurité proposées par W3Flow peuvent être classées en trois catégories :

- l'authentification, qui permet d'identifier les partenaires et postes de travail interagissant avec le système W3Flow;
- la confidentialité, qui permet d'empêcher que les données stockés et échangées par W3Flow ne puisse être lues par des tiers;
- l'administration au sens large, qui inclut la gestion des clés et mots-de-passe, mais également d'autres services à définir comme la sauvegarde des données sur site distant.

1.2.6. Interfaçage de W3Flow avec les systèmes d'information intras

L'interfaçage n'est pas à proprement parler une catégorie de fonctionnalités à part, mais son importance est cruciale pour un système de workflow inter-organisationnel [voir chapitre 1].

W3Flow localise les fonctionnalités d'interfaçage au niveau des postes de travail des acteurs comme l'illustre la figure II-2.

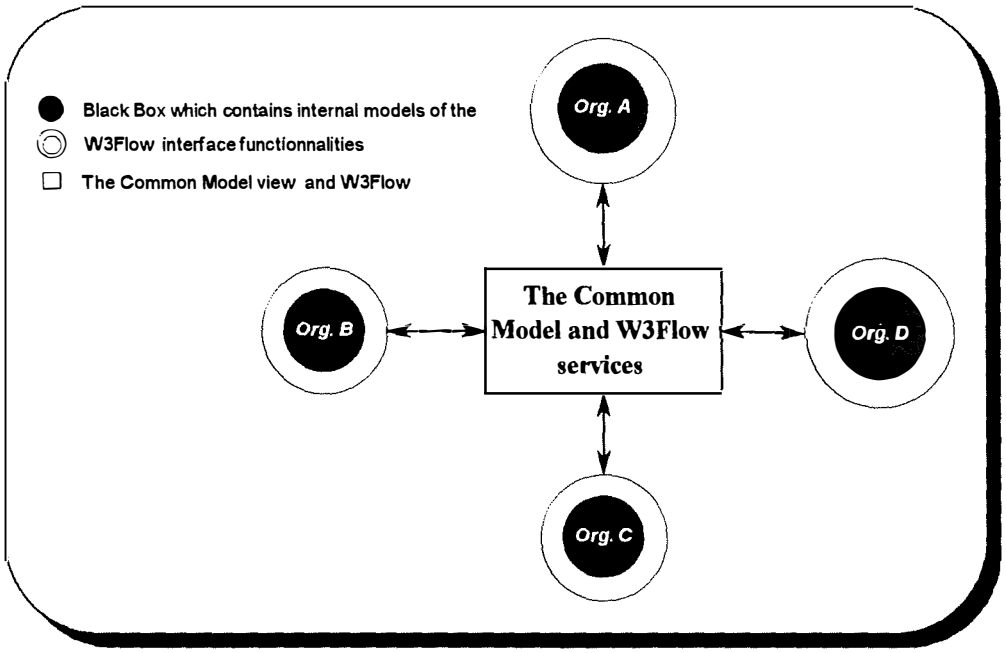


Figure II-2 : Illustration des fonctionnalités d'interfaçage de W3Flow [POOS96, pg.19]

Les fonctionnalités d'interfaçage proposées par W3Flow sur les postes de travail clients des organisations permettent de :

- convertir un format de données en un autre;
- importer les données de documents W3Flow vers des fichiers ou bases de données internes (c'est-à-dire vers les bases de données des organisations);
- exporter les données depuis les fichiers ou bases de données internes des organisations en documents W3Flow;
- appeler des applications internes à partir de W3Flow;
- appeler les services W3Flow à partir des applications internes des organisations.

II.2. Présentation du projet «permis de bâtir»

2.1. Contexte général

Le schéma directeur présente bien la philosophie générale qui guide l'évolution désirée des S.I. (systèmes d'information) pour la région wallonne; en voici un court extrait significatif [SDW, pg. 40]:

« *Le devenir de Namur comme centre administratif principal de la Wallonie constitue un contexte privilégié de mise en oeuvre des TI&C en vue d'améliorer l'efficacité des échanges inter-administrations et la qualité des services rendus aux citoyens. Ce devenir progressif représente une opportunité importante qu'il serait regrettable de ne pas saisir. Ainsi, la création d'un réseau interactif entre les différentes entités qui participent à la gestion d'un dossier devrait notamment permettre :*

- *d'accélérer les échanges entre ces entités;*
- *de mettre en place des dispositifs efficaces d'interaction et de dialogue;*
- *d'améliorer le suivi.*

[...]

Nous proposons comme projet-pilote la gestion des dossiers de permis de bâtir pour deux raisons essentielles : son importance au plan de la vie économique, dans le Namurois en particulier et son caractère extrêmement démonstratif quand à l'impact des TI&C. »

Pour information, deux autres projets pilotes ont été retenus : "téléformation et services associés", ainsi que "serveur Ville de Namur-Citoyens". Ces projets utilisent les mêmes technologies de technologies de l'information et de la communication (TI&C), mais ont d'autres finalités, telle que la promotion de ces TI&C auprès des citoyens et des PME. Toutefois, nous n'allons pas nous attarder sur ces projets, mais bien nous centrer sur la conception d'un système de coopération inter-administrations pouvant servir au projet «permis de bâtir».

2.2. Identification du projet «permis de bâtir»

Pour construire ou apporter des modifications à un immeuble, nécessaire par exemple pour l'implantation ou l'expansion d'une entreprise, il est nécessaire d'obtenir un permis de bâtir. L'attribution ou la non-attribution de ce permis est le fruit d'un processus décisionnel que nous nommerons dans la suite « (procédure de) permis de bâtir ». Cependant, nous y assimilerons deux procédures administratives semblables : le permis de lotir, et le permis d'urbanisme.

La procédure de permis de bâtir implique divers acteurs (partenaires) :

- *les architectes*, qui représentent leurs clients, et introduisent des demandes de permis de bâtir auprès de la ville (ex: de Namur), sous forme de dossier-papier;
- *la ville de Namur*, qui traite les demandes de permis; divers services internes coopèrent et, selon les cas, la commune peut ou doit faire appel à des acteurs externes (région, province,...) pour prendre sa décision d'octroi ou de refus de permis de bâtir;
- *la région wallonne* et *la Province (de Namur)*, qui interviennent selon les cas pour rendre un avis à la ville.

L'enjeu de ce projet est :

- d'*accélérer et automatiser les échanges* entre les partenaires : architectes, ville, province, région wallonne,...
- d'*améliorer le suivi*, pour connaître le stade courant d'accomplissement de chaque procédure;
- de *favoriser l'interaction et le dialogue* entre les différents partenaires.

II.3. Les modèles W3Flow et leur illustration au permis de bâtir

Cette section présente les modèles génériques de W3Flow sous forme de schémas Entité/Association [BOD89]. Cette description des modèles présentée dans [POOS96, pg. 11-14] «n'est pas une spécification complète, mais bien une présentation intuitive des concepts à la base de W3Flow»[4]: les informations, les acteurs/partenaires, les conversations, le temps, et l'exécution des tâches.

Nous allons cependant préciser ces modèles en définissant les concepts utilisés, et ensuite appliquer ces modèles à notre cas particulier «permis de bâtir». Nous donnerons le nom de «PBFlow» à W3Flow appliqué au cas permis de bâtir (W3Flow/PBIA).

Les modèles proposés par W3Flow sont les suivants:

- un modèle inter-organisationnel, définissant «QUI» (II.3.1);
- des modèles informationnel et conversationnel définissant «QUOI» et «COMMENT» (II.3.2. et II.3.3.);
- un modèle temporel définissant «QUAND» (II.3.4.);
- un modèle d'exécution pour exécuter les flux de travail inter-organisationnels (II.3.5.).

II.3.1. Modèle inter-organisationnel

Ce premier modélise l'environnement inter-organisationnel au niveau des ressources humaines. La figure suivante représente sous forme de schéma E/A le modèle inter-organisationnel de W3Flow.

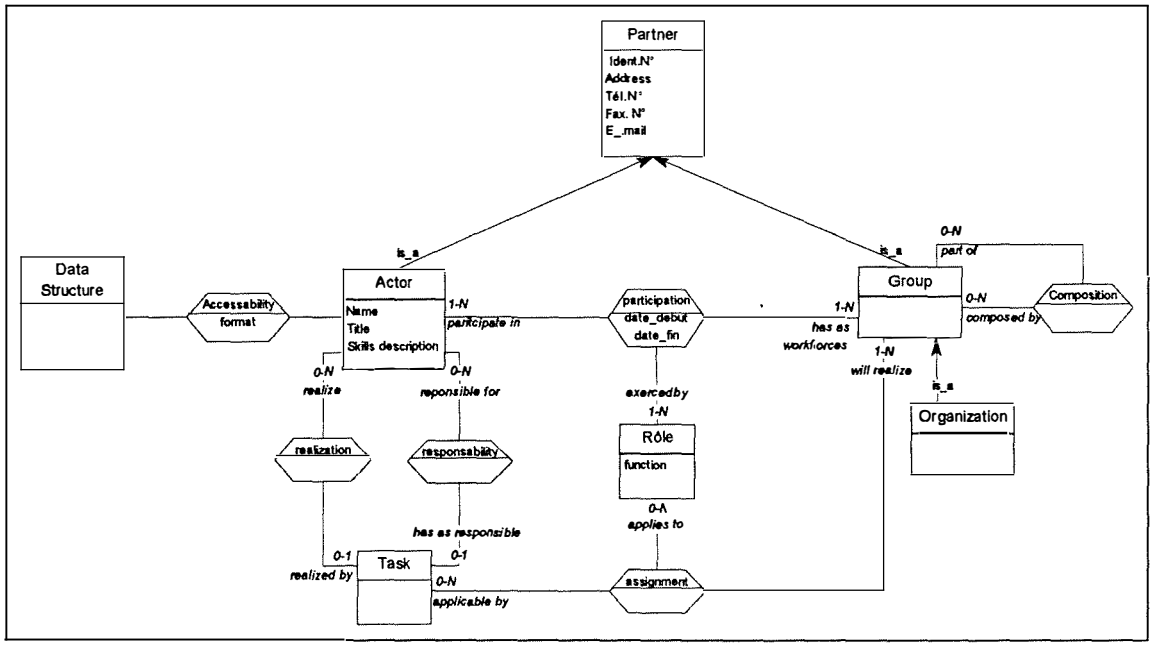


Figure II-3 : le modèle inter-organisationnel W3Flow [POOS96, pg.11]

[4] les modèles présenté ne sont pas commentés dans le document original

Les éléments remarquables de ce modèle sont les suivants[5]:

- un **partenaire** est un individu, ou ensemble d'individus formant un groupe;
- un **groupe** est un regroupement logique d'individus ou de groupes d'individus. Un type particulier de groupe est l'**organisation**, mais dans le contexte plus général de W3Flow, un groupe peut être formé d'acteurs appartenant à des organisations différentes, tout aussi bien qu'une organisation peut être composée de différents groupes.
- un **acteur** est une personne physique, appartenant au moins à un groupe (organisation). Un acteur assume aussi toujours au moins un rôle dans chaque groupe dont il fait partie. En outre, un acteur peut réaliser ou être responsable d'un certains ensemble de tâches.
- un **rôle** est l'expression d'une fonction au sein d'un groupe donné. Cette fonction est assumée par un ou plusieurs acteurs. Par exemple: secrétaires, directeur, etc. Son intérêt est de permettre la répartition dynamique du travail au sein des groupes.
- une **tâche** est un travail qui peut être réalisé avec ou sans la responsabilité d'un acteur; une tâche peut également être assignée à un ou plusieurs rôles de groupes donnés.

Le «**data struture**» modélise lui davantage un poste de travail qu'un acteur : ce sont les différentes structures de données qu'un poste de travail peut comprendre. Même si on peut faire l'hypothèse qu'à tout acteur correspond un et un seul poste de travail, nous préférons dans PBFlow ne pas mélanger ces deux concepts, et modéliser le poste de travail séparément.

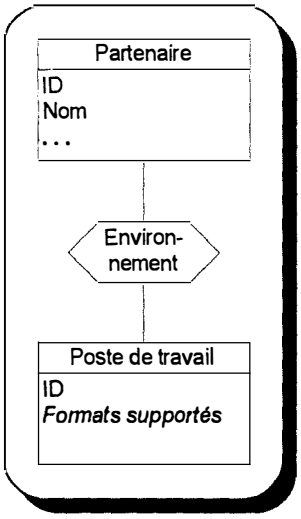


Figure II-4: Relation entre acteur et poste de travail

Illustrons les éléments de ce modèle au permis de bâtir par quelques exemples.

- Les **partenaires** peuvent être une organisation, comme la région wallonne, un groupe, comme le service technique de l'urbanisme de la ville de Namur, ou un individu, comme Mr. Pierrard, agent à la ville d'Andenne.

[5] les descriptions sont nôtres, le document W3Flow à notre disposition ne donnant que les schémas E/A.

- Un **acteur** est une personne physique, telle que Mr. Pierrard, assumant un certain **rôle**, chef de bureau, dans le **groupe** service administratif de l'urbanisme. Ce dernier groupe fait partie de l'**organisation** administration de la ville d'Andenne.
- Les principales **types d'organisations** impliquées dans le projet permis de bâtir sont :
    - les architectes et bureaux d'architectes (demandeurs);
    - les administrations communales, dont chacune est décomposée en 3 **groupes**:
        - le service administratif de l'urbanisme;
        - le service technique de l'urbanisme;
        - le service de l'infrastructure;
        - le collège des bourgmestre et échevins (rendant la décision finale);
    - la région wallonne, rendant son avis aux communes;
    - des intervenants externes, commissions et experts rendant des avis sur certains points précis du projet (incidence au niveau de la pollution sonore, etc.);

    Une liste de ces types de groupes est présentée en annexe («Transmis»). Le nombre total d'organisations et de groupes est donc très élevé, car il y a beaucoup d'architectes, et beaucoup de communes.
- Un **rôle** est une fonction assumée par un acteur (physique), comme secrétaire, chef de bureau, bourgmestre, etc.
- Une **tâche** réalisée et/ou sous la responsabilité d'un acteur; par exemple, à l'administration communale, il y a l'enregistrement d'une demande de permis de bâtir, la localisation et la recherche d'informations géographiques concernant le site du projet, l'élaboration d'un avis thématique, la rédaction du rapport technique par le service d'infrastructure, etc.
- Ces **tâches** seront de préférence assignées aux personnes assumant un certain **rôle** dans l'organisation; par exemple, la décision d'octroi reviendra au bourgmestre, avec l'appui des échevins.

II.3.2. Modèle informationnel

Le modèle informationnel W3Flow décrit les différents types d'informations qui sont les entrées ou sorties de tâches, ainsi que leurs relations avec les partenaires. Voici un schéma E/A illustrant ce modèle.

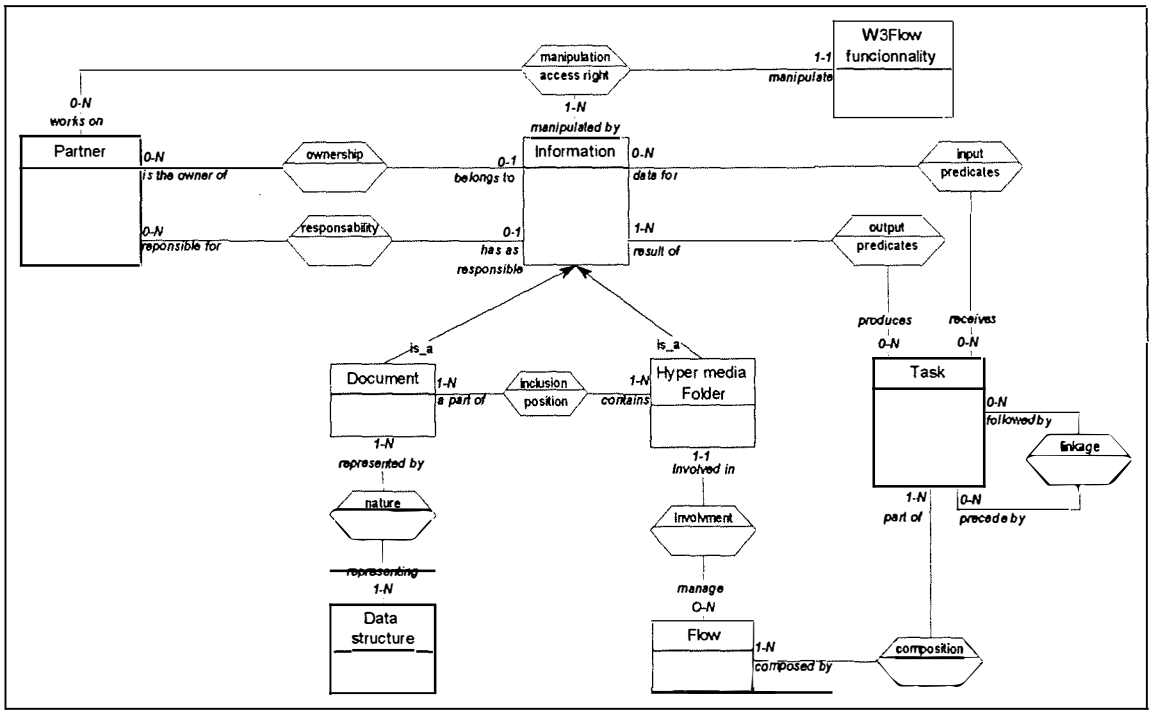


Figure II-5 : le modèle informationnel W3Flow [POOS96, pg. 12]

Une première constatation est que ce modèle présente certains concepts déjà évoqués dans le modèle inter-organisationnel, mais vus ici sous un autre aspect.

Ainsi, une **tâche** est vue dans le modèle informationnel comme une boîte noire : cette boîte noire est le lieu de transformation d'informations d'un certain type (informations en entrée), en autres informations (informations en sortie). Le modèle inter-organisationnel, lui, définissait les assignations, réalisations et responsabilités des exécutions des tâches.

Un **partenaire** est un acteur (individu) ou groupe d'acteurs pouvant être propriétaire ou responsable d'informations, et pouvant manipuler les informations sur lesquelles il travaille. Notons que W3Flow parle de partenaire et non d'acteurs : il suppose qu'une tâche est exécutée et/ou sous la responsabilité d'une personne physique, mais que le résultat produit par cette tâche peut appartenir à un groupe d'acteurs.

Illustrons maintenant ces différents éléments du modèle informationnel W3Flow dans le cas particulier du permis de bâtir:

- Un **partenaire**, par exemple Mr. Pierrard ou le service d'infrastructure de la ville de Namur, peut être propriétaire ou responsable d'une information, et la manipuler dans le contexte d'une fonctionnalité de W3Flow (ex.: gestion de documents).
- Une **information** est un document ou ensemble de documents produit par une tâche, et éventuellement entrée d'une autre tâche. Un exemple d'information (document) est le règlement d'ordre intérieur d'un camping annexe 1, ou une autorisation de dépôt de gaz, ou encore un formulaire d'affiche pour une enquête publique concernant un permis de lotir.

- Un **hyper media folder** est un dossier avec son flux associé, par exemple le dossier de permis de bâtir N°123 de Mr. Dupond.
- Une **tâche** est par exemple la décision d'octroi d'une prolongation d'un permis de bâtir; cette tâche fournira soit un document de refus d'un permis de bâtir, soit un document de prolongation d'un permis de bâtir.

Le concept d'information, présenté dans la figure II-5, reste assez vague : une information est soit un document, soit un ensemble de documents. Nous proposons les définitions suivantes pour définir les différents concepts informationnels...

Information

Nous définissons le concept d'information à partir du modèle de la linguistique [Saussure, cité dans Lesuisse96, § 3.1.1.]. Toute information peut être définie par l'ensemble de ses composantes, c'est-à-dire :

- son *signifiant*, ou forme, qui est l'ensemble de signes représentant l'information;
- son *signifié*, ou fond, qui est le contenu de l'information (telle qu'une idée);
- sa *syntaxe* (ou *grammaire*), qui est l'ensemble de règles permettant de faire correspondre à tout signifiant un signifié;
- son *référentiel*, ou contexte, qui est une référence vers le contexte où cette information prend tout son sens (ex.: une référence à l'instance du flux de travail particulier «permis de bâtir B122 de Mr X»).

Nous utilisons le terme «information (au sens large)» pour désigner aussi bien une information non-décomposable, qu'un ensemble d'informations (document, formulaire,...). Par «information (au sens strict)» ou «unité informationnelle» nous désignons une information contenue dans un document, message, ou formulaire, mais non le document lui-même.

Message

Un message est une information ou un ensemble d'informations échangé(e) entre deux postes de travail.

Document

Un document est un regroupement d'informations logiquement corrélés, ayant une signification sur le plan administratif.

Formulaire

Un formulaire est un document ayant une structure fixe, et pouvant donc être prédéfini.

Multimédia

Un document multimédia est un document contenant des informations de natures différentes (texte, son, image,...). Il en va de même pour un formulaire multimédia.

Dossier

Un dossier est un ensemble de documents relatifs à une même instance flux de travail (par exemple, le dossier relatif à la procédure d'attribution de permis n°123 de Mr. X). A tout dossier correspond donc une seule instance de flux de travail, et vice-versa. W3Flow utilise lui dans son modèle le terme de *«Hyper media Folder»*.

Data structure

La structure de données d'un document (ou formulaire) présentée dans W3Flow mérite toute notre attention, car c'est elle qui sera la pierre d'achoppement des fonctionnalités d'interfaçage. La notion de data structure, correspondant aux concepts de signifiant et de syntaxe/grammaire dans le modèle de la linguistique, est insuffisante pour modéliser des types d'informations. Pourquoi? Parce qu'elle poursuit un double objectif : premièrement la possibilité de lire document, et deuxièmement l'interfaçage entre systèmes de workflow (inter-organisationnel et intra-organisationnels).

C'est pourquoi nous introduisons en lieu et place de «data structure» les notions de structures externe et interne d'un document.

Structure externe (format ou support)

La structure externe d'un document décrit sa nature (texte, image, son,...) ainsi que la manière de le lire. La connaissance de la structure externe d'un document permet de s'assurer qu'on pourra techniquement lire ce document.

Structure interne

La structure interne d'un document ou formulaire décrit la manière dont sont structurées les informations qu'il contient. La connaissance de la structure interne d'un document permet de s'assurer qu'on pourra techniquement traiter et exploiter les informations. Cette exploitation nécessite la possibilité d'identifier les informations au sens strict, c'est-à-dire de pouvoir décomposer un document ou formulaire en informations plus détaillées (ex.: nom, prénom, adresse,...).

« Techniquement »

Le terme «techniquement» que nous utilisons dans les définitions de structures externe et interne indique qu'il s'agit seulement d'une possibilité technique. Par exemple, il se peut qu'un acteur ne puisse pas lire ou modifier un document non pas parce qu'il est illisible, mais parce que cet acteur n'a pas les droits d'accès suffisants pour lire ou modifier le document.

Exemple: la carte d'identité informatisée. C'est un formulaire multimédia, contenant texte et photo. Si nous pouvons visualiser une carte d'identité sur notre écran, nous pourrons dire que notre poste de travail en connaît sa structure externe. Mais pour pouvoir alimenter automatiquement notre base de données avec les nom, prénoms, adresses, etc. de chaque carte d'identité, il sera nécessaire de connaître sa structure interne.

La connaissance de la structure interne des documents est donc, dans notre cas, capitale car c'est elle qui va permettre l'interfaçage entre les données externes reçues de notre système PBFlow, et les données internes utilisées par chaque acteur. *Plus précisément, la structure interne définit la syntaxe ou grammaire qui permettra de décomposer un document ou formulaire en un ensemble de paires signifiant/signifié.*

Un poste de travail sera également défini par les structures de données qu'il est capable de manipuler.

Le modèle informationnel de W3Flow est certainement intéressant, mais nous ne pouvons pas l'appliquer tel quel dans notre système PBFlow. Pour pouvoir spécifier un flux d'informations, un dossier, une tâche ou un document, nous devons pouvoir distinguer les types et instances, ce que ce modèle ne fait pas [voir chapitre 1 pour la distinction type et instance]. C'est pourquoi nous proposons le modèle E/A que voici à la figure II-6 suivante.

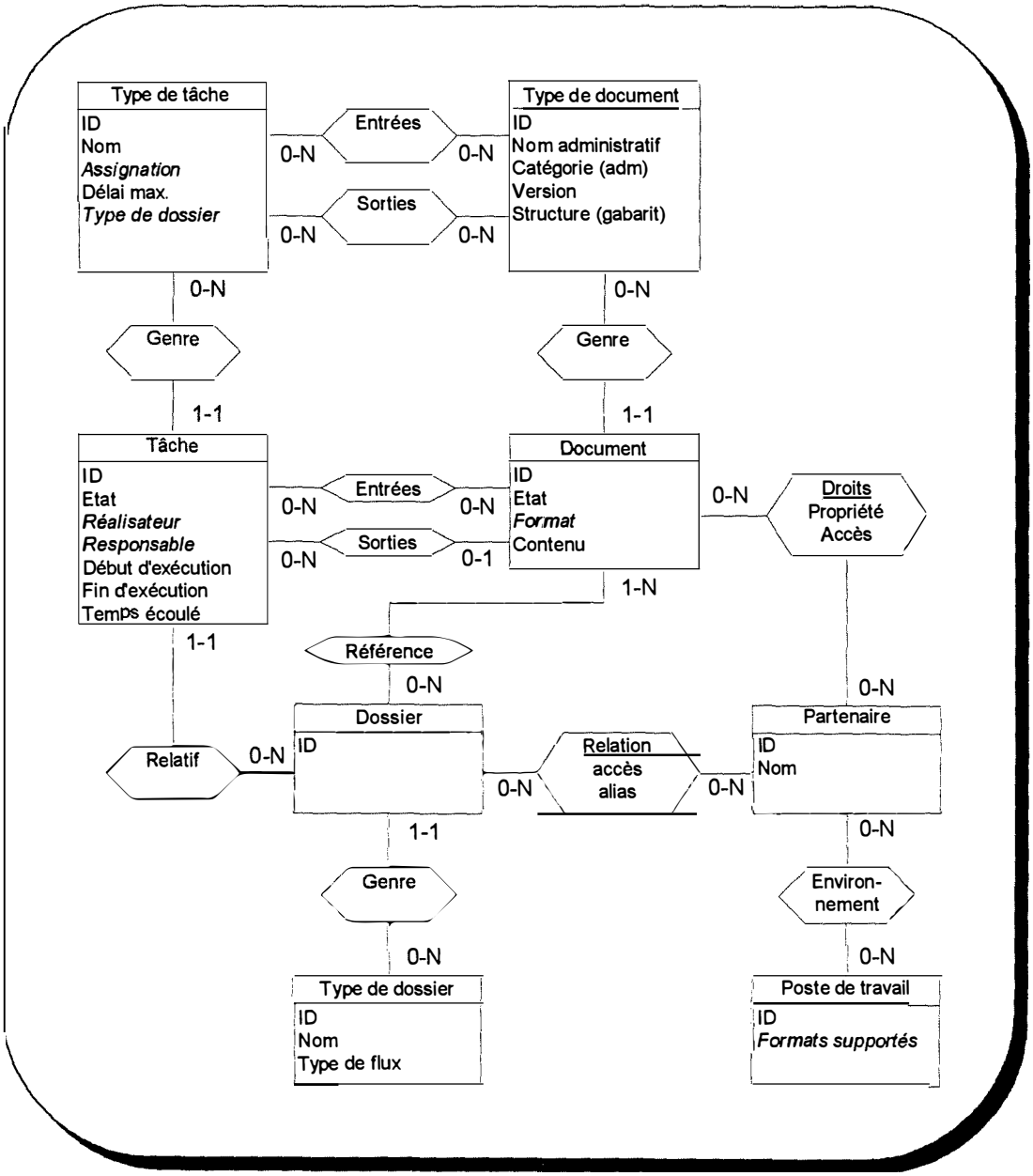


Figure II-6: Schéma E/A (partiel) de PBFlow

L'*alias* d'un dossier permettra à un partenaire d'appeler un dossier avec un nom évocateur pour lui (ex.: nom du client pour un architecte, nom de l'architecte pour une autre administration, etc.).

Ce qu'il est important de remarquer, c'est **l'unicité des instances** de tâches et documents **pour un même flux de travail**. Cela signifie que, dans le cadre d'un dossier donné, il n'y aura jamais simultanément qu'un seul document d'un type déterminé, et qu'une seule instance de tâche d'un type donné. Le contraire indiquerait une situation fort ambiguë.

Par exemple, il n'y aura jamais à un moment donné qu'un seul document «tableau de relevé du cadastre» dans le cadre du traitement «demande de permis de bâtir n°123 de

M. Dupond». Au cas où un document devrait être modifié, nous ne considérerons jamais que la dernière version de ce document, les autres - périmées - étant détruites ou archivées.

Cette hypothèse est essentielle, car elle nous permettra -dans la gestion d'un flux de travail- de nommer les documents simplement par le nom administratif de leur type de document (et de même pour les tâches).

II.3.3. Modèle conversationnel

Le modèle conversationnel représente les échanges de messages entre acteurs pour l'accomplissement de leur tâches ou la recherche d'informations. La figure II-7 représente sous forme de schéma E/A ce modèle.

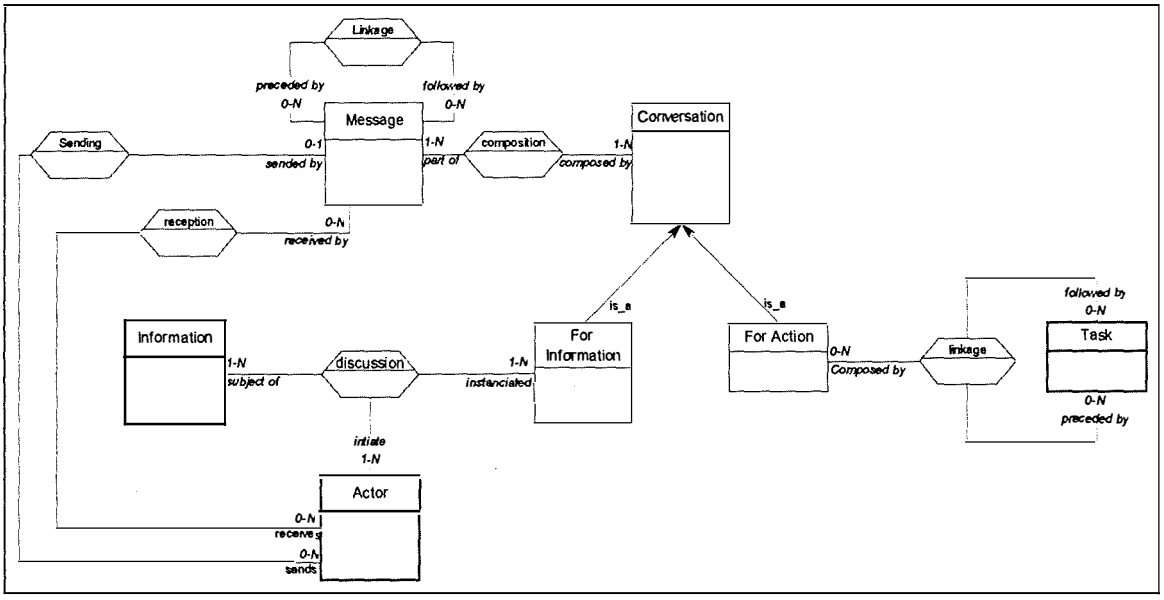


Figure II-7: Le modèle conversationnel W3Flow [POOS96, pg.13]

Une **conversation** est un échange de messages entre acteurs, relatif à une même discussion à propos d'une information, ou à propos d'un enchaînement de tâches.

Un exemple de conversation serait un échange de messages entre Mr. Jacquet, fonctionnaire au service d'administration technique de la ville de Namur, avec Mr. Dufour, délégué chez Belgacom. Cette conversation pourrait concerner une information, comme les conditions d'octroi de pose de câbles. Cette conversation pourrait également concerner une tâche : par exemple, Mr. Jacquet désirerait que le service de Belgacom étudie l'implantation de la pose de câbles sur un site particulier.

Dans W3Flow, le mécanisme de conversation est vu comme le moyen (métaphorique) de représenter les interactions entre les partenaires, comme l'illustre l'exemple de la figure suivante [figure II-8].

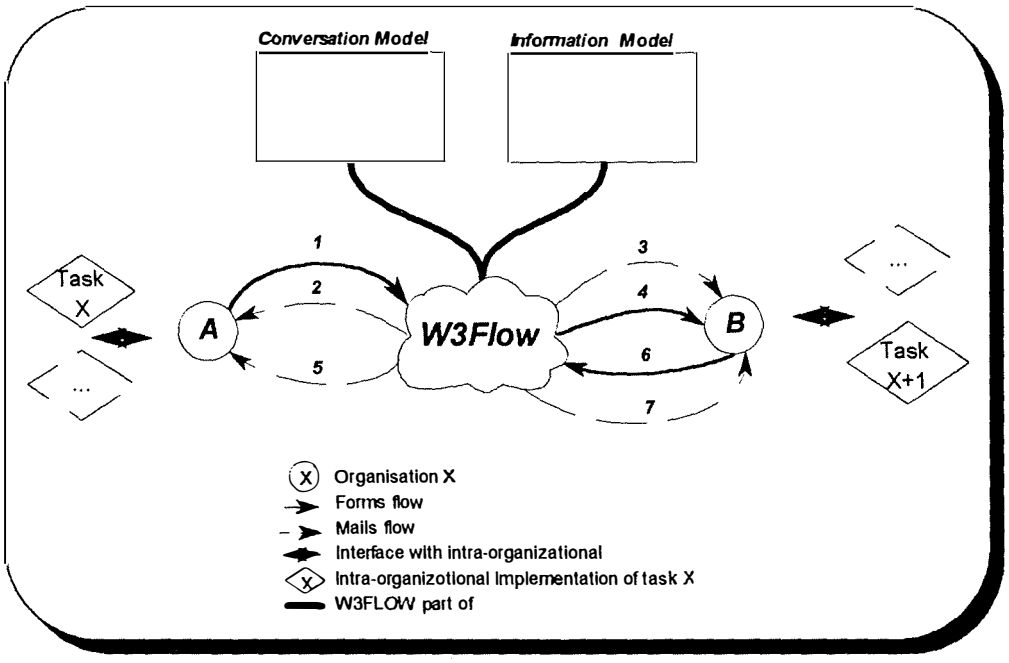


Figure II-8: La métaphore de la conversation, dans W3Flow [POOS96, pg.10]

La figure II-8 pourrait se lire comme suit :

1. Un acteur A, ayant exécuté (ou fait exécuté) la tâche X qui lui était assignée, renvoie le résultat de son travail - un formulaire - au système W3Flow.
2. Lorsque W3Flow reçoit ce formulaire, il envoie un accusé de réception à l'acteur A.
3. Le système W3Flow utilise alors la description de la conversation (depuis son modèle conversationnel), pour voir quelle tâche doit être exécutée après la tâche X. C'est la tâche X+1, et il envoie donc un message à l'acteur B, responsable de la réalisation de la tâche X+1.
4. L'acteur B lit son message, et se connecte à W3Flow. Le système W3Flow lui envoie alors un formulaire contenant les entrées informationnelles de la tâche X+1. L'acteur B peut donc commencer à exécuter la tâche X+1.
5. W3Flow envoie un message à l'acteur A pour lui indiquer que B a reçu les documents.
6. Lorsque l'acteur B a terminé son travail, il l'envoie au système W3Flow (cette étape est similaire à l'étape 1).
7. Lorsque W3Flow reçoit ce formulaire, il envoie un accusé de réception à l'acteur B, et ainsi de suite...

Comme l'illustre cet exemple, les entrées et sorties de tâches sont des documents, et plus précisément des formulaires, à la structure mieux définie. Les messages sont eux relatifs au séquencement des tâches (accusé de réception, avertissement de l'arrivée d'un nouveau document, etc.). Ces messages forment ainsi des «conversations».

II.3.4. Modèle temporel

Le modèle temporel de W3Flow fournit un moyen de représenter le temps, illustré sur la figure II-9.

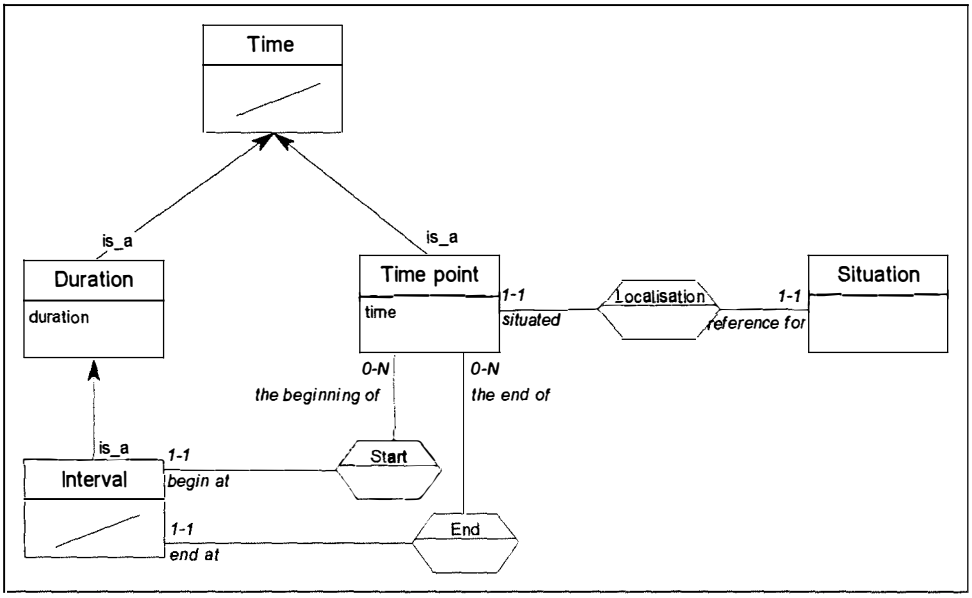


Figure II-9 : le modèle temporel utilisé dans W3Flow [POOS96, pg.13]

W3Flow utilise, pour représenter le temps, des unités abstraites appelées «time». Il l'utilise comme base essentielle pour coordonner les actions relatives à des processus inter-organisationnels.

Le temps est aussi utile pour représenter les deadlines, les moments d'achèvement et de commencement des tâches, etc.

Pour notre part, nous préférerons utiliser dans PBFlow l'échelle de temps habituelle (date et heure). Néanmoins, il n'est pas utile de détailler le temps jusqu'au niveau des minutes et des secondes : notre système PBFlow concerne en effet des tâches administratives comme le permis de bâtir, et non des opérations boursières.

II.3.5. Modèle d'exécution

Ce dernier modèle de W3Flow ne ressemble pas aux autres : il montre la vue qu'on les acteurs sur les tâches à réaliser (to-do list) et sur les dossiers (folders).

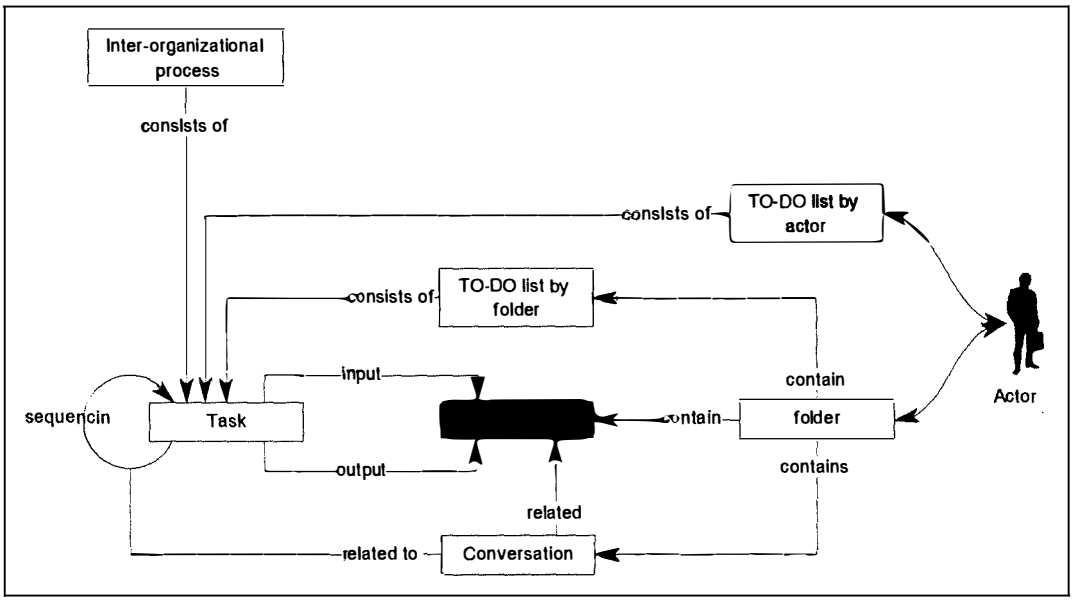


Figure II-10 : Le modèle d'exécution de W3Flow.

Outre des éléments utiles pour la gestion du travail (to-do-list par dossier et par acteur), nous voyons aussi ce modèle comme une intégration des modèles informationnels et conversationnels.

Ce modèle d'exécution exprime également mieux la notion de dossier (*folder*) : un dossier contient non seulement des documents relatifs à un même flux de travail, mais également une représentation de ce flux de travail (conversations relatives au séquencement des tâches).

II.4. Architecture technique de PBFlow

Notre but ultime est de concevoir un système de workflow inter-organisationnel, centré sur les documents et les tâches (document centric), et assez flexible (object oriented), permettant de gérer des procédures administratives telles que la procédure d'attribution de permis de bâtir. Notre projet s'insère donc dans celui de W3Flow.

L'architecture du système que nous proposons est légèrement différent de celle de W3Flow : nous prendrons comme hypothèse que les acteurs qui entreront en communication avec le système PBFlow n'utiliseront qu'un navigateur Web (browser www). Les fonctionnalités comme l'interfaçage seront elles assurées au niveau du serveur PBFlow.

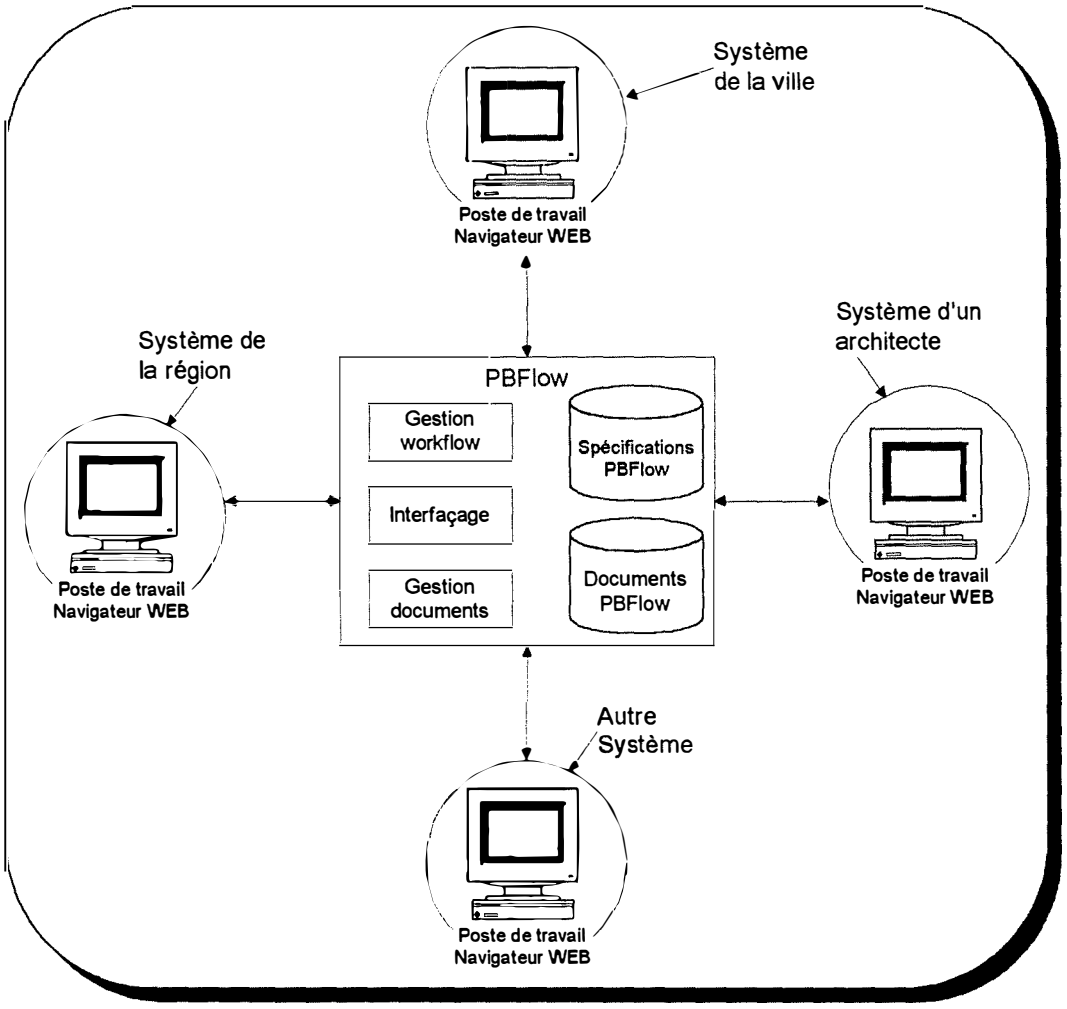


Figure II-11 : Architecture technique du système PBFlow

Les seules connaissances que nous aurons des systèmes intra-organisationnels sont :

- la description des postes de travail qui interagissent avec PBFlow;
- la description des organisations (acteurs, groupes, rôles);
- la spécification des tâches prises en compte dans le contexte inter-organisationnel, et en particulier les Entrées/Sorties de ces tâches.

Nous considérons donc toute organisation comme une boîte noire : nous ne pouvons pas intervenir sur les bases de données des organisations, qui restent leur propriété.

La gestion de la coopération sera effectuée par un serveur PBFlow; il s'agit donc d'une architecture assez centralisée (semblable à W3Flow).

La technologie que nous privilégions est celle d'Internet. Notre seule contrainte pour que les acteurs puissent utiliser notre système est qu'ils disposent d'un navigateur web avec ses fonctions intégrées (e-mail, etc.).

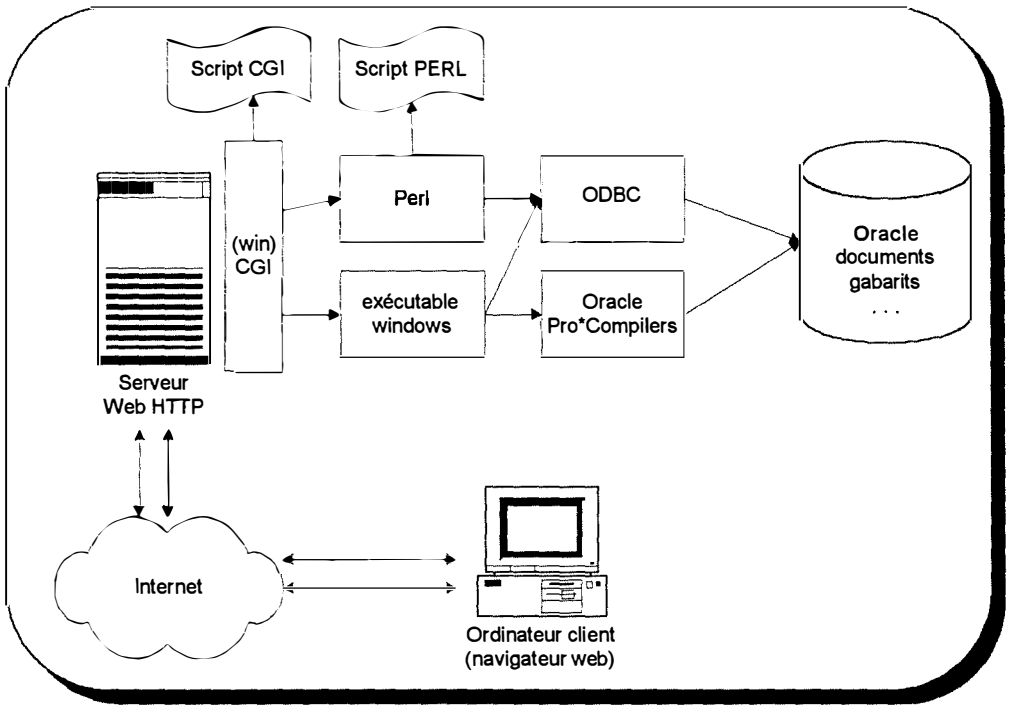


Figure II-12 : Architecture technique d'un serveur PBFlow

La figure II-12 illustre l'interaction entre les clients www et le serveur Web de PBFlow. Ce serveur va interagir, via la couche CGI *(Common Gateway Interface)* avec des applications écrites en PERL ou tout autre langage. Ces applications réaliseront les différentes fonctionnalités de notre système : gestion des documents, gestion du flux, interfaçage, etc. Elles utiliseront la couche ODBC de ms-windows pour accéder à la base de données qui contiendra les informations propres au système, ainsi que les différents dossiers et documents. Le lecteur davantage intéressé par ces aspects technique pourra se reporter à notre travail [SYR96].

Nous allons concentrer notre étude sur un sous-ensemble de fonctionnalités de W3Flow : celles qui concernent la gestion des documents, la gestion du flux de travail, et l'interfaçage entre les systèmes intra- et inter-organisationnels.

Nous aborderons donc la spécification :

- des types de documents, et ainsi des E/S des types de tâches;
- des postes de travail;
- de l'interfaçage;
- des flux de travail, et ainsi les enchaînements de tâches (prédéfinis et ad-hoc);
- des dossiers;
- du temps, notamment au niveau de l'exécution des tâches.

Les autres points - tels que la gestion des conversations - ne seront pas examinés prioritairement, tout d'abord parce que la littérature et les outils existants y consacrent une large part[6], mais ensuite aussi pour répondre aux spécificités de notre système de workflow «permis de bâtir» inter-organisationnel et centré sur les documents.

[6] voir notamment nos travaux dans le cadre du « The Milano System »

II.5. Comparaison entre W3Flow et d'autres systèmes.

Le but de cette section n'est pas de comparer de manière exhaustive un système de workflow à l'état de projet en cours (W3Flow/PBFlow) avec les autres systèmes existants. Nous allons ici exposer brièvement les fonctionnalités de Lotus Notes (II.5.1.) et du The Milano System (II.5.2.), pour ensuite les comparer avec celle de notre système de workflow.

II.5.1. Comparaison avec Lotus Notes

Lotus Notes est connu pour être le groupware par excellence, et donc «le» logiciel pour gérer les situations de collaboration au sein d'un groupe [CMC n°60, pg.48].

Lotus Notes, ce sont tout d'abord des *bases de données de documents*. Chaque document est vu comme un container regroupant les informations les plus diverses: des textes, des feuilles de travail, des illustrations, du son et même de la vidéo. Les caractéristiques de mise-en-page du document-source sont également conservés. Le format de données utilisé est propre à Notes, mais des encapsulations d'objets via OLE[7] sont possibles.

Mais Notes se présente plutôt comme un environnement de développement, mettant en oeuvre des "forms", "views", "navigators" et "agents". Les *forms* de Notes sont des formulaires se présentant comme des écrans d'entrées de données (chaque champ d'un formulaire fait référence à un champ d'une base de données). Les *views* déterminent la manière dont les informations stockées seront présentées. Les *navigators* sont eux des aides à la réalisation de tâches, et les *agents* enfin interviennent pour automatiser des tâches.

Les fonctionnalités de Notes restent essentiellement *mail & document-centric*, et axées sur la collaboration au sein d'un groupe; ces fonctionnalités sont relatives :

- au courrier électronique et à la gestion de conversations et discussions de groupe;
- aux banques de connaissances, avec moteur de recherche sur base de mots-clés, sujet, auteur, résumé, etc.
- aux documents, avec partages de l'information et gestion des droits d'accès;
- au contrôle de projets;
- à la gestion d'agenda partagé, avec un gestionnaire de rendez-vous;
- ...

Une de ces fonctionnalités dans le cadre du partage d'informations est la réplication. Ce procédé permet de répartir les informations d'un organisme au sein de

[7]OLE = Object Linking and Embedding, de ms-windows [DELPHI, pg.819]

plusieurs établissements : à intervalles réguliers, les différentes bases de données des établissements vont se synchroniser, en envoyant aux autres leurs informations nouvelles ou récemment modifiées. Notes permet donc d'avoir une architecture assez distribuée.

Certaines fonctionnalités proposées par Notes se retrouvent dans notre système PBFlow, notamment celles relatives aux documents multimédias et aux formulaires.

Mais Notes est un système intra-organisationnel, qui ne tient pas compte de la problématique inter-organisationnelle. En particulier, il impose à chaque acteur collaborant à disposer de l'application client spécifique à Lotus Notes, il ne prévoit pas d'interfaçage entre différents systèmes de workflow.

En outre, il met peu l'accent sur la notion de dossier : les documents se retrouvent tous un peu pêle-mêle dans ses bases de documents. Des fonctionnalités comme la recherche de documents sur base de mots-clés prend alors toute son importance. Dans notre contexte par contre, ces fonctionnalités sont moins utiles.

D'un autre côté, Notes offre toute une série de fonctionnalités inutiles dans un contexte inter-organisationnel, comme la réplication (nécessaire du fait de son architecture fortement distribuée). D'autres fonctionnalités de Notes sont également inutile dans notre contexte de collaboration : la gestion de rendez-vous et d'agendas partagés sont peu utiles dans le cadre de procédures administratives inter-organisationnelles.

II.5.2. Comparaison avec The Milano System

Le système Milano, *"The Milano System"* (ou TMS) est un système de collaboration (groupware) [AGOS95, TAES96]. Il est composé de trois grandes composantes formant un tout :

1. the Organization Handbook (MOH);
2. the conversation Handler (MCH);
3. the Workflow Managment System (MWMS).

Le MOH est le reflet de l'organisation et présente non seulement la hiérarchie des objets de base de l'organisation, mais aussi la répartition de ses compétences. Aussi, il permet aux acteurs de naviguer, de retrouver, et localiser l'information qu'ils recherchent, et quels acteur(s) la détermine(nt).

Le MCH offre des fonctionnalités de gestion de communications interpersonnelles, et est actuellement proposé pour des conversations via e-mail (courrier électronique). Le MCH serait cependant généralisable à d'autres technologies, telles que les vidéoconférences.

Le MWMS gère l'enchaînement des tâches, leur coordination, ainsi que la circulation des documents. Le système workflow (flux de travail) MWMS est basé sur la technologie e-mail.

The Milano System est un donc système de workflow *mail-centric*, qui utilise le courrier électronique comme middleware ou "enabling-technologie". La raison de ce choix est que le courrier électronique offre des standards comme MIME[8] qui peuvent apporter une solution entre autres au problème d'hétérogénéité des environnements.

Par rapport à PBFlow, The Milano System est un système coopératif qui s'applique à des situations aussi bien intra- qu'inter-organisationnelles. Cependant, il se restreint à des situations fortement prédéfinies, et relègue la gestion des documents au second plan. C'est essentiellement un système de workflow mail-centric : il met l'accent sur la gestion des conversations et du flux de travail (prédéfini). Des extensions vers des situations moins prédéfinies ainsi qu'une gestion des documents sont toutefois à l'étude.

[8] Multipurpose Internet Mail Extensions [RFC 1521-1522]

Chapitre III - Spécification et gestion des documents

Dans ce chapitre, nous allons spécifier les documents et type de documents. Ceci nous amènera à voir comment gérer les documents, et en particulier le problème de l'interfaçage entre postes de travail des différents partenaires. La figure III-1 ci-dessous illustre ce que nous allons spécifier.

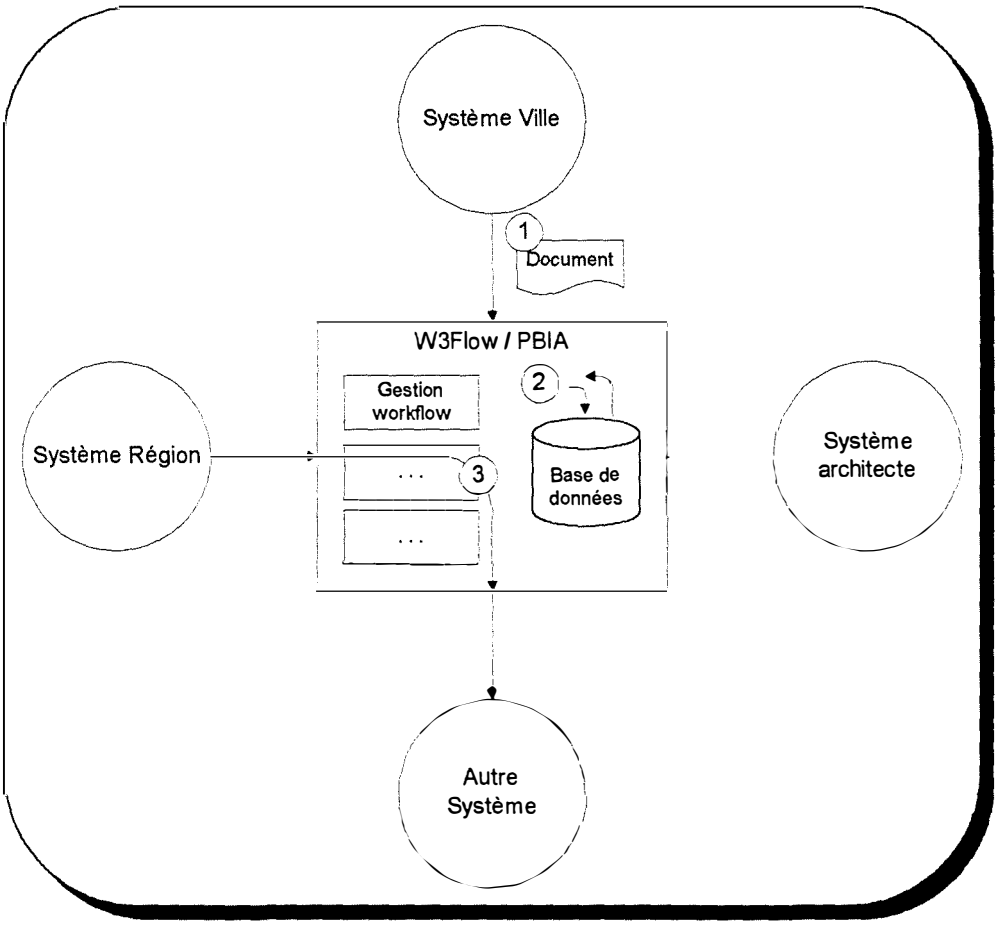


Figure III-1 : PBFLow et les documents

Nous allons spécifier :

1. les types de documents et documents échangés entre les acteurs (III. 1);
2. la manière dont ces documents seront stockés dans notre système PBFlow (III.2);
3. comment sera réalisé l'interfaçage entre les différents systèmes des acteurs (III.3).

III.1. Spécification des types de documents

III.1.1. Que spécifier pour un type de document

Rappelons ce que nous avions exposé dans le chapitre 2 à propos de la spécification des types de documents et des documents[9] :

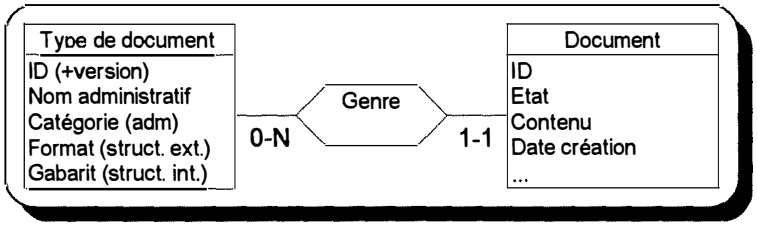


Figure III-2 : Relation entre type de document et document

Pour spécifier un type de document, nous devons définir :

1. ses *caractéristiques générales* :
   - *Identifiant*;
   - *Nom administratif* (ex.: «CWATUP Annexe 21»);
   - *Description* (facultatif);
   - *Catégorie*, parmi : «permis de bâtir», «permis de lotir», «permis d'urbanisme». Notons que certaines administrations utilisent les termes « *type* de formulaire » pour *«catégorie»*.
2. sa *structure externe*, ou *format*.
3. sa *structure (interne)*, essentielle lorsqu'il s'agit d'un formulaire.

Pour spécifier un document, nous devons définir :

1. ses *caractéristiques générales* :
   - *Identifiant*;
   - *Etat* (créé, en cours, archivé,...);
2. son *type* (de document);
3. son *contenu*, ayant une structure correspondant à son *type*.

En outre, la spécification des (types de) documents est importante car ils sont en relation avec beaucoup d'autres entités : les entrées/sorties de tâches sont des documents, les entrées/sorties de type de tâches des types de documents, etc. [cf. II.3.]

Nous allons maintenant détailler la manière dont nous représentons les structures externes et internes.

[9] les relations avec les autres types d'entités ne sont pas mentionnées ici [II.3.].

III.1.2. Structure externe d'un type de document

1.2.1 Les formats de fichiers

D'une manière générale, pour pouvoir lire un document informatique (soit généralement un fichier de données), il suffit de savoir quelle application a créé le document qu'on veut lire, et de disposer de cette même application ou d'une fonctionnalité d'importation dans une application équivalente.

Ceci permet de dire qu'on peut spécifier la structure externe d'un fichier-document en désignant l'application qui l'a créée. Sur PC ms-windows, cette indication est donnée par l'extension du fichier, et ms-windows tient une liste d'association entre types de fichiers (extensions) et applications. C'est ce que ms-windows nomme les «File Types».

On serait donc tenté de se dire que ça suffit. Il n'en est rien, car ce serait oublier les documents multimédia : comment les traiter? En outre, l'utilisation d'extensions n'est pas standardisée : un fichier avec l'extension .DOC peut avoir été créé par plusieurs programmes différents et incompatibles entre eux. Pire: certains environnements ignorent même carrément leur utilisation (ex: Macintosh).

C'est pourquoi nous proposons de définir la structure externe d'un document par son en-tête MIME. MIME est l'acronyme de *Multipurpose Internet Mail Extensions* [RFC 1521-1522], un standard au départ destiné à être utilisé pour le courrier électronique e-mail sur Internet, mais de portée plus générale. Ainsi, MIME est utilisé dans le contexte du WWW par les serveurs et clients HTTP. Voici un exemple d'en-tête MIME d'une page HTML:

```
Content-Type: text/html; charset=iso-8859-1; name="annexe21.html"
Content-Transfer-Encoding: 8bit
Content-Disposition:inline; filename="annexe21.htm"
Content-Base: "ftp://nora.info.fundp.ac.be/wwwroot/Pbia/annexe21.htm"
```

1.2.2. MIME

MIME est un standard fort répandu, mais méconnu : beaucoup de gens croient qu'il se limite au «file attach» proposé par des éditeurs de messages e-mail pour Internet. C'est pourquoi nous présentons MIME ici.

A. Mise en appétit

En 1982, le RFC 822 d'Internet a défini un format standard et une structure pour les messages e-mail. Malheureusement, ce format est très limité : les messages ne peuvent

contenir que du texte, et encore... en 7 bits, donc sans accents, avec des lignes de texte et des messages de tailles limitées. Pour des utilisations telles que du transfert de documents entre personnes, ce standard est tout à fait insuffisant. Quant aux en-têtes de messages, ils se limitent pratiquement à "From", "To", "Subject" et "CC".

Le MIME, acronyme de *"Multipurpose Internet Mail Extensions"*, a été défini en 1992, et reprend les fonctionnalités du vieux standard de 1982 sur la structure des messages e-mail. Mais il fait plus : tout d'abord, il standardise des champs additionnels des en-têtes de messages du vieux protocole. Ces champs décrivent de nouveaux types de contenus et définit une structure modulaire aux messages. Mais aussi, MIME substitue la notion «d'entité MIME» à celle de message : une entité MIME peut être un document, ou une partie de document.

C'est pourquoi les applications du standard MIME s'appliquent de plus en plus à d'autres contextes, comme le World-Wide-Web. C'est logique lorsqu'on pense qu'un message n'est rien d'autre qu'un document échangé entre deux postes de travail, ou entre un serveur (ex.: HTTP) et un client (ex.: browser web).

B. Fonctionnalités

Un message MIME peut contenir :

- plusieurs objets emboîtés : une entité MIME peut en contenir plusieurs;
- des textes et lignes de longueurs illimitées;
- l'utilisation de caractères autres que ASCII (1-127), donc l'emploi d'accents, etc.
- plusieurs polices de caractères dans le texte;
- des fichiers spécifiques, binaires ou applications;
- des informations sous forme d'images, de son, de vidéo, et multimédia.

C. Format des documents

Dans un message ou document MIME qui contient un seul objet, on trouve un en-tête ("*header*") qui est un ensemble de champs décrivant le contenu, ainsi que le contenu lui-même, appelé "*body*".

Un message MIME contenant plusieurs objets est un message contenant un seul objet 'tableau'. Ce tableau est défini dans l'en-tête du message, et le corps du tableau est un ensemble d'objets. Chaque objet du tableau est à nouveau un header+body, comme le montre l'exemple de la figure suivante :

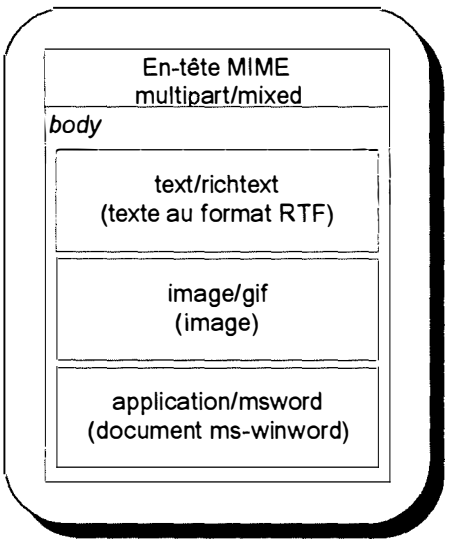


Figure III-3 : Exemple de structure d'un message MIME

Les champs définis l'en-tête ont la structure suivante:

```
Content-Type := type "/" subtype [";" parameter]_
```

Content-Type indique le type de contenu, où «*type*» peut être "Text" (texte), "Audio" (son) "Image" (image), ou "Application" (un fichier spécifique à une application). Le «*subtype*» précise cette définition. Par exemple: "text/ascii", "text/richtext", "text/html",...

Les paramètres facultatifs «*Parameter* » qui suivent dépendent du type et du sous-type, et indiquent certains paramètres nécessaires pour lire ou désigner correctement l'information. Par exemple, un paramètre peut contenir le nom d'un fichier, le nom de la table de caractère utilisée (pour du texte), les librairies AutoCAD utilisées, etc.

Le type "*Application*" signifie que la partie "body" qui suit contient un type de fichiers qui n'est pas défini dans le standard de 1992/93. Cela permet d'associer un type de fichiers à une application. Par exemple, le type "Application/msword" indique que le corps qui suit contient un fichier microsoft winword. Ces (sous-)types sont standardisés par l'INA [RFC 1521-1522].

Si le sous-type commence par "x-" il s'agit d'un sous-type défini bilatéralement par le développeur de l'application et l'utilisateur, sans aucune standardisation. Ex.: Application/x-msword. C'est un peu un problème du MIME, car certains développeurs d'applications utilisent des "x-" différents pour designer les mêmes types de fichiers.

Le type '*Multipart*' indique que le corps (body) qui suit contient en fait plusieurs objets. La séparation entre les objets se fait grâce à l'identification de chaînes de caractères spéciales définies en paramètres. On les appelle "boundary" (boundaries).

Voici un exemple de message MIME :

```
From: Frederic Taes
To: William Poos
Subject: Test message MIME
MIME-Version: 1.0
Content-type: multipart/mixed; boundary="limite1234"

Remarques. Generalement destine a ceux qui n'ont pas de lecteur
MIME. Les lecteurs MIME ignorent les remarques.
--limite1234
Content-type: text/html; charset=iso-8859-1;

<b>Ceci constitue le premier élement de texte du message.</b>
--limite1234
Content-type: text/plain; charset=us-ascii

Et ceci le second, mais ce pourrait etre du son XXencode par exemple,
ou autre (ici: ascii 7 bits). UUencode n'est pas utilise, question
d'eviter l'emploi de caracteres qui servent a limiter les champs MIME
(ex.: -,>,<)
--limite1234

Remarques (epilogue). Egalement ignore par les lecteurs MIME.
```

Figure III-4: Exemple de message MIME

Les fichiers peuvent être codés selon différentes tables. Actuellement, BASE64, QUOTED-PRINTABLE, 8BIT, 7BIT, BINARY et x-EncodingName (défini par le développeur d'applications).

Un autre type très intéressant est le type "*Message/External-Body*" qui indique que le corps (body) n'est pas contenu dans le message lui-même mais dans un fichier externe, c'est-à-dire soit sur disque local, soit sur un site FTP anonymous, ou bien FTP non-anonyme.

Enfin, le sous-type "*Alternative*" (ex.: "multipart/alternative") permet de donner plusieurs représentations d'un même objet. Par exemple, si le lecteur du message dispose de l'application associée à la première partie du message, c'est celle-ci qui sera exécutée. Sinon, ce sera l'application associée à la seconde partie du message qui sera exécutée.

D. Informations techniques

Les spécifications complètes du MIME se trouvent dans les RFC 1521 et 1522. Les RFCs 1341 et 1342 sont obsolètes. Actuellement (1996), la dernière version du MIME est toujours celle de septembre 1993 (v 1.0). Il y a également la FAQ du newsgroup comp.mail.mime. Attention toutefois car ces documents sont énormes point de vue taille.

E. Extensions

MIME ne propose rien de plus qu'exposé ci-dessus (dans les grandes lignes) et, notamment, rien point de vue sécurité (authentification, etc.). Le S/MIME est un working group de l'IETF[10], donc un standard potentiel. Par rapport au MIME, il rajoute à la fois authentification et cryptage (chiffrement). Par rapport à PGP, il rajoute une gestion hiérarchisée des accès, par rôle et selon les utilisateurs, et une meilleure intégration au courrier électronique (de type MIME).

Quant au RFC 1651 de 1994 sur les "SMTP Service Extensions" nomme explicitement le MIME, sans se limiter à lui.

1.2.3. Pourquoi MIME nous convient-il ?

MIME nous convient car :

- MIME a été conçu indépendamment d'environnements logiciels particuliers.
- MIME permet de spécifier la nature du document grâce à ses types et sous-types. Exemples de types : text, audio, image,... Exemples de sous-types pour le type text: plain, html.
- MIME est défini de manière récursive : une entité MIME peut être du type «multipart», et contenir des parties MIME différentes. Ceci nous permet de spécifier et de manipuler des documents multimédia facilement (le «file attach» en courrier utilise ce principe).
- MIME permet de faire référence à une source de donnée externe : une entité MIME du type «external» indique que son contenu peut se trouver sur disque ou sur un site distant.
- MIME est extensible : on peut définir de nouveaux types et sous-types MIME à souhait, et associer des applications à ces sous-types.
- ...

Notons aussi que ce que NetScape appelle «Helper Apps» dans son célèbre navigateur (browser web), n'est autre qu'un sous-ensemble de types / sous-types MIME. Il s'agit en fait -implicitement- de la description des structures de données que peut manipuler le poste de travail du client WWW. Nous reprendrons ceci dans la spécification des postes de travail.

[10] Internet Engineering Task Force

Similairement, MS-Windows95 parle lui de «File Types» pour désigner la définition des types / sous-types MIME.

1.2.4. Qu'allons-nous garder de MIME ?

MIME est peu utilisé comme standard pour spécifier l'encapsulation de données de différents formats au sein d'un même document. En conséquence, très peu de programmes de conversion de formats permettent d'utiliser MIME pour spécifier l'encapsulation.

C'est pourquoi nous nous orienterons vers le format HTML pour ce qui est de l'encapsulation de données : en effet, HTML nous permet - avec même plus de souplesse - de réaliser l'équivalent des fonctionnalités *multipart/mixed* et *external* de MIME. Nous pouvons donc associer à un élément d'un fichier HTML toute entité sonore, audio, vidéo, ou spécifique à une application. C'est exactement ce dont nous avons besoin.

Pour le reste, MIME nous servira toujours pour spécifier le format (externe) de données. Nous utiliserons les types MIME exactement comme le font les navigateurs Web («helper apps»), et même Windows («files types»).

1.2.5. MIME et OLE

MIME n'est pas le seul standard à permettre de spécifier l'encapsulation de documents: il y a également OLE.

OLE
OLE signifie *Object Linking and Embedding* [DELPHI, pg.819]; c'est une norme de Microsoft définissant l'encapsulation dans des documents d'entités de types différents (texte, tableau, graphique, image,...). Cette norme est toutefois restreinte à l'environnement ms-windows.

Par rapport à OLE, MIME est un standard plus ouvert, mais pas seulement parce qu'il est multi-plateformes, mais également parce qu'il peut se combiner facilement à d'autres systèmes.

Par exemple, MIME peut être combiné avec HTML comme nous venions de l'évoquer : les clients et serveurs Web (HTTP) utilisent d'ailleurs MIME comme standard d'identification des types d'informations. MIME est également utilisé dans des environnements «agents» : ***«Les agents sont des objets logiciels portables qui peuvent "voyager" d'un ordinateur à l'autre dans le but d'accomplir certaines fonctions.»*** [TAES, pg.11]. Dans le contexte MIME-Internet, les langages agents les plus répandus sont TCL et Java.

OLE est lui un standard monolithique, fermé. Il a récemment pris le nom d'ActiveX, supportant le concept d'agent qui vient d'être exposé (ActiveX est donc un concurrent de Java).

Nous proposons l'adoption de la lignée MIME/HTML comme standard, étant donné que nous voulons donner une dimension universelle à notre système, et qu'il s'appuiera sur les technologies Internet.

L'adoption exclusive du standard OLE nous limiterait à l'environnement ms-windows. Dans ce cas, il aurait été plus opportun de développer une application windows basée sur des documents de la suite ms-office (winword, excel, etc.).

Cependant, notre système utilisera le standard OLE au niveau de la conversion externe. En effet, si un document utilisant OLE parvient au système PBFlow, il devra pouvoir être pris en charge par notre système.

III.1.3. Structure interne d'un type de formulaire

Un principal intérêt de la spécification interne d'un type de formulaire est, comme signalé précédemment, de permettre l'interfaçage le plus automatique possible entre notre système inter-organisationnel et les différents systèmes intra-organisationnels.

Pour y parvenir, nous proposons une méthode comportant deux étapes majeures et reposant sur le concept de *gabarit* (ou template, ou modèle). Nous étendrons ensuite cette méthode aux documents en général et aux situations ad-hoc.

Gabarit (template)
Un gabarit d'un document ou formulaire est un descriptif de sa structure interne, qui identifie les unités informationnelles contenues et qu'on désire manipuler de manière autonome. Notre notion de gabarit s'inspire de celle de patron en couture, de template en informatique, ou encore de document-type en matière administrative[11].

Concepteur de gabarit
Pour chaque type de formulaire échangé dans l'application permis de bâtir, un «concepteur de gabarit» aura pour rôle de spécifier les gabarits, les document-types. Nous préférons le terme de «concepteur de gabarits» plutôt que les termes classiques d'*application engineer*, ou *application developer*, termes peu adaptés dans notre contexte [POOS94, pg.69]. Ces concepteurs de gabarit peuvent tout aussi bien faire partie des organisations utilisatrices, que d'un comité inter-organisationnel.

En annexe, nous avons joint une liste - non-exhaustive - de quelques 80 formulaires utilisés actuellement par les services d'administration d'urbanisme des communes en région wallonne, rien que pour la délivrance de permis de bâtir/lotir, et rien qu'au niveau des documents produits par ces services. Au total, ce sont plusieurs centaines de types de documents qui sont manipulés.

La similitude frappante entre certains types de formulaires, comme le modèle «ENQ-OUAV» pour les permis de bâtir, de lotir et d'urbanisme nous amènent à réfléchir sur la nécessité et les conséquences d'avoir autant de types de formulaires. Notre explication est que la gestion du flux se fait essentiellement en fonction de ces types de formulaires. Par exemple, si une tâche donne comme résultat un document de type «Refus d'un permis de bâtir» (REFUS-C), une tâche de notification au demandeur et à la région démarrera.

Nous proposerons par contre une gestion du flux qui pourra tenir compte d'informations contenues dans les documents. Le grands nombre de documents administratifs à gérer nous impose toutefois à adopter une méthode simple, permettant de

[11] Nous nous inspirons également de logiciels que nous avons écrit précédemment.

créer facilement des gabarits à la volée, sur base de documents réels et non par d'harrassantes définitions préalables. Ceci permettra aussi de prendre en compte des situations ad-hoc plus facilement.

Nous allons à présent détailler les étapes de conception d'un gabarit en illustrant notre méthode sur base d'un cas réel : la manipulation du formulaire «Annexe CWATUP n°21» envoyé par un architecte avec chacune de ses demandes de permis de bâtir. Nous verrons comment ce document peut être traité pour être pris ensuite en charge par un système de workflow intra-organisationnel particulier. Puis, nous formaliserons cet exemple.

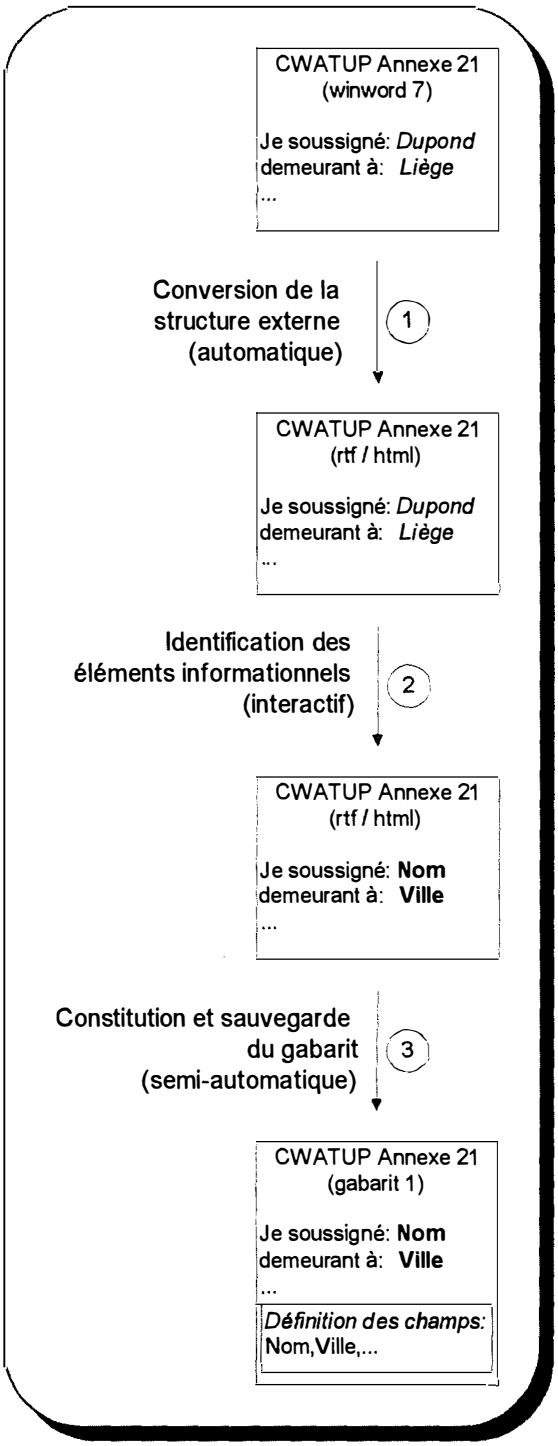


Figure III-5 : Méthode de création d'un gabarit

1. *Uniformisation de la structure externe*. Un architecte envoie une demande de permis de bâtir au système PBFlow. Cette demande contient, entre autres, le formulaire CWATUP, annexe 21. L'architecte utilisant winword 7.0, c'est sous ce format que le document parvient au système. La première étape pour PBFlow consiste à convertir ce document en un format plus maniable pour lui, standard, et multimédia (ici: HTML). Cette opération est effectuée automatiquement.

2. *Identification de la structure interne*. Le document issu de l'étape précédente est présenté au concepteur de gabarit. Cet utilisateur sélectionne à la souris les données (instances) du document, une à une. Pour chacune d'elle, il indique un nom de champ significatif du type de données, lequel se substituera à la donnée. Par exemple, il sélectionne *«Dupond»* et indique *«Nom architecte»* comme nom de champ.

3. *Enregistrement du gabarit*. Les champs étant définis univoquement, l'utilisateur peut encore facultativement modifier la définition des champs (dans le cas d'un document moins structuré par exemple). Ensuite, le gabarit est sauvegardé.

La même démarche peut être adoptée par les différents intervenants. Les concepteurs de gabarits spécifieront ainsi à l'aide d'une application indépendante les gabarits des différents types de formulaires utilisés.

La figure III-6 qui suit donne un exemple de formulaire CWATUP annexe 21 envoyé par un architecte (note: les données contenues sont elles fictives).

CWATUP Annexe 21
Attestation de l'architecte

Je soussigné : **Albert Douchamps**

demeurant à : **Bruxelles**
Rue : **Avenue de l'Hannibal** N°: **214**
Code Postal : **1150** Localité : **Woluwe-Saint-Pierre**

Téléphone : **02/ 771.09.26**

Atteste :

1. que je suis (cochez la mention adéquate):
   (**X**) inscrit au tableau de l'Ordre des Architectes de la province de Namur;
   ( ) porteur de l'autorisation visée à l'article 8, alinéa 3, de la loi du 26 juin 1963 créant un Ordre des Architectes, autorisation qui m'a été accordée par le Conseil de l'Ordres des Architectes de la province de _____ le __/__/____

2. que j'ai été chargé par M. Bodard
   [X] de l'établissement des plans relatifs à l'installation d'une teinturerie (nature et destination du projet) à effectuer sur une parcelle sise à Etterbeek, chaussée Saint-Pierre 328
   (commune, rue, n°)
   [X] du contrôle de l'exécution des travaux imposé par la loi.

[...] suite effacée dans l'exemple pour but de brièveté (modèle complet en annexe)

Fait à **Bruxelles**, le **12/08/1996**.

Figure III-6 : Extrait d'un document winword (formulaire)

La conversion de la structure externe de ce document ne posera pas de problème: de nombreux logiciels le permettent et reconnaissent des dizaines de formats de fichiers (= de structures externes). Nous déciderons ultérieurement le format que nous adopterons pour notre système PBFlow.

Certes, certains attributs du document original seront perdus, comme la présentation et la mise-en-page, la gestion des styles et polices de caractères, etc. Toutefois, ces attributs sont sans intérêts pour nous : l'application permis de bâtir requiert essentiellement l'utilisation de formulaires (structurés) et non de documents où mise-en-page et présentation sont importants.

Pour le reste, il suffit de se fixer comme règle simple que toute information utile doit se trouver écrite dans le document. En particulier, le nom de l'auteur ainsi que la date du document devront figurer texto dans celui-ci.

Enfin, rappelons que l'utilisation d'un en-tête MIME permettra de convertir le document de manière correcte [III.1.2]. Cette première étape est nécessaire pour permettre d'écrire un gestionnaire, au sein de PBFlow, qui ne doivent tenir compte que d'un seul format de fichier. Nous déciderons ultérieurement lequel sera le plus approprié.

La seconde étape consiste, pour le concepteur de gabarits, à sélectionner à la souris les informations dynamiques du formulaire, et à attribuer pour chacune d'elle un nom unique dans le document. Voici ce que cela peut donner dans notre exemple :

CWATUP Annexe 21

Attestation de l'architecte

Je soussigné : **NOM_ARCHITECTE**

demeurant à : **VILLE_ARCHITECTE**
Rue : **RUE_ARCHITECTE** N°: **NO_RUE_ARCHITECTE**
Code Postal : **CP_ARCHITECTE** Localité : **LOCALITE_ARCHITECTE**

Téléphone : **TELEPHONE_ARCHITECTE**

Atteste :

1. que je suis (cochez la mention adéquate):
   (**CHOIX_1A**) inscrit au tableau de l'Ordre des Architectes de la province de PROVINCE_1A;
   (**CHOIX_1B**) porteur de l'autorisation visée à l'article 8, alinéa 3, de la loi du 26 juin 1963 créant un Ordre des Architectes, autorisation qui m'a été accordée par le Conseil de l'Ordres des Architectes de la province de **PROVINCE_1B** le **DATE_1B**.

2. que j'ai été chargé par CHARGE_PAR
   [**CHOIX_2A**] de l'établissement des plans relatifs à NATURE_PROJET (nature et destination du projet) à effectuer sur une parcelle sise à COMMUNE_PROJET, RUE_PROJET, NO_PROJET (commune, rue, n°)
   [**CHOIX_2B**] du contrôle de l'exécution des travaux imposé par la loi.

[...] suite effacée dans l'exemple pour but de brièveté (modèle complet en annexe)

Fait à **FAIT_A**, le **FAIT_LE**.

Figure III-7 : Exemple de transformation d'un formulaire en gabarit

Une représentation des données (dynamiques) identifiées dans ce gabarit peut être donnée par la table donnée à la page suivante.

Nom du champ	Val./défaut	Texte avant	Attribut	Texte après
NOM_ARCHITECTE		Je soussigné:	G	
VILLE_ARCHITECTE		demeurant à:	G	
RUE_ARCHITECTE		Rue:	G	
NO_RUE_ARCHITECTE		N°:	G	
CP_ARCHITECTE		Code Postal:	G	
LOCALITE_ARCHITECTE		Localité:	G	
TELEPHONE_ARCHITECTE		Téléphone:	G	
CHOIX_1A		(	G	) inscrit au tabl...
PROVINCE_1A	Namur	de la province de		;
CHOIX_1B		(	G	) porteur de l'aut..
PROVINCE_1B		de la province de	G	le
DATE_1B		le	G	.
CHARGE_PAR		j'ai été chargé par	S	[
CHOIX_2A		[	G	] de l'établisse...
NATURE_PROJET		des plans relatifs à	S	(nature et desti...
COMMUNE_PROJET		une parcelle sise à	S	⊿
RUE_PROJET		⊿	S	⊿
NO_PROJET		⊿	S	(commune, rue,...
CHOIX_2B		[	G	] du contrôle de...
FAIT_A		Fait à	G	, le
FAIT_LE		le	G	.

Figure III-8 : exemple partiel de tableau d'un gabarit (cwatup annexe 21)

Cette table reprend, dans l'ordre, les différents champs (éléments informationnels) du texte. La colonne «val/défaut» permet au développeur d'application de mettre une valeur par défaut à un paramètre. Les trois colonnes «Texte avant», «Attribut» et «Texte après» permettent au programme d'identifier les informations, et en particulier l'utilisation d'attribut **gras** et souligné. Notons que, pour la comparaison du texte précédent et suivant, notre système pourra ignorer : la casse (= différence entre majuscules et minuscules), les accents et les espaces.

L'utilisation d'une telle table n'est pas indispensable pour les formulaires : tous les éléments peuvent être placés dans le texte du gabarit même, mais la présentation d'un tel tableau au concepteur de gabarit peut être utile. Notamment, la liste des unités informationnelles (champ) est donnée de manière synthétique. Nous pourrions encore rajouter une colonne facultative contenant une description.

L'utilisation de ce genre de table sera par contre davantage nécessaire pour spécifier des documents moins structurés. Nous examinerons plus loin [III.5. et IV.2.] une proposition pour le traitement de documents dans des situations ad-hoc.

III.2. Stockage des informations dans PBFlow

Maintenant que nous avons vu de quelle manière nous spécifions un document (formulaire), il nous reste à voir ce que nous devons stocker comme informations dans notre système PBFlow (III.2.1.), et comment (III.2.2.). Nous aborderons en particulier le cas des formulaires multimédias (III.2.3.), et nous verrons alors comment il est possible de spécifier les postes de travail (III.2.4.).

III.2.1. Quelles informations stocker dans PBFlow

Nous devons tout d'abord stocker les informations relatives aux spécifications et au déroulement du workflow : les types de tâches, tâches, types de documents, documents, etc. Cela, nous l'avions déjà signalé et nous avons donné le schéma Entité/Association de cette base de données [II.3.]. Nous pourrons par exemple utiliser une base de données Oracle pour stocker ces informations.

Mais il nous faut aussi stocker les informations échangées entre les partenaires, avec comme objectif sous-jacent de permettre un interfaçage entre les différents systèmes inter- et intra-organisationnels.

Stocker les documents reçus tels quels n'est pas une bonne idée dans notre contexte de permis de bâtir, car la plupart du temps ce sont des formulaires qu'on manipule. Seules les informations dynamiques nous intéressent, localisées dans les champs des formulaires. Or, la construction de gabarits nous a permis d'identifier ces informations dynamiques. Ceci nous permet d'affirmer que toute instance de formulaire peut être représentée comme l'illustre la figure suivante.

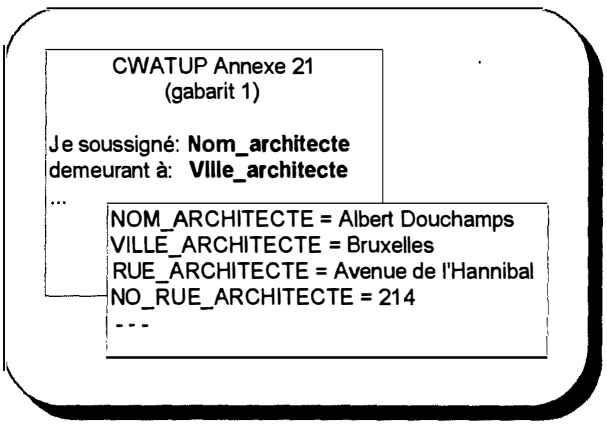


Figure III-9 : Exemple des informations stockées d'un formulaire

Le formulaire stocké se compose de deux parties :

1. *une partie invariable*, c'est le gabarit du document;
2. *une partie variable*, ce sont les données (dynamiques).

Pour la partie variable, nous pouvons à nouveau utiliser une base de données classique, avec des tables pour chaque type de document administratif.

Pour la partie invariable, le gabarit, nous proposons d'utiliser le format HTML (HyperText Markup Language).

III.2.2. Choix de HTML comme format standard

Le problème du choix d'un format de document n'est pas un problème nouveau comme en témoigne cet extrait (SGML86, A Basic Introduction, 1986) :

« Did you ever wish that instead of maintaining a document in a bunch of different formats you could keep your documents in just one format that every computer and every application could understand ? »

SGML, acronyme de *Standard Generalized Markup Language*, est un standard né en 1986, qui a pour but de proposer une représentation «format-neutral and content-enriching». La principale innovation de SGML, c'est la manière dont il stocke les informations : alors que les programmes habituels se préoccupent de la manière dont un document sera affiché ou imprimé, SGML lui se préoccupe des types d'informations contenues dans le document.

Ce qui est intéressant, est que SGML permet d'isoler n'importe quel composant informationnel d'un document SGML. Pour ce faire, SGML utilise un format simple, textuel, enrichi de "*tags*". Un *tag* est une annotation textuelle standard utilisée pour décrire le contenu d'un texte ou document multimédia. En voici un exemple :

```
<PARAGRAPH>Ceci est un paragraphe .</PARAGRAPH>
<VIDEO SOURCE="FILE///demo.mpg">
```

Les tags utilisés indiquent ce que l'information est, soit dans notre exemple un paragraphe et une animation vidéo. La manière dont l'information sera affichée ne rentre pas dans cette description.

Parallèlement, les tags utilisés sont décrits dans un DTD - *Document Type Definition* - encore appelé "*Rules File*". Un sous-ensemble extrêmement connu de DTD n'est autre que... HTML, *Hyper Text Markup Language*.

SGML répond exactement à nos besoins, et plus précisément HTML, car :

- HTML un format standard universellement (re)connu, notamment pour le World-Wide-Web.
- W3Flow est précisément basé, d'un point de vue technique, sur l'Internet Suite; ceci permettra au sein du système PBFlow (dérivé de W3Flow), de présenter directement aux utilisateurs les formulaires, au format HTML, et donc lisibles par n'importe quel browser web (navigateur internet).
- HTML est un format très maniable pour un logiciel, notamment parce qu'il est essentiellement basé sur du texte (son type MIME est d'ailleurs *text/html*).
- HTML permet de manipuler des documents multimédias, intégrant texte, image, animations vidéos, son, etc. ainsi que tout autre format qu'on peut associer à une application.
- ...

Une limitation cependant : si un champ dynamique est défini dans un document (winword par exemple), le nom de ce champ dynamique sera perdu lors de la conversion (il s'agit d'une limite intrinsèque aux programmes de conversion).

III.2.3. Formulaires multimédias

Le formulaire CWATUP que nous avons présenté dans les sections précédentes contient essentiellement du texte. La question qui se pose maintenant est de savoir si la méthode que nous avons exposée s'applique à des documents multimédias, tenant compte du choix pour le format HTML. Par exemple, un document pourra contenir, outre du texte, une image, un plan d'architecte, un tableau Excel ou 1-2-3, une animation vidéo, etc.

Deux cas sont à distinguer :

1. Les tableaux Excel, fichiers dBase, etc. peuvent être aisément convertis vers le format .HTML. Il va de soi que nous choisissons la norme 3 de HTML, supportant la gestion de tables notamment. Ces cas ne posent donc pas de problème.
2. Les sons, images, et données propres aux applications (ex.: plan d'architecte) ne peuvent être complètement traduits en HTML. Ce cas mérite notre attention.

La figure suivante III-10 illustre le processus de transformation d'un document multimédia vers le format .HTML. Cette transformation est effectuée automatiquement par des logiciels tels que HTML-Transit ™, ou les produits SoftQuad ®.

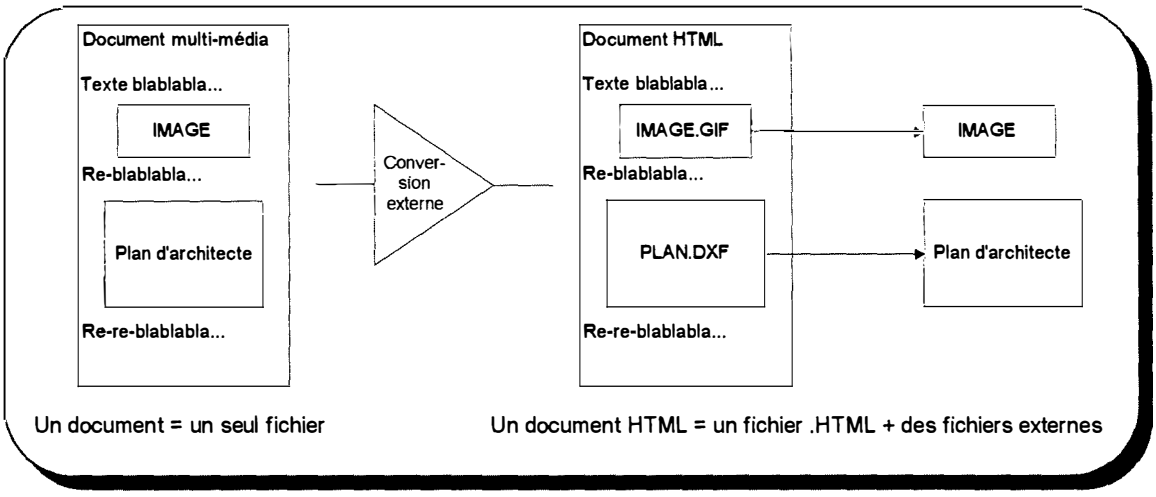


Figure III-10 : Conversion d'un document multimédia vers le format HTML

Que pouvons-nous observer ? Le document multimédia de départ occupait un seul fichier : par exemple, il pouvait s'agir d'un document wordperfect qui intégrait -grâce à OLE- une image au format TIFF ou PCX, ainsi qu'un plan d'architecte au format .DXF d'AutoCAD. Toutes ces parties sont intégrées directement dans le même fichier (note: il est possible d'utiliser des liens vers des fichiers externes, mais cela ne change rien à notre propos).

Le document d'arrivée n'occupe plus un seul fichier, mais trois : seules les informations représentables sous forme textuelle (text/html) se trouvent dans le fichier .HTML créé. Les autres informations, image et plan, se trouvent dans des fichiers séparés.

Dans le fichier .HTML, on retrouvera, à la place de l'image et du plan, le nom des fichiers contenant ces informations. Par exemple:

```
<IMAGE SRC=Maison.GIF>
<A HREF= «file:///Plan.DXF»>Plan d'architecte</A>
```

Nous pensons raisonnable de faire l'hypothèse que ces informations (image et plan) sont non-décomposables dans le contexte de notre application de workflow. Si les besoins nécessitent de gérer des éléments contenus dans ces entités indépendamment, alors il sera nécessaire de créer dès le départ plusieurs images séparées, ou plusieurs plans séparés.

Cette hypothèse étant faite, nous pouvons considérer tout formulaire multimédia comme un formulaire texte. La seule chose qui changera pour le concepteur de gabarits, est qu'il ne verra pas directement certaines données apparaître sur le formulaire, mais seulement le nom des fichiers les contenant.

Il reste une question importante : que deviennent les formats des fichiers encapsulés ? Sont-ils également convertis ?

Nous proposons d'adopter la proposition suivante :

- pour les formats de données propres à des applications (ex.: fichier .DXF d'AutoCAD), nous proposons de garder les fichiers tels quels au sein de PBFlow;
- pour les formats de données textuelles, sonores, images et animation vidéos, nous proposons de les convertir vers le format le plus universel pour chaque type de données; par exemple, GIF/TIFF/JPG pour les images, AU/WAV pour le son, MOV/AVI pour les animations vidéos. PBFlow utilisera lui ces formats universels pour stocker ses données.

Néanmoins, ces données encapsulées devront parfois être reconverties lorsque les documents devront être exportés vers un poste de travail. Heureusement, des logiciels de conversion de documents comme HTML Transit permettent de spécifier ce que deviendront les formats des fichiers encapsulés [Transit]. Ces logiciels assurent donc une conversion de la structure externe des fichiers, mais de manière récursive : pour chaque type de données, on peut indiquer le format cible.

Maintenant, la question est de savoir comment savoir quel est le format cible approprié : pour cela, notre système doit tenir compte des caractéristiques du poste de travail pour lequel il prépare le document.

III.2.4. Spécification des postes de travail

Dans notre optique, spécifier un poste de travail revient à indiquer quelles sont les structures externes et internes que le système d'un acteur est capable de lire [voir II.3.2. et III.1.2. à III.1.3].

A ce propos, nous avions adopté MIME comme standard pour spécifier les structures externes [III.1.2.]. La spécification des structures externes supportées par un poste de travail ressemblera donc à une liste de formats MIME comme celle-ci :

```
audio/x-wav        audio/basic          image/x-MS-bmp
image/tiff         image/gif            text/plain
text/html          text/ascii           video/quicktime
video/mpeg         application/word6    application/excel7
```

Or, il se fait que le protocole HTTP permet au serveur HTTP (Web) de connaître la liste des formats MIME supportés par la machine cliente : la primitive *accept()* permet cela. Nous l'utiliserons au besoin pour définir dynamiquement, à l'exécution, les caractéristiques des postes de travail qui interagissent via le Web avec notre système PBFlow.

La spécification d'un poste de travail contiendra en outre la liste des gabarits (structures internes) manipulées par ce poste de travail.

III.3. Fonctionnalité d'interfaçage

Dans W3Flow, la fonctionnalité d'interfaçage avec les systèmes d'information intra-organisationnels est présentée comme essentielle. Nous allons montrer comment cet interfaçage peut se réaliser dans PBFlow. Tout d'abord, nous examinerons l'interfaçage entre formulaires de types différents (III.3.1), et ensuite nous montrerons comment un formulaire peut être traité pour être pris en charge par les systèmes intra-workflow des partenaires (III.3.4 et suivants), ceci grâce à diverses extensions à notre système (III.3.2 et suivants).

III.3.1. Interfaçage de formulaires

Nous avions défini un gabarit pour l'annexe 21 du CWATUP envoyée par un architecte. Parallèlement, d'autres gabarits pour ce même document auront pu être composés, par le service d'urbanisme de la ville de Namur par exemple. Il est à noter que bon nombre de formulaires (comme les CWATUP) sont imposés au niveau de la région wallonne.

L'étape suivant pour assurer l'interfaçage consiste à mettre en correspondance ces deux gabarits : ceci se fait très simplement pour le concepteur de gabarits en association les champs 2 à 2 de chaque gabarit. Une fois fait, notre système sera capable de faire automatiquement la conversion du formulaire CWATUP annexe 21. La figure III-11 qui suit illustre le procédé.

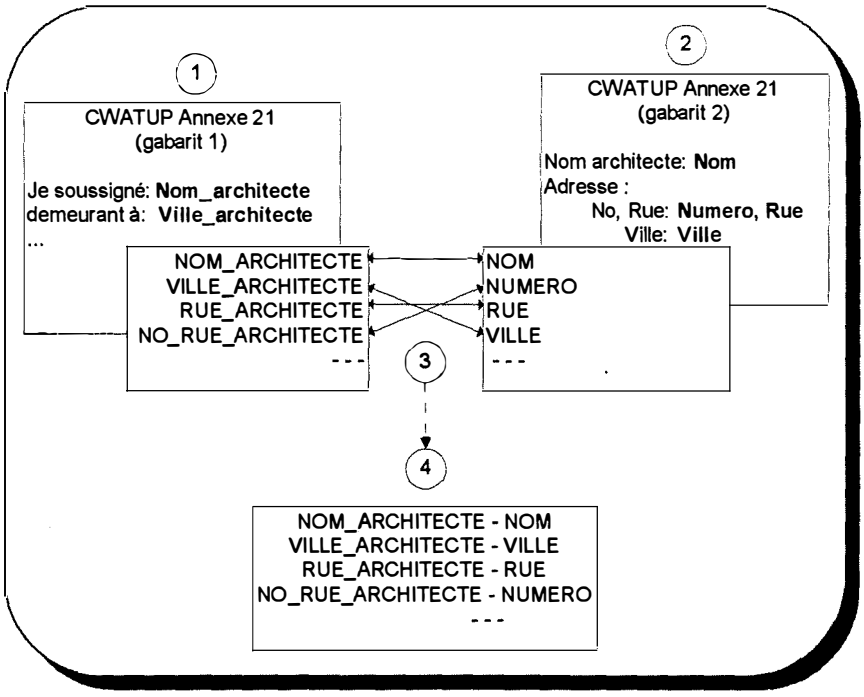


Figure III-11 : Exemple de fonctionnement de l'interfaçage de formulaires dans PBFlow

Explication de la figure III-11:

1. Un gabarit est créé à partir d'un document réel (format source de l'architecte);
2. Un gabarit est créé à partir d'un document réel (format cible utilisé par la ville);
3. La mise-en-correspondance des deux gabarits est effectuée;
4. Le résultat est une association entre chaque champ (information) des deux formats de documents.

En pratique cependant, nous utiliserons au sein de PBFlow un gabarit unique pour chaque type de document. Nous nommerons ce gabarit le gabarit standard, ou gabarit par défaut.

Si nous n'adoptons pas de gabarit unique pour chaque type de document, nous devrions prévoir des conversions entre chaque type de gabarit. Par exemple, mettons qu'il y ait 5 acteurs, utilisant respectivement pour un type de document les gabarits A, B, C, D et E. Nous devrions donc prévoir 10 conversions : A↔B, A↔C, A↔D, A↔E, B↔C, B↔D, B↔E, C↔D, C↔E, D↔E. Avec seulement une douzaine d'acteurs, le nombre de conversions différentes possibles serait de 77.

En conséquence, l'adoption d'un gabarit standard permet de limiter les conversions:

- à l'importation vers PBFlow: d'un format propriétaire (gabarit de l'acteur qui envoie le document) vers le gabarit standard;
- à l'exportation vers un acteur: du format standard (gabarit de PBFlow) vers le gabarit propriétaire du poste de l'acteur qui recevra le document.

Avec une douzaine d'acteurs qui utiliseraient chacun des gabarits différents, le nombre de conversion n'est alors plus que de 12. L'importation d'un document dans notre système PBFlow s'opérera donc de la manière suivante (figure III-12):

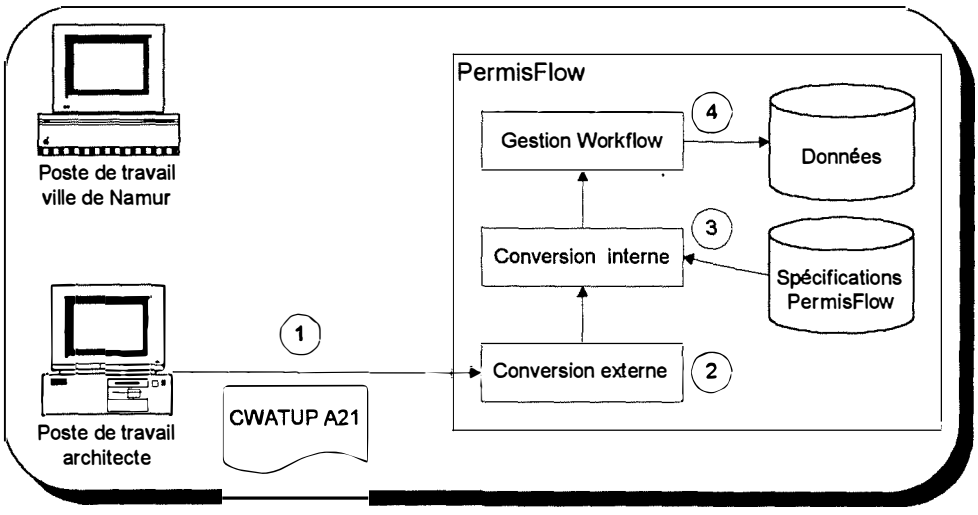


Figure III-12 : Importation d'un formulaire dans PBFlow

Explications de la figure III-12:

1. L'architecte envoie son formulaire complété et envoyé au système PBFlow.
2. La conversion de la structure externe du document vers le format .HTML est automatiquement opérée, sur base de l'en-tête MIME du document.
3. La conversion de la structure interne du document est effectuée. Le format source est le gabarit correspondant au document entrant (CWATUP A21) pour le poste de travail source (Poste architecte). Le format cible est celui défini dans le système PBFlow pour les documents CWATUP A21.
4. Les données sont stockées dans le système PBFlow, et le gestionnaire de flux (workflow) peut les exploiter. Nous y reviendrons lorsque nous parlerons de la gestion du flux. Dans ce cas-ci, le document reçu doit être envoyé[12] au service d'urbanisme de la ville de Namur. C'est ce qui est illustré sur la figure suivante (III-13).

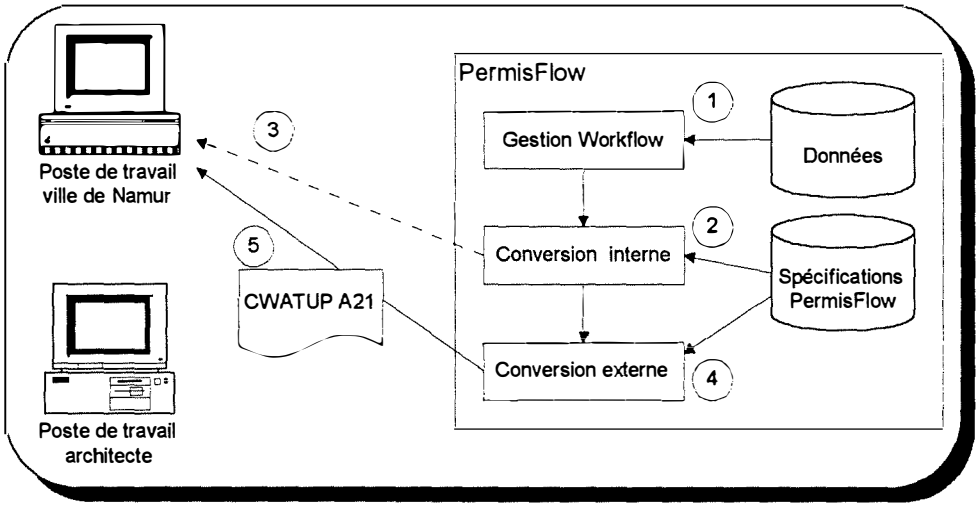


Figure III-13 : Exportation d'un formulaire dans PBFlow

Explication de la figure III-13:

1. Le système de workflow doit envoyer le formulaire CWATUP A21 au service d'urbanisme de la ville de Namur. Pour ce faire, il extrait les données de ce formulaire dans sa base de données.
2. La couche de conversion interne reçoit les données à envoyer. La conversion de la structure externe va ici avoir pour effet d'habiller les données selon le gabarit utilisé. Le gabarit utilisé sera le CWATUP A21 pour le poste de travail d'administration de la ville de Namur. Si aucun gabarit n'est défini pour ce poste de travail, le gabarit utilisé sera le gabarit standard, celui défini par défaut au sein de PBFlow. Les spécifications de ce gabarit sont enregistrées dans la bases des spécifications du système PBFlow.

[12] = nous passons ici sous silence les conversations.

3. A ce stade-ci, on peut déjà proposer au récepteur du document la possibilité de le visualiser avec un simple browser web. Les deux étapes suivantes ne seront nécessaires que s'il décide effectivement de rapatrier ce formulaire sur son système local. Néanmoins, nous verrons loin [III.3.4] qu'en pratique ce seront les gabarits standards de PBFlow (c'est-à-dire les gabarits définis par défaut) qui seront utilisés pour cette présentation du document aux acteurs, via le World-Wide-Web.
4. Une fois lancée, la conversion de la structure externe du document .HTML qui vient d'être créé est automatiquement opérée. Le format cible est défini dans la base des spécifications du système selon le poste de travail. Ici, le poste de travail cible est celui du service d'administration de l'urbanisme de la ville de Namur.
5. Le document, arrivant au service d'urbanisme de la ville, peut être directement exploitable.

III.3.2. Utilisation de dictionnaires

Une première extension que nous proposons à notre système est l'utilisation de dictionnaires lors de la conversion de structure interne. La motivation derrière cette proposition tient du fait que différentes représentations d'une même information sont possibles.

Citons deux cas en exemple :

1. Un formulaire en néerlandais indique, pour le champ relatif à la province: «Namen».
2. Le système utilisé par un acteur utilise des valeurs «true/false» en lieu et place de boîtes à cocher.

C'est pourquoi nous proposons que, pour chaque association de champs, il soit possible au développeur d'application de choisir un dictionnaire traductif de valeurs. Voici deux exemples de dictionnaires :

Valeurs A	Valeurs B
Namen	Namur
Luik	Liège
...	...

Valeurs A	Valeurs B
' '	NON
X	OUI

Figure III-14 : Exemple de dictionnaires traductifs de valeurs

Ce qu'il est intéressant de constater est que le fait d'utiliser de tels dictionnaires permet toujours une transformation réversible, bidirectionnelle, des documents. En d'autres termes, si l'on peut transformer un document correspondant au gabarit A en un

document de gabarit B, il sera également toujours possible de transformer un document correspondant au gabarit B en document de gabarit A.

III.3.3. Valeurs par défaut

Un formulaire peut contenir des éléments facultatifs. Il est donc intéressant de proposer un mécanisme permettant de spécifier explicitement ce que ces informations deviennent.

Par ailleurs, une information peut être utile pour un organisme et inusitée chez un autre. Par exemple, les demandes qui arriveront à l'administration de la ville de Namur concerneront toujours des sites situés dans la province de Namur : pour eux, cette information n'est donc pas représentée.

C'est pourquoi nous proposons, aux concepteurs de gabarits, la possibilité de consulter et de modifier la représentation sous forme de tableau synthétique présentée à la figure III-8. Revoici un extrait, modifié, de cette table :

Nom du champ	Val./défaut	Texte avant	Attribut	Texte après
NOM_ARCHITECTE		Je soussigné:	Gras	
VILLE_ARCHITECTE		demeurant à:	Gras	
...	...	...	...	...
PROVINCE_PROJET	Namur		Hidden	
ASSUJETTI	non	Je suis assujetti:	Gras	
...	...	...	...	...

Figure III-15 : Exemple d'utilisation d'une valeur par défaut, pour un champ non-défini

Dans cet exemple, le paramètre PROVINCE_PROJET a été rajouté à la table, même s'il ne fait pas partie du document initial. Que se serait-il passé si ce champ n'avait pas été défini dans la table ? Lors de l'interfaçage avec un autre gabarit qui imposerait ce champ, la valeur nulle (vide) aurait été automatiquement assignée comme valeur, ce qui peut être gênant pour un autre partenaire, car pour lui cette information peut être nécessaire.

Autre exemple: le paramètre ASSUJETTI reçoit la valeur «non». C'est le corollaire du cas précédent : certains gabarits ne l'utilisent pas, par exemple les pays qui ne connaissent pas la TVA. Dans ce cas, la valeur «non» sera automatiquement assignée.

Notons qu'ici, pour la première fois, nous avons perdu une certaine symétrie: la valeur par défaut n'est utilisée lors l'interfaçage que dans un sens.

III.3.4. Correspondance unidirectionnelle

Ce que nous avons proposé jusqu'ici est la possibilité de traduire un formulaire en un autre formulaire. Ce n'est pas toujours suffisant. Lors de notre étude à ce propos nous avons fait les deux observations suivantes :

1. Tout partenaire dispose toujours d'un mécanisme simple pour *exporter* ses données sous forme de formulaire. Ceci est inhérent à tous les systèmes : il est facile d'exporter les informations contenues dans une base de données dans des formulaires. Néanmoins, le format utilisé (structures interne et externe) est généralement choisi par le partenaire, et lui imposer d'autres formats n'est pas toujours envisageable. Les contenus et types de formulaires par contre sont imposés de manière plus stricte par la loi ou la région wallonne.
2. Un partenaire dispose rarement d'un mécanisme simple pour *importer* des données contenues dans n'importe quel formulaire vers son système interne.

La première observation nous tranquillise : la gestion de formulaire est suffisante pour importer des données dans notre système PBFlow. La seconde observation nous amène à constater que notre modèle actuel, basé sur des gabarits simples et un procédé bidirectionnel, peut être insuffisant.

Pour illustrer ce fait, nous allons prendre un exemple plus complexe : le service d'administration d'urbanisme de la ville utilise une base de données SQL. Nous allons voir que l'utilisation de règles de correspondance permet de créer un script SQL qui va directement permettre d'injecter les données dans la base de données SQL de la ville.

Le gabarit pour l'annexe 21 du CWATUP pourrait donc être le suivant pour la ville (note: nous n'avons gardé ici que quelques attributs pour l'exemple).

```
Connect DATABASE@username/password;
INSERT INTO TABLE_ARCHITECTE
(NAME, NO, STREET, ZIP_CODE, CITY, PHONE, DATE, STATE)
VALUES
( NOM,RUE,NUMERO,CP,VILLE,TELEPHONE,DATE,PROVINCE );
```

Figure III-16: Exemple de script SQL pour l'importation des données

Sur la ligne du dessous, nous reconnaissons les noms des champs définis dans le gabarit. Ces noms de champs sont mis en correspondance avec ceux de l'autre gabarit que nous avons présenté plus haut [III.1.3], exactement comme indiqué en [III.3.1].

La différence essentielle entre ce gabarit-ci et les précédents, est qu'il n'est pas bidirectionnel. En effet, lorsque le système de la ville va générer la liste des architectes, ce ne sera probablement pas un script SQL avec des «INSERT INTO».

Ce genre de gabarit présente aussi un avantage indéniable, car on peut spécifier que les scripts .SQL soient du type MIME «application/x-SQL». Lorsque le client chargera pour la première fois son script .SQL, son navigateur web lui proposera de sauvegarder le script .SQL sur son disque dur (par exemple), ou mieux d'associer ce type MIME à une application, telle qu'un interpréteur SQL. Lorsque l'acteur aura réalisé cette association ("helper apps" dans Netscape), chaque fois qu'il téléchargera un script SQL, il pourra directement être exécuté.

Ceci nous amène à conclure que, dans nombre de cas, le gabarit à utiliser à l'importation ne sera pas le même à l'exportation. Ceci n'est pas dramatique: il ne sera pas nécessaire de définir deux gabarits à chaque fois. Un partenaire est en effet rarement fournisseur et demandeur d'un même type de données. Par exemple, seul un architecte aura à remplir le formulaire CWATUP annexe 21, les autres ne devront que l'importer.

Néanmoins, ceci pose problème pour la présentation des formulaires aux acteurs, via le World-Wide-Web. Il est en effet hors de question de proposer un script SQL à l'écran de celui qui veut consulter les informations. C'est pourquoi, pour l'étape de présentation des données via le WWW [III.3.1, fig. III-13], sera réalisé avec le gabarit HTML défini par défaut au sein de PBFlow pour ce type de document.

III.3.6. Limites de notre approche

La méthode que nous proposons a ses limites. Elles sont de deux types :

1. les mises-en-correspondance de gabarits restent assez simples, et ne peuvent pas toujours permettre de prendre en compte une plus grande complexité (ex.: cas de scripts SQL plus complexes).
2. les mises-en-correspondance ne tiennent compte que de l'aspect «techniquement possible» de l'interfaçage, sans ce soucier - entre autres - si les droits d'accès nécessaires et la disponibilité de la base de données sont assurés.

Pour répondre au premier point, nous pouvons étendre notre système avec l'instauration de règles de correspondances. Ces règles de correspondance permettraient de «calculer» la valeur de certains champs en fonction d'autres, là où l'utilisation de dictionnaires de valeurs s'avérerait insuffisante. Par exemple, une date au format JJ/MM/AAAA pourrait alors être traduite au format AAAA-MM-JJ. Notre système tel

que nous l'avons présenté ne permet pas ces conversions, mais des extensions sont aisément envisageables.

Pour répondre au second point, nous rappelons que nous ne pouvons nous immiscer dans les systèmes intra-organisationnels que jusqu'à un certain point. Chaque organisation reste toujours pour nous une boîte noire dont nous ne connaissons que ce qui en sort et ce qui y rentre. Nous nous limitons donc à proposer un système qui permette aux acteurs d'éviter au maximum des ré-encodages fastidieux, mais la gestion interne de leurs systèmes restera sous leur responsabilité.

III.4. Spécification des Entrées/Sorties de tâches

Nous avons maintenant presque tout ce qu'il faut pour définir les Entrées/Sorties de chaque type de tâche : pour chaque type de tâche, il suffit d'indiquer les types de documents reçus et générés. Il est en effet inutile de spécifier les structures externe et interne au niveau de la tâche puisqu'elles dépendent des postes de travail et non des tâches en elles-mêmes.

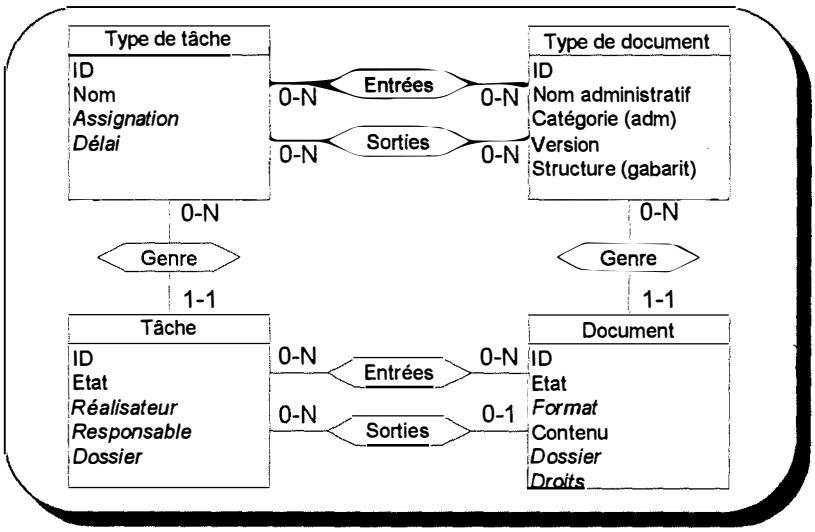


Figure III-17: Extrait du schéma E/A de PBFlow; (types de) tâches et documents

La figure III-17 ci-dessus rappelle les éléments à spécifier pour les types de documents et documents, et leurs relations avec les tâches et types de tâches.

Les Entrées/Sorties de tâches sont les documents reçus et générés dans le cadre de l'exécution d'une instance particulière d'une tâche donnée. Ces Entrées/Sorties ne sont pas prédéfinies mais vues uniquement à l'exécution.

Les Entrées/Sorties de types de tâches par contre sont définies a priori, et sont des types de documents. La question qui se pose est de savoir si les types de documents en

entrée et sortie de type de tâches peuvent toujours être connus d'avance. La réponse est, dans le cas de notre application permis de bâtir : non. Par exemple, s'il est possible de définir a priori une tâche « élaboration d'un avis par la commission des "Sites Mosans et Champêtres" », le type de document fourni par ce type de tâche ne sera pas nécessairement connu d'avance.

Pour résumer, ce que nous représenterons dans notre modélisation, sont :

- *en entrée de type de tâche*, les types de documents nécessaires à son exécution. Il s'agit donc d'une définition a priori, et lors de l'exécution d'une instance de la tâches, d'autres documents additionnels pourront être requis (ex.: un PPA, plan particulier d'aménagement).
- *en sortie de type de tâche*, les types de documents qui peuvent être générés, étant entendu que toute tâche produira au moins un document si elle achève son exécution (ex.: une attestation d'acceptation de permis de bâtir, ou un refus de permis de bâtir).

Il reste maintenant à examiner la prise en compte de situations non-prédéfinies.

III.5. Prise en compte de situations ad-hoc

Tout ce que nous avons décrit précédemment peut fonctionner parfaitement dans des cas prédéfinis, lorsque le concepteur de gabarits aura défini tous les gabarits utilisés. En pratique, ce ne sera pas toujours le cas. C'est pourquoi nous allons présenter différents cas non-prédéfinis, et la manière de les traiter, toujours dans le contexte de notre application permis de bâtir.

A. La structure externe d'un document a changé.

Un architecte qui utilisait Winword version 6 passe à Winword version 7 ou à Lotus Wordpro. Ce changement de structure externe est sans incidence pour nous. Dans le cas des formulaires, la conversion de structure externe vers le format .HTML est toujours opérée automatiquement dans PBFlow grâce à l'en-tête MIME des documents : ceci permet d'uniformiser de manière automatique la structure externe (le format) du document.

Le seul problème qui pourrait se poser est qu'un poste de travail adopte un format inconnu pour notre convertisseur (par exemple, GeoWrite v2.01). Dans ces rares cas, il sera demandé à l'utilisateur d'exporter lui-même ses documents dans un format plus courant (fonction disponible dans pratiquement tous les logiciels).

B. La structure interne d'un document a changé.

Si la structure interne d'un document change, un concepteur de gabarit devra définir un nouveau gabarit, éventuellement sur base de l'ancien si la structure n'a pas trop changé. C'est la raison pour laquelle nos types de documents ont un attribut «N° de version» : les anciens types de documents devront rester disponibles afin de pouvoir traiter les documents qui n'ont pas encore adopté la nouvelle structure.

C. Une nouvelle disposition légale impose le changement d'un type de formulaire.

Si une nouvelle disposition légale impose le changement d'un type de formulaire, nous nous retrouverons dans une situation similaire au cas précédent. Néanmoins, nous pourrions offrir en plus une fonctionnalité supplémentaire permettant d'avertir les acteurs concernés par ces changements. Nous pourrons également faire appel aux fonctionnalités de gestion du temps (voir chapitre suivant) afin de gérer un éventuel «flag day», date clé où tout le monde devra se conformer aux nouveaux formulaires.

D. Un document n'a pas de structure définie.

Si un document n'a pas de structure définie, nous ne pourrons que le considérer que comme une entité fixe, comme un fichier non-manipulable. Nous pouvons considérer ce genre de document comme un formulaire monolithe, avec un seul champ «contenu».

E. Une tâche non-prédéfinie envoie un résultat.

Le cas où une tâche non-prédéfinie envoie un résultat sera examiné dans le chapitre suivant sur la gestion du flux.

F. Un nouvel enchaînement de tâches est décidé à l'exécution.

Le cas où un nouvel enchaînement de tâches serait décidé à l'exécution sera examiné dans le chapitre suivant sur la gestion du flux.

Chapitre IV - Gestion du flux

Dans le chapitre précédent, nous avons spécifié les informations et types d'informations, ce qui nous a aussi permis de spécifier les types de tâches. Nous avions aussi indiqué dans les chapitres 1 et 2 que nous divisions le travail (procédure) en différentes tâches. Mais il ne s'agit pas d'un simple "saucissonnage", et il nous faut à présent modéliser les enchaînements entre tâches. En d'autres termes, nous allons à présent spécifier le flux, c'est-à-dire caractériser les "linkages" entre tâches présentés dans les modèles W3Flow [II.3.].

Nous allons commencer par voir ce qu'il est nécessaire de modéliser dans un type de flux de travail (IV.1.). Ensuite, nous étudierons les particularités des enchaînements ad-hoc (IV.2.). Puis, nous présenterons une modélisation existante tirée de la littérature (IV.3.). Cette modélisation ne nous conviendra toutefois pas, et nous en proposerons une autre pour spécifier les flux (IV.4.). Enfin, nous synthétiserons les éléments de ce modèle dans une perspective d'exécution du flux de travail (IV.5.)

IV.1. Que modéliser dans le flux

Pour caractériser et spécifier un type de flux (informationnel), il est important de discerner ce qui doit être prédéfini et ce qui est décidé à l'exécution. La figure qui suit illustre à ce propos des échanges d'informations entre acteurs dans notre contexte permis de bâtir[13].

[13] = les messages relatifs à la conversation qui s'instaure lors d'échanges d'informations ne sont pas abordés ici.

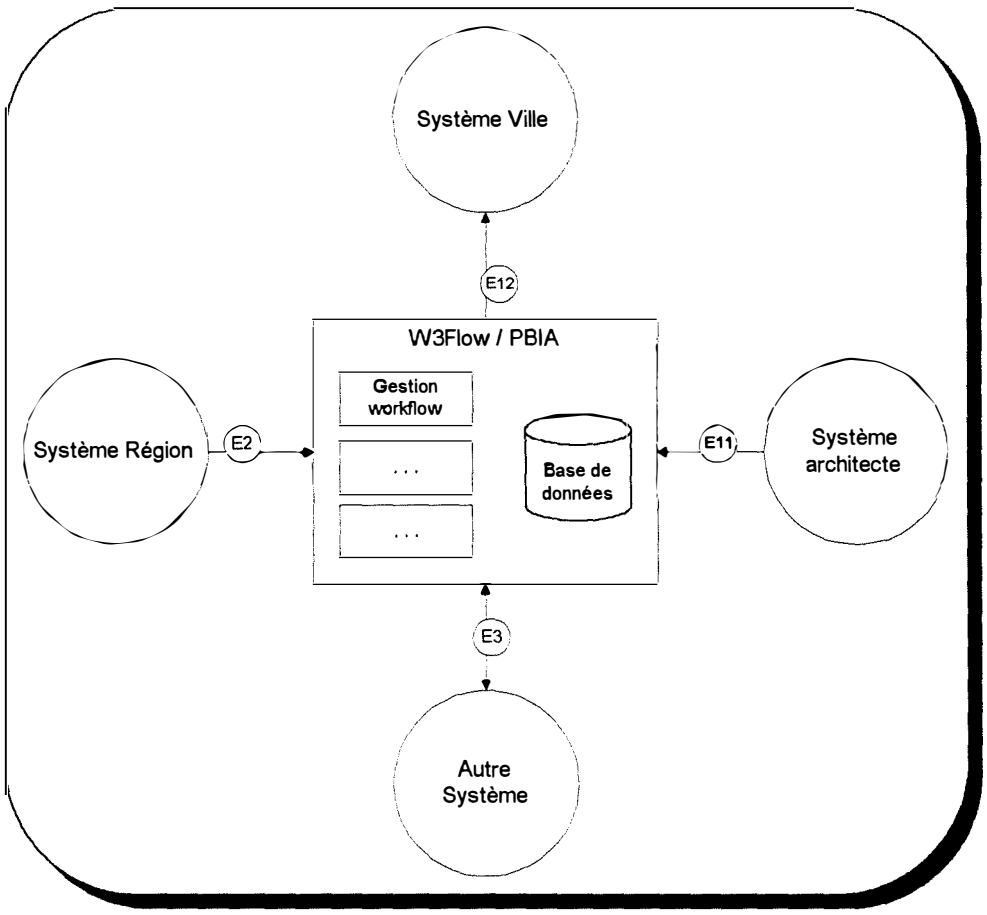


Figure IV-1 : Illustration de différents échanges d'informations dans PBFlow

Deux types d'échanges d'informations peuvent être observés sur cette figure; il y a:

1. *les échanges d'informations directement liés à un flux de travail particulier* : ce sont les entrées/sorties de tâches. Généralement parlant, il s'agit d'échanges d'informations entre deux acteurs différents, lequel échange peut être prédéterminé.
2. *les échanges d'informations non-directement liés à un flux de travail particulier* : ce sont des échanges d'informations intervenant pendant la réalisation d'une tâche, et à la requête expresse d'un acteur. Généralement parlant, il s'agit d'une demande d'information non-prédéfinie, impliquant un acteur et le système PBFlow ou un tiers.

Des exemples pour chacun de ces deux cas est illustré sur la figure IV-1 :

1. *Cas lié à un flux particulier*: l'architecte envoie sa demande de permis de bâtir au système PBFlow (=E11), lequel fait parvenir le dossier au service d'administration de l'urbanisme de la ville (=E12).
2. *Cas non-lié à un flux particulier*: la région instaure un nouveau plan particulier d'aménagement (=PPA) et l'envoie au système PBFlow (=E2). Ou encore, un autre acteur consulte de sa propre initiative des données dans la base de données PBFlow (=E3).

Maintenant, quels sont les fonctionnalités principales W3Flow mises-en-oeuvre dans les deux cas d'échanges d'informations ?

1. dans le cas «lié au flux», ce sont la gestion du flux (workflow) et l'interfaçage;
2. dans le cas «non-lié au flux», ce sont la sécurité (droit d'accès) et l'interfaçage.

Le problème de l'interfaçage a déjà été abordé dans le chapitre précédent. Ce qui nous intéresse maintenant, c'est la modélisation du flux prédéfinis. Cependant, nous ne pourrons pas imposer un flux totalement prédéfinis; nous devons laisser certaines libertés aux acteurs, leur permettant de déterminer le flux eux-mêmes.

IV.2. Flux prédéfini et ad-hoc

Des systèmes de workflow tels que *The Milano System* [] sont essentiellement axé vers des tâches structurées, des situations prédéfinies. Ceci signifie que, lorsqu'une tâche est terminée, le système de workflow sait exactement ce qu'il a à faire : quelle est la tâche suivante, qui doit l'exécuter, etc. On parle alors de flux prédéfini [chap.I].

Notre cas permis de bâtir est quelque peu différent : les procédures sont largement prédéfinies, mais pas exclusivement. Nous devons donc introduire un peu de souplesse dans notre système, et permettre aux acteurs d'intervenir sur la gestion du flux. Plusieurs cas sont à considérer; nous allons les examiner un à un.

IV.2.1. Enchaînement par défaut (prédéfini)

Notre système n'imposera pas d'enchaînement aux acteurs, il leur proposera l'enchaînement calculé par notre système de workflow. Voici, sur cette figure IV-2, un exemple de ce que cela pourrait donner pour un acteur.

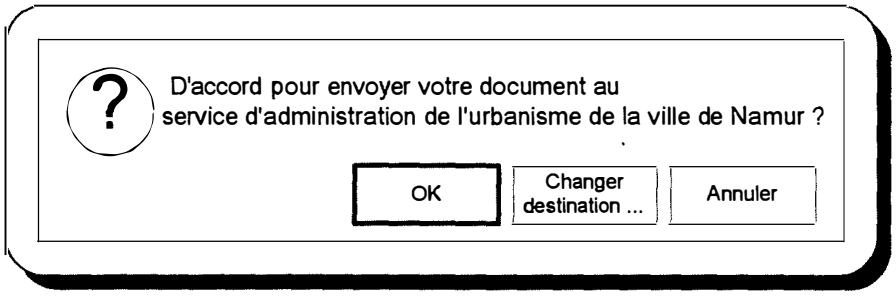


Figure IV-2 : Exemple d'enchaînement par défaut

Lorsque notre système PBFlow peut déterminer de lui-même, grâce aux spécifications prédéfinies du flux, les enchaînements de tâches, il les proposera par défaut, à l'utilisateur. Celui-ci aura néanmoins toujours la possibilité de modifier cet enchaînement prédéfini.

Lorsque notre système PBFlow ne peut déterminer un enchaînement de tâches de lui-même, ce sera aux acteurs à préciser eux-mêmes l'enchaînement, à l'exécution.

IV.2.2. Reprise (ad-hoc) d'une tâche en amont (prédéfinie)

Une première situation non-prédéfinie qui peut se présenter est une reprise en amont dans le flux de travail. Ce cas peut notamment se produire lorsqu'un acteur constate une anomalie ou un manquement dans l'exécution d'une tâche précédente. Il est donc nécessaire de recommencer l'exécution d'une tâche précédemment exécutée.

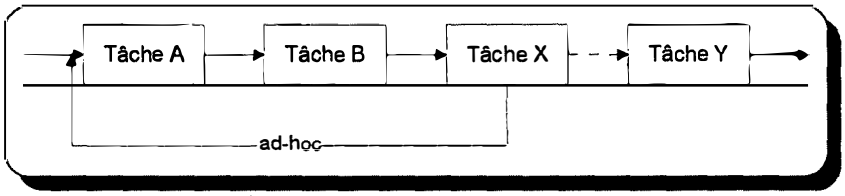


Figure IV-3: Exemple de reprise ad-hoc d'une tâche en amont

La figure IV-3 ci-dessus illustre un exemple: lors de l'exécution d'une tâche X, un acteur décide de reprendre (à) la tâche A[14].

Mais il est nécessaire de pousser notre réflexion plus loin : que devient l'enchaînement prédéfini du flux ? Que devient la tâche X ? Deux solutions sont envisageables.

1. Soit la tâche X est suspendue, et la tâche A est relancée. Lorsque l'exécution de la tâche A sera achevée, la tâche X sera réactivée, et le flux prédéfini reprendra en ce point-là. Les tâches intermédiaires (ici B) ne seront pas réexécutées.
2. Soit la tâche X est abandonnée, et l'exécution du flux reprend au niveau de la tâche A. Lorsque l'exécution de la tâche A sera achevée, le flux prédéfini reprendra à ce point-là. Dans notre exemple, B est X seront ensuite (ré)exécutés.

Le choix entre l'une ou l'autre de ces deux possibilités sera donnée aux acteurs, lors de l'exécution du flux. Notons qu'au niveau des entrées/sorties de tâches, rien n'a changé dans ce cas-ci.

IV.2.3. Enchaînement (ad-hoc) vers une autre tâche (prédéfinie)

Après l'exécution d'une tâche, un acteur peut décider lui-même de la suite du déroulement de la procédure. En effet, il ne sera pas toujours possible de formaliser tous

[14] à nouveau, nous passons ici sous silence la conversation (pour action) qui aura lieu.

les enchaînements possibles de tâches, la réalité étant parfois fort complexe. Nous supposons toutefois que les différents types de tâches ont tous été prédéfinis.

Ce que nous voulons faire, c'est donner à notre système PBFlow une souplesse permettant aux acteurs, en cours d'exécution, de décider eux-mêmes quelle est la tâche suivante à exécuter dans une situation particulière, comme l'illustre la figure IV-4 qui suit.

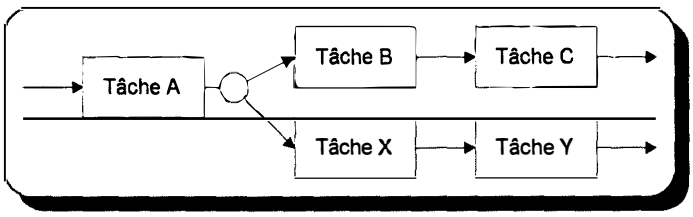


Figure IV-4: Exemple d'enchaînement ad-hoc de tâches prédéfinies

Le cas illustré sur la figure IV-4 se présente soit lorsqu'un acteur décide de changer l'enchaînement prédéfini (par exemple A-B-C-...), soit lorsque le système PBFlow n'a pas les éléments suffisants que pour décider lui-même de l'étape suivante. En tous les cas, le flux suivra ultérieurement son chemin prédéfini, comme indiqué sur la figure.

Un autre apport que peut permettre notre système dans ce cas particulier, est de proposer aux acteurs une liste avec les enchaînements qu'ils choisissent généralement, classés par ordre décroissant de fréquence d'enchaînement.

Précisément au niveau de l'enchaînement, et contrairement au cas précédent, il peut se poser un problème au niveau des entrées/sorties de documents. Par exemple, il n'a pas été prévu que la tâche X succède à la tâche A. Dès lors, que deviennent les documents en entrées, nécessaires à l'exécution de la tâche X ?

Pour répondre à ce problème, nous proposons une fonctionnalité supplémentaire pour aider les acteurs et supporter ces situations ad-hoc. Comme les types de tâches sont toutes prédéfinies, en ce compris leurs types d'entrées/sorties, il est possible de dresser la liste des documents disponibles et des documents manquants pour l'exécution de la tâche X choisie.

Liste des documents nécessaires pour l'exécution de la tâche X:

- ☑ demande de permis de bâtir
- ☑ attestation de l'ordre des architectes
- ❑ formulaire CWATUP Annexe 17 (manquant)
- ❑ plan d'exploitation (manquant)

Figure IV-5: Exemple de liste de vérification des documents disponibles

Cette liste, présentée aux acteurs (figure IV-5), les guidera dans leur choix d'un enchaînement de tâches : ils pourront ainsi constater la disponibilité ou l'indisponibilité de documents essentiels, et décider sur base de cette information de l'enchaînement. Ils pourront également rajouter d'autres documents complémentaires, qui n'étaient pas spécifiés comme entrées du type de tâche X.

Nous pouvons également proposer cette fonctionnalité pour tous les types d'enchaînements prédéfinis, et de proposer ces listes tant en sortie qu'en entrée de tâche. Elle permettra alors aux acteurs de voir quels documents ils doivent encore fournir.

IV.2.4. Enchaînement vers une tâche non-prédéfinie

Le dernier cas qu'il nous reste à examiner est celui où un acteur désire ou doit lancer l'exécution d'une tâche qui n'est définie nulle part. Il est évident que, la tâche suivante étant indéfinie, l'enchaînement de tâches sera lui aussi indéfini.

Bref, un acteur confiera à un tiers l'exécution d'une tâche non-prédéfinie[15] : cette tâche, non-prédéfinie, s'inscrit dans un contexte de négociation : le tiers pourra par exemple refuser d'exécuter cette tâche.

Le fait que cette tâche, ainsi que l'enchaînement, ne soient pas prédéfinis pose deux types de questions :

1. que deviendra le flux, après l'exécution de cette tâche ad-hoc ? Peut-on en laisser la responsabilité à l'exécutant de la tâche ad-hoc (non-prédéfinie) ?
2. que deviendront les entrées/sorties de tâches, et quelle sera la responsabilité de l'exécutant de la tâche non-prédéfinie à cet égard ?

Notre analyse de ce problème nous a conduite à distinguer deux cas : primo, celui où la tâche ad-hoc est exécutée *pendant* l'exécution d'une autre tâche, et secundo celui où la tâche ad-hoc est exécutée *après* l'exécution d'une autre tâche.

Examinons le premier cas présenté à la figure IV-6. Pendant l'exécution de la tâche A, l'acteur exécutant cette tâche décide de faire appel à un tiers afin qu'il exécute un travail pour lui (la tâche Y), dans le cadre de l'exécution de cette tâche A. Il s'agit donc d'une tâche complémentaire à une autre, que certains auteurs appellent tâche greffée [THP].

[15] à nouveau, nous passons ici sous silence la conversation (pour action) qui aura lieu.

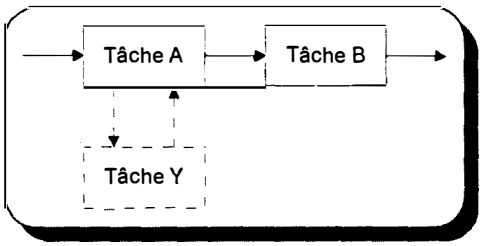


Figure IV-6: Insertion d'une tâche ad-hoc complémentaire

Nous nous ramenons alors à un cas similaire à celui de la reprise ad-hoc et en amont du flux [IV.2.2.] : une fois la tâche Y exécutée, l'exécution du flux reprendra au niveau de la tâche A. Dans ce cas-ci également, notre système PBFlow n'aura pas à ce soucier des entrées/sorties de tâches :

- pour la tâche A, rien n'est modifié au niveau des entrées/sorties d'informations avec les tâches suivantes et précédentes;
- pour la tâche Y, l'entrée d'information est ce que l'acteur exécutant de A lui aura envoyé, et l'intégralité des documents produits après l'exécution de la tâche Y reviendront à l'acteur exécutant la tâche A.

Il y a cependant une différence par rapport au cas d'une reprise en amont du flux : la tâche A ne doit pas nécessairement être suspendue. Lorsqu'un acteur lancera l'exécution d'une tâche non-prédéfinie, nous lui proposerons donc deux choix :

1. *suspendre* l'exécution de la tâche en cours jusqu'à ce que la tâche non-prédéfinie soit terminée;
2. *continuer* l'exécution de la tâche en cours, parallèlement à celle de la tâche non-prédéfinie.

Nous imposons toutefois que l'exécution de la tâche A ne puisse se terminer avant celle de la tâche non-prédéfinie (Y). Cette condition est nécessaire pour pouvoir assurer que dans tous les cas, lorsque l'exécution de la tâche ad-hoc est terminée, le résultat de cette tâche peut être pris en charge par PBFlow. Pratiquement, les résultats fournis par la tâche ad-hoc complémentaire (Y) reviendront à l'acteur exécutant la tâche A, acteur qui a voulu l'exécution de cette tâche ad-hoc. Il faut donc bien dans ce cas que la tâche A ne puisse être achevée.

Le deuxième cas à examiner est celui où l'exécution de la tâche ad-hoc a lieu postérieurement à celle de la tâche prédéfinie (la tâche *A* dans notre exemple). Il s'agit alors d'une tâche supplémentaire, qui s'insère dans le flux de tâches, et non plus au niveau d'une tâche. La figure IV-7 ci-dessous présente schématiquement les deux possibilités que nous envisageons pour le cas présent.

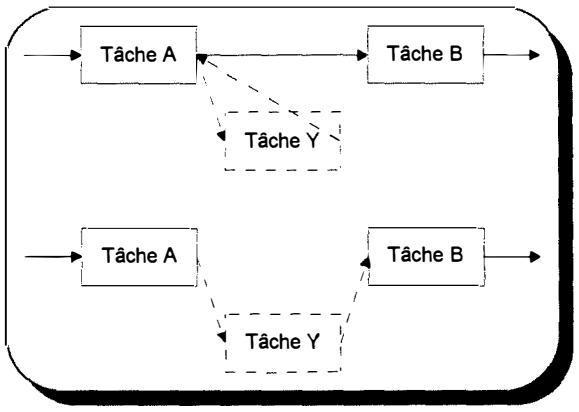


Figure IV-7: Insertion d'une tâche ad-hoc supplémentaire au sein d'un flux

Tout d'abord, on peut adopter une optique similaire à celle du cas précédent: l'exécutant de la tâche ad-hoc (*Y*) n'est pas responsable de la continuation du flux, et après l'exécution de sa tâche, ce sera à l'acteur de la tâche prédéfinie (A) à reprendre l'exécution du flux, où à laisser la main au système PBFlow.

Mais nous laisserons aussi la possibilité à l'acteur exécutant la tâche prédéfinie (A) de léguer sa responsabilité dans la gestion du flux à celui qui exécutera la tâche ad-hoc supplémentaire. C'est ce qu'illustre la seconde partie de la figure IV-7.

Le mécanisme proposé à l'acteur exécutant la tâche ad-hoc sera alors similaire au cas de l'enchaînement (ad-hoc) vers une autre tâche prédéfinie [IV.2.3.] : une liste des tâches suivantes lui sera proposée, avec correspondance des entrées/sorties. Il choisira alors une tâche parmi celles proposées.

S'il ne trouve aucune tâche prédéfinie qui lui convienne, notre système PBFlow agira alors comme si l'acteur A n'avait pas légué sa responsabilité dans la gestion du flux : ce sera à l'acteur A, commanditaire, de continuer à gérer le flux. Celui-ci pourra alors éventuellement lancer une nouvelle tâche supplémentaire selon le même mécanisme.

Nous n'envisageons donc pas de laisser la possibilité d'enchaîner plusieurs tâches non-prédéfinies, par des acteurs indéfinis. Ceci nous orienterait vers un système de workflow complètement différent, et peu approprié à notre cas permis de bâtir. Car n'oublions pas que la procédure de permis de bâtir, ainsi que la plupart des procédures administratives, sont largement (mais non-exclusivement) prédéfinies.

IV.3. Modélisation des enchaînements de tâches (état de l'art)

Nous avons vu *ce qu'*il faut modéliser; il reste à savoir *comment*. Il y a premièrement les composantes statiques du flux de travail qu'il nous faut décrire : les tâches, les documents (entrées/sorties de tâches), les postes de travail (réceptacles des entrées/sorties de documents), et l'inter-organisation. Sauf pour la modélisation de l'inter-organisation, nous en avons déjà assez parlé. En second lieu, il y a la modélisation des enchaînements, dynamiques.

A ce propos, nous avons trouvé très intéressant le modèle IDA *(Interactive Design Approach)* de la dynamique des traitements [BOD89, pg.70-91] : « ***Ce modèle est principalement destiné à représenter, et donc à spécifier, les conditions d'exécution et d'enchaînement des traitements en vue de caractériser le comportement d'un système d'information.*** »

Ce modèle est un modèle causal qui s'appuie sur les concepts d'événement et de processus : « ***Un événement représente un changement d'état qui survient à un moment donné de l'évolution du système d'information et qui correspond à un stimulus auquel ce système doit réagir, principalement par le déclenchement de certains processus.*** » [BOD89, pg.72-73]

Sans entrer dans les détails, citons deux événements remarquables : le déclenchement, et la terminaison d'un processus (*tâche* dans notre contexte).

Ce modèle identifie aussi différents types d'enchaînements primitifs, pouvant être combinés entre eux. Un type d'enchaînement primitif peut être :

- *séquentiel* : lorsqu'une tâche A achève son exécution, une autre tâche B commence;
- *parallèle* : lorsqu'une tâche A achève son exécution, plusieurs autres tâches B commencent;
- *multiple* : lorsqu'une tâche A achève son exécution, plusieurs instances d'une seule autre tâche B commencent (il s'agit donc d'un cas particulier de l'enchaînement parallèle);
- *convergent* : l'exécution d'une B commence lorsque l'exécution de l'une OU l'autre tâche déterminée termine.
- *synchrone* : l'exécution d'une B commence lorsque l'exécution de l'une ET l'autre tâche déterminée termine.
- *conditionnel* : l'exécution d'une tâche A provoque sélectivement le déclenchement d'un autre processus en fonction d'une condition.

Pour réaliser ces types d'enchaînements, la modèle de la dynamique des traitements de la méthode IDA fait appel à différents mécanismes :

- *un mécanisme de synchronisation*, agissant comme un mécanisme de coordination d'événements qui permet de construire un nouvel événement à partir d'autres plus élémentaires;
- *un mécanisme de sélection*, permettant de sélectionner - sur base d'une condition portant sur l'état du système - la réaction appropriée à la survenance d'un événement;
- *un mécanisme de duplication*, permettant à un seul événement de contribuer au déclenchement de plusieurs processus.

Ces mécanismes peuvent en outre être combinés entre eux. Par exemple dans le cas du mécanisme de sélection, la réaction appropriée à la survenance d'un événement peut être la contribution à une synchronisation. La figure IV-8 qui suit l'illustre.

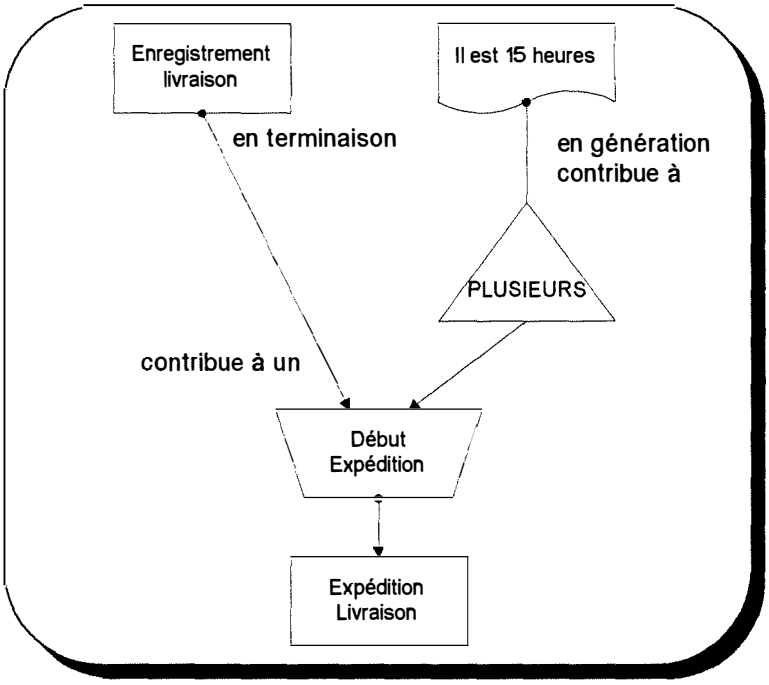


Figure IV-8: Exemple de représentation graphique - Modèle de la dynamique des traitements [BOD89, pg.87]

Sur cette figure, nous pouvons observer deux tâches *(enregistrement et expédition livraison)*, un événement *(il est 15 heures)*, ainsi que deux mécanismes, l'un de duplication *(plusieurs)* et l'autre de synchronisation *(début expédition)*.

IV.4. Spécification du flux (nouvelle approche)

IV.4.1. La nécessité d'un nouveau modèle

Le modèle IDA de la dynamique des traitements [IV.3], pris tel quel, ne nous convient pas. Il est en effet mal adapté dans notre contexte de gestion de permis de bâtir.

Premièrement, ce modèle est trop rigide pour nous : il s'applique bien à des flux prédéfinis, mais nous voulons concevoir un système plus souple. En particulier, nous ne pourrons pas toujours spécifier complètement les enchaînements de tâches, et a fortiori les mécanismes utilisés. Nous préférerons donc une modélisation ne faisant pas appel à des mécanismes externes de gestion du flux : nous préférerons une modélisation où la gestion du flux sera définie au niveau des tâches en elles-mêmes.

Ensuite, ce modèle IDA de la dynamique des traitements s'insère mal dans notre système de workflow inter-organisationnel (PBFlow). Notamment, il ne tient pas compte de notre problématique d'entrées/sorties de tâches et d'interfaçage. Or, notre gestion du flux fait également intervenir les documents.

Enfin, le modèle IDA a été élaboré dans un but de spécification du flux, qui ne débouche pas nécessairement sur quelque chose d'opérationnel. Ce qu'il nous faudrait, c'est un moyen de spécification fonctionnel tel que nous puissions ensuite rendre ces spécifications exécutables.

Néanmoins, ce modèle de la dynamique des traitements, extrait de la méthode IDA *(Interactive Design Approach)* nous a inspiré lors la conception du nouveau modèle que nous proposons ci-après.

IV.4.2. Comportement du système de workflow

Nous modélisons le comportement de notre système de workflow par un un ensemble de triplets *stimulus - décision - action*. Le système de workflow réagit suite à des stimuli (les événements), qui l'incitent à prendre des décisions (sur base d'informations), qui se traduiront ensuite en actions. Ces triplets, nous les appellerons ultérieurement *clauses*.

Ces triplets interviennent dans la dynamique du flux entre l'exécution de tâches, comme l'illustre la figure suivante.

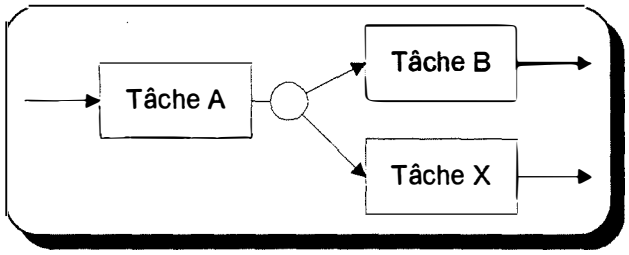


Figure IV-9: Localisation d'un triplet stimulus-décision-réaction

La figure IV-9 illustre l'endroit, le moment central d'un enchaînement (le cercle):

1. Un événement survient : la terminaison de la tâche A; c'est le stimulus. Nous pouvons également dire que la survenance de l'événement "terminaison de la tâche A" est la condition de déclenchement.
2. Une décision est prise pour l'enchaînement des tâches: quelle sera la ou les tâches suivante(s) à exécuter ?
3. Une action est prise; dans notre exemple, les tâches B et X sont lancées.

Si nous voulons maintenant caractériser les différents types d'enchaînements en fonction de critères, deux optiques différentes se posent à nous, deux types d'enchaînements :

- *les enchaînements "post"* tout d'abord, qui sont effectifs à la sortie d'une tâche; la question posée est donc : "lorsqu'une tâche déterminée est en terminaison d'exécution, quel(s) est/sont les tâches suivantes à effectuer ?". Ainsi en est-il des enchaînements séquentiels, parallèles, etc.
- *les enchaînements "pré"* ensuite, qui sont effectifs à l'entrée d'une tâche; la question posée est donc : "quand peut-on exécuter démarrer l'exécution d'une tâche déterminée ?". Ainsi en est-il des enchaînements synchrones.

Comme nous pourrons le constater, certains enchaînements sont uniquement de type post (ex.: enchaînement parallèle), ou uniquement de type pré (ex.: enchaînement synchrone), alors que d'autres peuvent être post/pré selon les cas (ex: enchaînement conditionnel).

Cette distinction pré/post est intéressante car elle nous permettra de modéliser les flux. Mais pour l'instant, il nous manque encore un élément essentiel pour caractériser les flux : les clauses (ou modalités).

IV.4.3. Le concept de clause

Dans cette section, nous allons définir le concept de clause (4.3.1.). Nous verrons alors qu'il y a deux types de clauses : les clauses post (4.3.2.), et les clauses pré (4.3.3.). Nous comparerons cette modélisation avec celle d'IDA (4.3.4.) et nous distinguerons bien notre concept de clause de ceux de pré/post-conditions (4.3.5.). Il nous restera alors à voir comment spécifier une clause (IV.4.4.).

4.3.1. Définition d'une clause

Les clauses interviennent dans la régulation de flux. Sous notre concept de clause, nous regroupons trois éléments :

- d'une part, *le prédicat de la clause*, qui est un stimulus, c'est la condition qui, si elle est vérifiée, activera la clause; elle dit QUAND agir;
- d'autre part, un ensemble de couples condition-action (qu'on pourrait assimiler à un "point de décision" dans le flux de travail) :
  • *la condition* (ou expression décisionnelle), facultative, est une expression dont l'évaluation débouchera sur une décision (d'agir); elle dit QUOI faire;
  • *les (ré)actions à prendre* en fonction du résultat de l'évaluation de la condition; elle dit COMMENT agir.

Si nous préférons le nom de "clause" plutôt que de "condition", c'est parce que notre concept est plus vaste que la notion de "condition" dans un enchaînement conditionnel, et distinct des pré/post-conditions. Ainsi, dans notre modélisation, une synchronisation sera représentée par une clause (pré).

Les deux paragraphes suivants présentes les spécificités de ceux deux types de clauses.

4.3.2. Les clauses POST

Aux clauses POST correspondent entre autres les mécanismes IDA de sélection et de duplication comme l'illustre la figure IV-10.

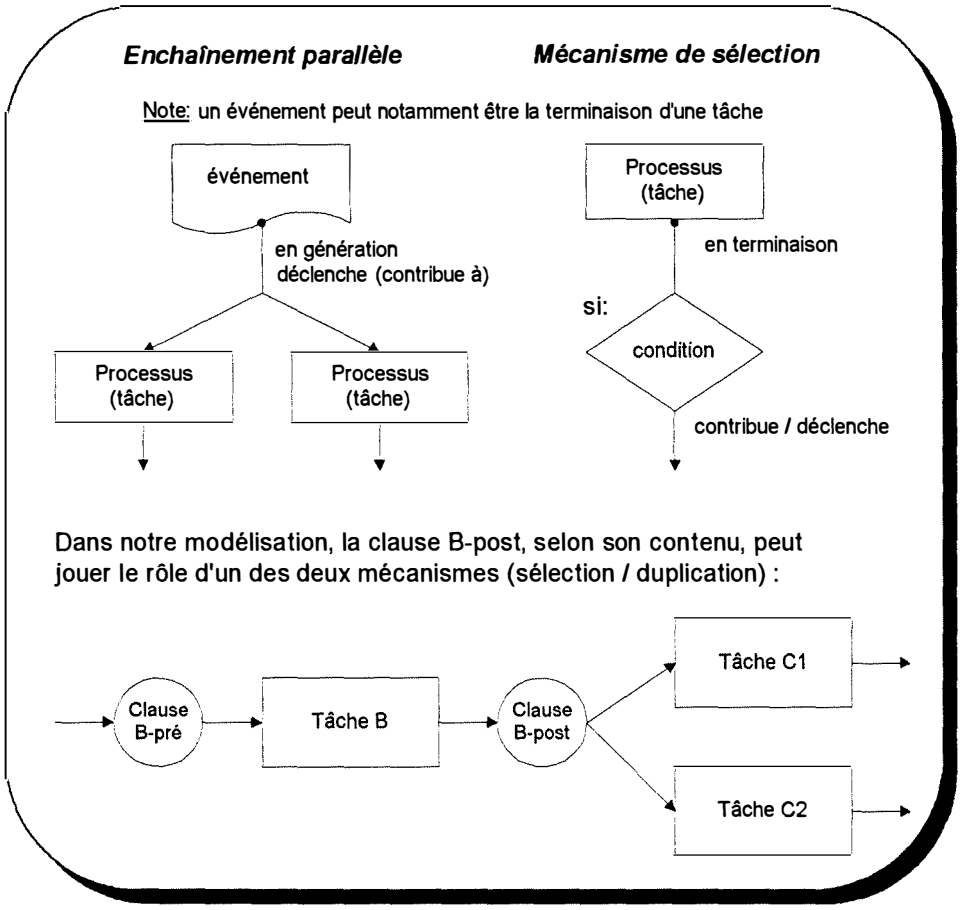


Figure IV-10: Exemple de clause post et comparaison avec IDA

La clause B-post de la figure IV-10 peut modéliser tout aussi bien un enchaînement conditionnel (alternatif) ou que parallèle. La clause B-post pourrait par exemple s'énoncer comme suit :

- condition de déclenchement (*prédicat*) : terminaison de la tâche B;
- expression décisionnelle : "Quelle réponse a fournit la tâche B ?" (vrai ou faux)
- réaction à prendre: vrai ⇒ exécuter les tâches C1 et C2; faux ⇒ exécuter C2.

Ce type d'enchaînement, très classique, est du type "post". En d'autres termes, la condition est évaluée postérieurement à l'exécution d'une tâche déterminée (ici B), en fonction du seul résultat fourni par cette tâche. Nous pouvons donc dire qu'il y a un prédicat implicite aux clauses POST d'une tâche donnée : la terminaison de cette tâche.

4.3.3. Les clauses PRE

Voici à présent un exemple d'enchaînement du type "pré", correspondant au mécanisme de synchronisation (IDA).

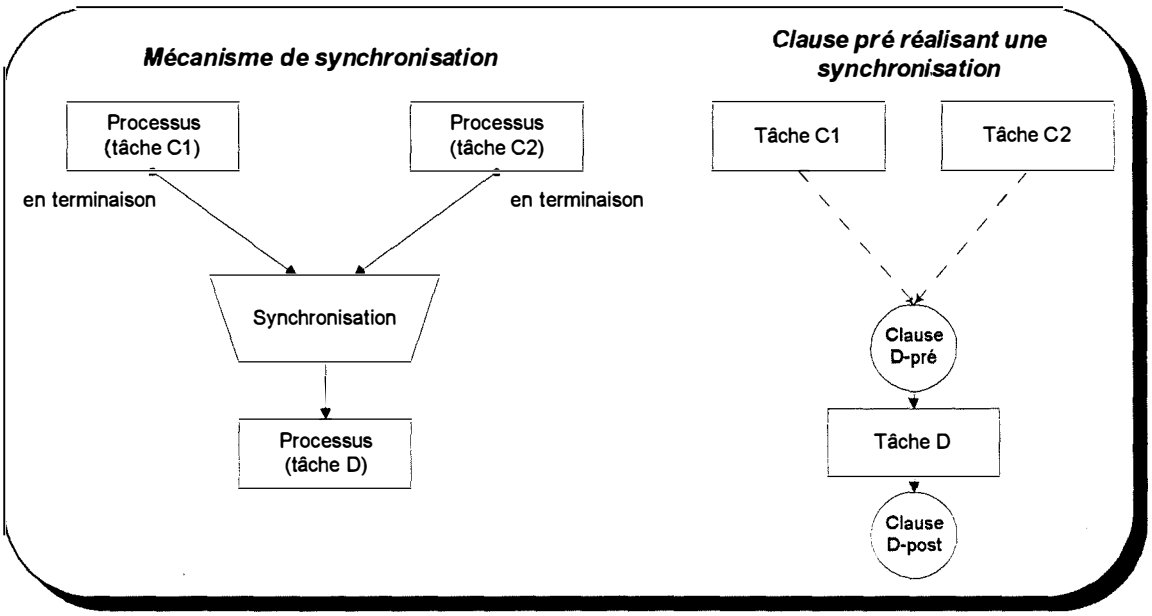


Figure IV-11: Exemple de clause pré (synchronisation) et son équivalent IDA.

La clause D-pré de la figure IV-11 modélise ici une synchronisation :

- condition de déclenchement (*prédicat*) : terminaison des tâches C1 et C2[16];
- expression décisionnelle : (aucune)
- réaction à prendre : exécuter la tâche D.

Ce type d'enchaînement, très classique, est du type "pré". En d'autres termes, la condition est évaluée antérieurement à l'exécution d'une tâche déterminée (ici D). Dans ce cas, c'est la réaction à prendre qui est implicite : exécuter la tâche associée à la clause (pré).

4.3.4. Comparaison avec IDA

La différence essentielle entre notre modélisation et celle d'IDA est que nous ne faisons appel à aucun mécanisme externe particulier. En fait, nous modélisons tous les enchaînements au niveau des tâches, dans les clauses qui leur sont associées. Les enchaînements sont donc représentés de manière moins explicite.

Ce qu'il nous reste encore à voir, c'est de quelle manière nous allons spécifier nos clauses, et ce qu'elles vont pouvoir contenir comme conditions, expressions et actions.

[16] une conjonction d'événements simultanés serait vraiment exceptionnelle : en fait notre système *attend* que tous les événements de la conjonction aient eu lieu, et il réagira alors.

4.3.5. Conditions et clauses

Si les concepts de pré-condition et de clause-pré coïncident, il est toutefois très important de bien distinguer notre "clause post" de la notion de "post condition" en algorithmie classique. En algorithmie, la post-condition d'une procédure (au sens du Pascal par exemple) n'est pas à proprement parler une *condition*, mais bien ce à quoi les résultats fournis suite à l'exécution de la procédure DOIVENT se conformer. La post-condition est donc une réelle obligation.

En outre, il ne faut pas confondre les propriétés des résultats produits (= post-condition), c'est-à-dire les sorties informationnelles d'une tâche, avec la clause post. La clause post n'est pas orientée vers le passé, mais bien vers l'avenir : elle sert à déterminer ce qui va se passer ensuite, quitte à ce que l'avenir soit de recommencer la tâche qui vient d'être exécutée.

Enfin, tant pour la clause pré que post, *le référentiel est le flux de travail*, et non la tâche prise isolément. En algorithmie par contre, pré- et post-conditions se réfèrent uniquement à l'algorithme (ou procédure Pascal) auquel elles se rattachent.

En d'autres termes, on peut rattacher la notion de pré-condition à celle d'*entrées* d'un type de tâche, et celle de post-condition aux *sorties* d'un type de tâche.

IV.4.4. Spécification d'une clause

Pour rappel, trois éléments sont à spécifier dans une clause :

- le prédicat, ou condition de déclenchement (implicite pour une clause post) [4.4.1.];
- l'expression décisionnelle (facultativement) [4.4.2.];
- la ou les réaction(s) à prendre (implicite pour une clause pré) [4.4.3.].

4.4.1. Spécification d'un prédicat

Le prédicat d'une clause est la condition de déclenchement, le stimulus, qui va faire réagir le système de workflow. La spécification d'un prédicat reposera donc sur le concept d'événement.

événement

Un événement représente un changement d'état, qui survient à un moment donné de l'évolution d'un objet (tâche ou document). Cet événement agit comme un stimulus, auquel le système de workflow peut réagir, notamment par le déclenchement de tâches.

état

L'état d'un objet (tâche ou document) rend compte du dernier événement survenu le concernant.

Par exemple, un objet dans l'état inexistant passera à l'état «créé» après la survenance de l'événement de «création».

Pour une tâche, les différents événements que nous prenons en compte pour notre système PBFlow sont :

- lancement d'une tâche (création d'une instance de tâche);
- terminaison (normale) d'une tâche;
- suspension d'une tâche;
- reprise d'une tâche;
- annulation d'une tâche (terminaison anormale).

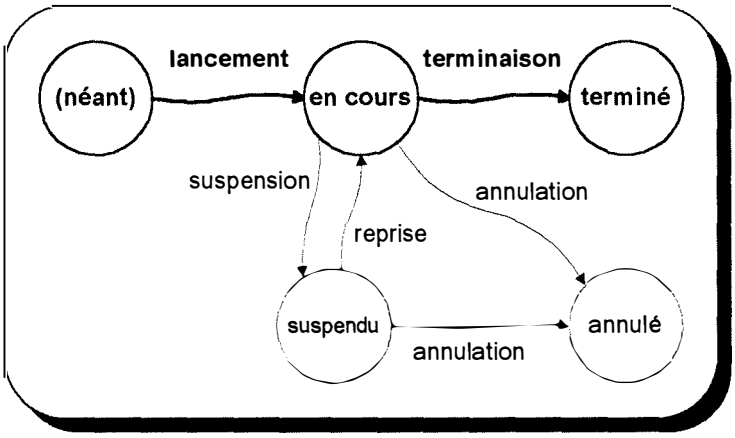


Figure IV-12 : Etats et événements (liens) de la vie d'une tâche

La figure IV-12 qui précède montre le cycle de vie d'une tâche : les différents événements, leurs états correspondants, ainsi que les transitions possibles.

Pour un document, seuls deux événements nous intéressent :

- création d'un document (sortie de tâche);
- destruction d'un document (cas anormal, annulation d'un document).

Deux événements complémentaires, relatifs à l'association document-partenaire peuvent être rajouté :

- envoi d'un document (depuis un poste de travail);
- réception d'un document (sur un poste de travail).

Remarquons que, notre système de workflow étant inter-organisationnel, nous ne voyons les documents que comme entrées et sorties de tâches. Nous ne savons pas ce que deviennent les documents une fois reçus par les postes de travail. Des événements tels que *«ouverture d'un document en modification»* ou *«fermeture d'un document»* ne nous intéressent donc pas. En fait, les documents ne sont jamais modifiés par le système PBFlow[17].

Notre système PBFlow étant inter-organisationnel et centralisé, nous n'avons pas non plus besoin de mécanismes de réplication, de gestion de versions de documents, etc. [voir chapitre 2, et notamment la comparaison avec Lotus Notes]. C'est aussi pourquoi nous considérons si peu d'événements concernant les documents.

En résumé, le prédicat d'une clause est défini sur des événements. Nous proposons la syntaxe suivante.

```
<prédicat>    ::= <proposition>
<proposition>::= <événement> | '[' <proposition> <op> <proposition>']'
<op>          ::= {'ET'|'OU'}
<événement>   ::= {'lancement'|'terminaison'|'suspension'|'reprise'|
                  'annulation'} '(T:'<tâche>')' |
                  {'création'|'destruction'} '(D:'<document>')' |
                  {'envoi'|'réception'} '(DP:'<document.partenaire>')'
```

Figure IV-13: Syntaxe de définition d'un prédicat de clause

Un prédicat de clause est soit un événement, soit une combinaison d'événements. Cette combinaison d'événements peut être une conjonction (ET) ou disjonction (OU). Il est sous-entendu que les événements concernant les tâches et documents concernent le même flux (le même dossier).

Exemple de prédicat (implicite) pour une clause post :

- terminaison (T:examen dossier)
  Lorsque la tâche «examen dossier» est terminée, la clause est déclenchée.

Exemple de prédicat pour une clause pré :

- [[terminaison(T:avis STU) ET terminaison(T:avis SI)] OU envoi(DP:dérogation.ministre)]
  Lorsque, ou les tâches d'élaboration d'avis STU et SI sont terminées, ou que le ministre a envoyé une dérogation, la clause est déclenchée.

[17] la fonctionnalité d'interfaçage n'étant qu'une conversion, transparente pour les partenaires.

4.4.2. Spécification d'une expression

Une expression décisionnelle d'une clause est le coeur de la décision concernant le flux. Cette expression s'insère dans un modèle causal *condition* ⇒ *action*. Ce que nous spécifions ici, sont les conditions.

Néanmoins, cette condition pourra être vide dans certaines circonstances : c'est le cas des enchaînements séquentiels, parfaitement prédéfinis. Dans ces cas-là, il n'y aura qu'une seule action possible, toujours la même, et donc il est inutile de spécifier une condition (prédicat et action unique suffisent).

Qu'est-ce qui peut, à l'exécution, déterminer le choix à prendre dans le flux de travail, comment déterminer la ou les tâche(s) suivante(s) ?

Cette décision peut s'opérer sur base de différents éléments :

- *des états*, relatifs à des tâches ou documents du même flux de travail;
- *des informations*, contenues dans des documents (relatives ou non au même flux) ou internes au système (ex.: la date).

Notons la différence avec le prédicat d'une clause : la décision se prend sur base d'états (stables), et non d'événements (furtifs). Nous proposons la syntaxe de définition que voici à la figure IV-14.

```
<condition>::= 'SI' <prop> | {'∃'|'∀'} '%'<variable>
                              ['PAS'] '∈['<scalaire>,...'] SI' <prop>
<prop>     ::= ['PAS'] '('<info> <comp> <info>')' [<op> <prop>]
<op>       ::= {'ET'|'OU'}
<comp>     ::= {'='|'<='|'>='|'<'|'>'|'<>'}
<info>     ::= <valeur> | <objet>'('<paramètre>')'
<objet>    ::= {'T:'<tâche>|'D:'<document>'|
                'DP:'<document.partenaire>|'P:'<partenaire>|...}
<paramètre>::= <champ>'['<sous_champ>']'
<valeur>   ::= <scalaire> | '%'<variable>
```

Figure IV-14: Syntaxe de définition d'une condition dans une clause

Une condition d'une clause est définie sur une proposition, qui peut être soit vraie soit fausse. Si une variable est utilisée dans une proposition, celle-ci doit utiliser un quantificateur (nous en verrons un exemple plus loin). Voici un exemple assez simple de condition définie dans une clause :

```
SI ( D:cwatup_A17(contenu.PPA)="" ) ALORS ...
```

Si le document CWATUP annexe 17 ne mentionne aucun plan particulier d'aménagement (PPA), alors...

Une information peut concerner n'importe quel objet de notre système, un objet correspondant à un type d'entité (ou d'association) défini dans notre base de données PBFlow:

- T: une tâche;
- D: un document;
- P: un partenaire;
- R: un rôle;
- ...

Nous ne considérons cependant que les instances des objets : une décision dans la gestion du flux pourra être prise en fonction de caractéristiques de documents, de tâches, de dossiers, etc. mais toujours en tant qu'instances.

Or, la cardinalité des types d'association «genre» est toujours 1-1 pour le rôle «est du genre...». Ceci nous permet d'assimiler les instances à leur type comme l'illustre la figure IV-15 pour les documents et types de documents.

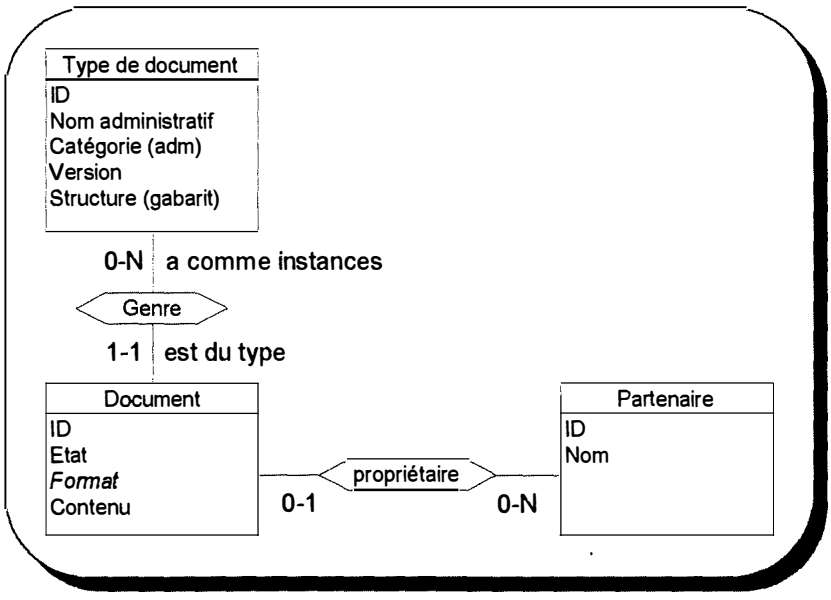


Figure IV-15: Relation entre type et instance de documents

En clair, cela signifie que l'on n'accédera - dans la spécification des clauses - qu'indirectement aux types d'entités dénommés «Type de ...». Par exemple, la catégorie d'un document sera celle du type de document qui lui est associé, et qui est unique.

Nous généralisons ce principe pour les autres association où la cardinalité est 0-1 ou 1-1. Par exemple, un document n'ayant qu'un seul responsable, et un seul propriétaire,

nous pourrons parler de propriétaire ou responsable d'un document de la manière suivante: « `SI (D:document (propriétaire[nom])="Durand") ...` ». Dans ce cas-ci, nous accédons en réalité au champ "nom" de l'entité "acteur" (partenaire) qui est propriétaire du document "*document*".

Un champ particulier est celui d' "état". Il indique l'état d'un objet tel qu'une tâche ou un document. Par exemple: « SI (T: *tâche* (état) = 'en cours') ... »

La particularité est que, pour les documents, il est possible de référencer la valeur d'un champ dynamique : dans notre exemple CWATUP A21, ce pourra être `"contenu[nom_architecte]"`, `"contenu[province]"`, etc. Attention cependant: les noms de champs utilisés doivent être impérativement ceux définis dans le gabarit standard, par défaut, de PBFlow pour ce type de document.

La convention pour nommer un document est de donner le nom (administratif) du type de document. Pour rappel, nous supposons en effet que, dans une instance de flux particulière, il n'existe à un moment donné qu'une seule version d'un document d'un type donné, et qu'une seule instance de tâche d'un type donné. Au cas où un type de tâche serait exécuté plusieurs fois, ou au cas où un document déjà existant serait recréé, nous ne tenons compte dans notre gestion du flux que de la dernière instance créée, ce qui est logique.

Il en va de même pour les tâches : le nom d'une tâche, utilisé dans nos spécifications, sera le nom du type de tâche.

Un scalaire enfin, est une valeur entièrement définie par elle-même (contrairement à une variable). Dans notre contexte, ce peut être "vrai", "faux", "créé", "terminé", "1", "48", "Dupond", etc. La valeur "" (vide) désigne une absence de valeur, ou un état indéfini. Notons à ce propos que le langage de spécification que nous proposons n'est pas typé (nous nous inspirons du langage TCL [TCL]).

4.4.3. Spécification d'une action

Les actions qui peuvent être prises dans une clause sont essentiellement des lancement d'exécutions de tâches. Nous reprenons ici toute la syntaxe définie pour la spécification d'une condition (expression), c'est-à-dire qu'il est également possible, dans la spécification d'une action, d'utiliser une information contenue dans un document ou dans la base de données PBFlow.

```
<action> ::= {'ALORS','SINON'} LANCER T:'<tâche>
             ['AVEC' <champ>['['<sous_champ>']'] '=' <valeur>...]
```

Figure IV-16: Spécification d'une action dans une clause

Exemple d'action :

```
ALORS LANCER T:creer_rapport
```

L'action est de lancer la tâche d'exécution d'un rapport; toutes les références au flux sont implicites. En particulier, l'assignation et les documents spécifiés en entrées/sorties, et définis au niveau du type de tâches (prédéfinis) sont utilisés.

Exemple complet de spécification d'une clause:

```
Prédicat: terminaison(analyse)
SI ( D:document22(contenu[importance])='urgent' )
ALORS LANCER T:creer_rapport
      AVEC exécutant=D:document22(propriétaire[ID])
```

Par rapport au cas précédent (exemple d'action), la différence est que si le document 22 indique comme valeur 'urgent' pour le champ 'importance', alors l'exécutant de la tâche sera l'auteur (le propriétaire) du document22 !

4.4.4. Extension pour des enchaînements ad-hoc

Afin de gérer plus facilement certaines situations ad-hoc, nous proposons une extension intéressante dans la définition des clauses. La nouvelle syntaxe de définition d'une condition (expression) serait la suivante :

```
<condition>::= 'SI' <prop> | {'∃'|'∀'} '%'<variable>
                           ['PAS'] 'є['<scalaire>,...'] SI' <prop>
<prop>     ::= ['PAS'] '('<info> <comp> <info>')' [<op> <prop>]
<op>       ::= {'ET'|'OU'}
<comp>     ::= {'='|'<='|'>='|'<'|'>'|'<>'}
<info>     ::= <valeur> | <objet>'('<paramètre>')'
<objet>    ::= {'T:'<tâche>|'D:'<document>'|
                'DP:'<document.partenaire>|'P:'<partenaire>|
                'Q:("'<question>'",['<scalaire>','...'])'}
<paramètre>::= <champ>'['<sous_champ>']'
<valeur>   ::= <scalaire> | '%'<variable>
```

Figure IV-17: Syntaxe étendue de définition d'une condition dans une clause

L'innovation est de permettre de poser une question à un acteur. Notre idée est la suivante : jusqu'à présent, nous avons considéré qu'une source d'information de base est un document; pourquoi un acteur ne serait-il pas aussi une source d'information ? Nous pourrions par exemple poser la question suivante à un acteur :

```
SI ( Q:("Le dossier est-il urgent ?",["Oui","Non"])="OUI" )
```

La question posée à l'acteur est de savoir si le document est urgent ou non, élément que notre système PBFlow n'est pas capable de deviner par lui-même. Nous faisons donc comme si, une fois que l'acteur aura répondu à la question, la réponse se trouvait dans un document supplémentaire séparé.

Le nombre de valeurs (scalaires) de réponses possibles doit être supérieur à un, avec la première valeur comme réponse par défaut.

Cet ajout permet de modéliser les situations ad-hoc où il n'est pas possible de formaliser en toute généralité une condition d'enchaînement de tâches.

Mais à qui sera posée cette question, quand, et comment ? La réponse n'est pas toujours facile : prenons l'exemple d'une clause pré réalisant une synchronisation (figure IV-18).

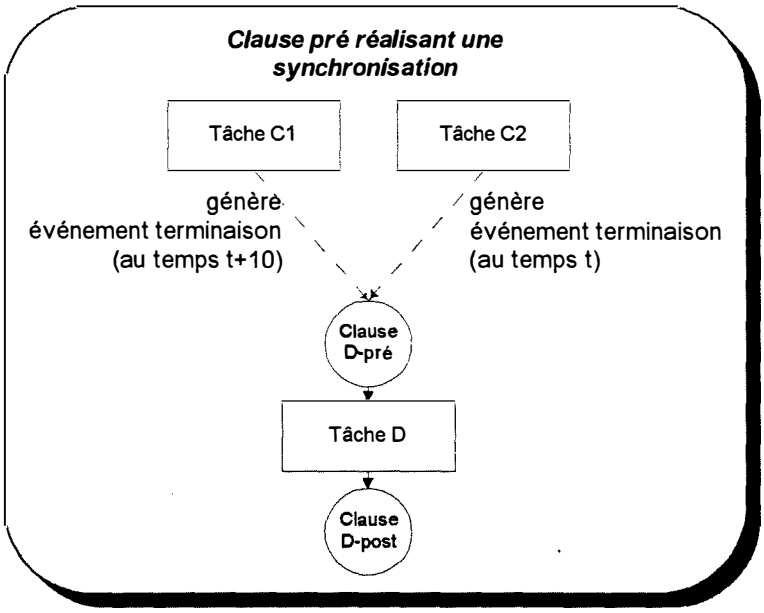


Figure IV-18: Exemple de synchronisation (clause pré)

Le prédicat de la clause D-pré est (terminaison C1 et terminaison C2), et l'action est le lancement de la tâche D.

Nous pouvons observer deux problème dans cet exemple. Le premier problème est le manque d'unicité de l'acteur en tant que source d'information. En effet, l'acteur à qui PBFlow serait susceptible de poser la question n'est pas unique : la question pourrait être

posée à l'acteur ayant réalisé la tâche C1 tout aussi bien qu'à l'acteur ayant réalisé la tâche C2. Pour résoudre ce premier problème, on serait tenté de simplement rajouter la possibilité de spécifier dans la clause à quel acteur PBFlow posera la question. Mettons dans notre exemple que ce soit à l'acteur ayant réalisé la tâche C2 : cela nous amènerait un second problème.

Le second problème est l'indisponibilité de l'acteur en tant que source d'information. Dans notre exemple, la clause D-pré est activée lorsque les événements terminaison de C1 et terminaison de C2 ont tous deux eu lieu, soit au temps t+10... soit aussi bien longtemps après que l'acteur C1 ait terminé sa tâche. Entre-temps, l'acteur C1 est peut-être déjà en congé...

En conclusion, nous limiterons l'usage de questions interactives (Q:) aux clauses POST. Nous savons en effet qu'une clause POST est activée par la terminaison d'une et une seule tâche, à l'exécutant unique. Dès lors, les deux problèmes que nous avons soulevés n'ont plus de raison d'être.

IV.4.5. Gestion du temps

Le temps doit pouvoir être pris en compte dans la gestion du flux.

Il interviendra tout d'abord pour indiquer, pour mémoriser, à quels moments des événements ont eu lieu : il s'agira donc d'informations temporelles qui devront être stockées avec les objets, et qui concerne avant tout les fonctionnalités d'audit et d'historique.

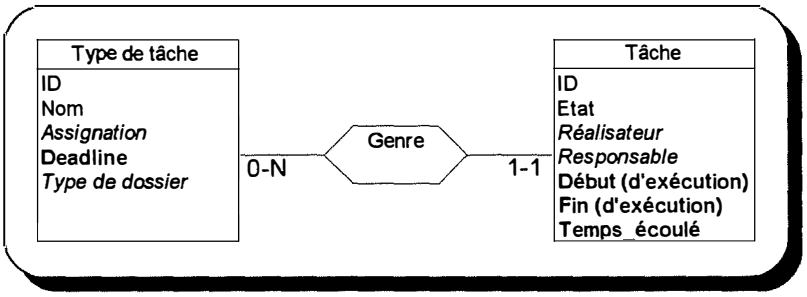


Figure IV-19: Relation entre tâche et type de tâche

Mais le temps intervient également directement dans la gestion du flux, essentiellement sous forme d'*événements temporels*. Un événement temporel est lié à la survenance d'un moment particulier, dans un contexte absolu (ex.: le 1er janvier), mais le plus souvent dans un contexte particulier, relatif à un objet déterminé. Le cas typique est le dépassement d'une «deadline» (date limite) dans le contexte de l'exécution d'une tâche donnée. Nous abordons ici le problème de la gestion des retards.

Pour pouvoir gérer le temps, deux ajouts sont nécessaires : un dans la définition du prédicat des clauses, pour faire explicitement référence aux événements temporels, et l'autre dans la définition des conditions, pour représenter les états liés au temps (date/heure).

La définition de prédicats de clauses pourrait donc se conformer à la syntaxe suivante.

```
<prédicat>    ::= <proposition>
<proposition>::= <événement> | '[' <proposition> <op> <proposition>']'
<op>          ::= {'ET'|'OU'}
<événement>   ::= {'début'|'fin'} '(S:'<temps>')' |
                  {'lancement'|'terminaison'|'suspension'|'reprise'|
                  'annulation'} '(T:'<tâche>')' |
                  {'création'|'destruction'} '(D:'<document>')' |
                  {'envoi'|'réception'} '(DP:'<document.partenaire>')'
```

Figure IV-20: Syntaxe étendue de définition d'un prédicat de clause

Le «temps» pourra être une journée, une heure, une date, ou un laps de temps (voir modèle temporel [II.3.4.]).

La figure suivante (IV-21) synthétise la syntaxe de définition de couples condition-action dans une clause.

```
<couple C/A> ::= <condition><action> ...
<condition>::= 'SI' <prop> | {'∃'|'∀'} '%'<variable>
                           ['PAS'] '∈['<scalaire>,...'] SI' <prop>
<prop>     ::= ['PAS'] '('<info> <comp> <info>')' [<op> <prop>]
<op>       ::= {'ET'|'OU'}
<comp>     ::= {'='|'<='|'>='|'<'|'>'|'<>'}
<info>     ::= <valeur> | <objet>'('<paramètre>')'
<objet>    ::= {'T:'<tâche>| 'S:'<systeme> |'D:'<document>'|
                'DP:'<document.partenaire>|'P:'<partenaire>|
                'Q:("'<question>'",['<scalaire>','...'])'
<paramètre>::= <champ>'['<sous_champ>']'
<valeur>   ::= <scalaire> | '%'<variable> |
               <scalaire> {'+'|'-'} <scalaire>
<action> ::= {'ALORS','SINON'} LANCER T:'<tâche>
             ['AVEC' <champ>['['<sous_champ>']'] '=' <valeur>...]
```

Figure IV-21: Syntaxe étendue de définition d'une condition dans une clause

Voici un exemple de clause (pré) faisant intervenir le temps :

```
SI ( S:temps(date) >= T:Analyse(deadline+début) ) ALORS ...
```

Si la date du système est postérieure à la deadline de la tâche d'analyse (délai maximum+date de début d'exécution de la tâche) , alors...

Voici un autre exemple de clause (pré) faisant intervenir le temps :

```
SI ( T:Analyse(deadline) >= T:Analyse(temps_écoulé) ) ...
```

Si la deadline de la tâche d'analyse (délai maximum) est supérieur au temps d'exécution de cette tâche, alors... Notons ici qu'on parle de temps d'exécution écoulé; en d'autres termes, si la tâche a été suspendue un certain temps, le temps écoulé n'a pas varié pendant toute cette durée.

Nous avons toutefois pensé à inclure des quantificateurs universels pour la définition des clauses : ceci nous permettra de gérer les retards de manière plus générique. Pour chaque type de flux, nous pourrons définir des tâches dont le but sera de gérer les deadlines de n'importe quelle tâche : ces tâches pourront alors, par exemple, envoyer un e-mail à un responsable pour l'avertir de la situation. Voici un exemple de ce que pourrait donner une clause pré d'une tâche gérant les deadlines :

```
Prédicat: début(S:journée)
∀ %TACHE ∈['Analyse','Création_Rapport',...]
SI ( T:%TACHE(deadline) <= S:date )
ALORS LANCER T:Gestion_retard AVEC entrée=T:%TACHE
```

Dans cet exemple, lorsqu'une nouvelle journée commence, toutes tâches qui sont dans la liste donnée et dont leur deadline a été dépassée déclencheront le lancement de la tâche «Gestion_retard», avec comme entrée (informationnelle) les informations relatives à toutes les tâches en retard.

Nous pouvons aussi utiliser tous ces mécanismes dans un autre but : gérer un changement intervenant dans une procédure, suite à une décision sur le plan législatif par exemple.

Cette modification légale pourrait affecter la définition de formulaire-type, ainsi que de tâches. Par exemple, il peut avoir été décidé que la province n'a plus à rendre d'avis concernant les procédures d'attribution de permis de bâtir. Dans ce cas, nous pourrions définir une couple condition-action qui aurait la forme suivante :

```
SI ( S:temps(date) < '01/01/1996' )
ALORS LANCER T:Avis_province
```

Si la date est antérieure au 1er janvier 1996, la province sera sollicitée; dans le cas contraire, elle ne le sera pas.

```
SI ( S:temps(date) < '01/01/1996' )
ALORS LANCER T:Ancienne_tâche
ALORS LANCER T:Nouvelle_tâche
```

Ce dernier exemple illustre la possibilité de programmer un changement de type de tâche à une date déterminée.

IV.5. Exécution du flux

Pour mieux se représenter la manière dont notre système de workflow va gérer le flux à l'exécution, nous pouvons représenter - pour chaque tâche - les différents éléments de spécification d'un type de flux sur le schéma Entité/Association présenté ci-dessous.

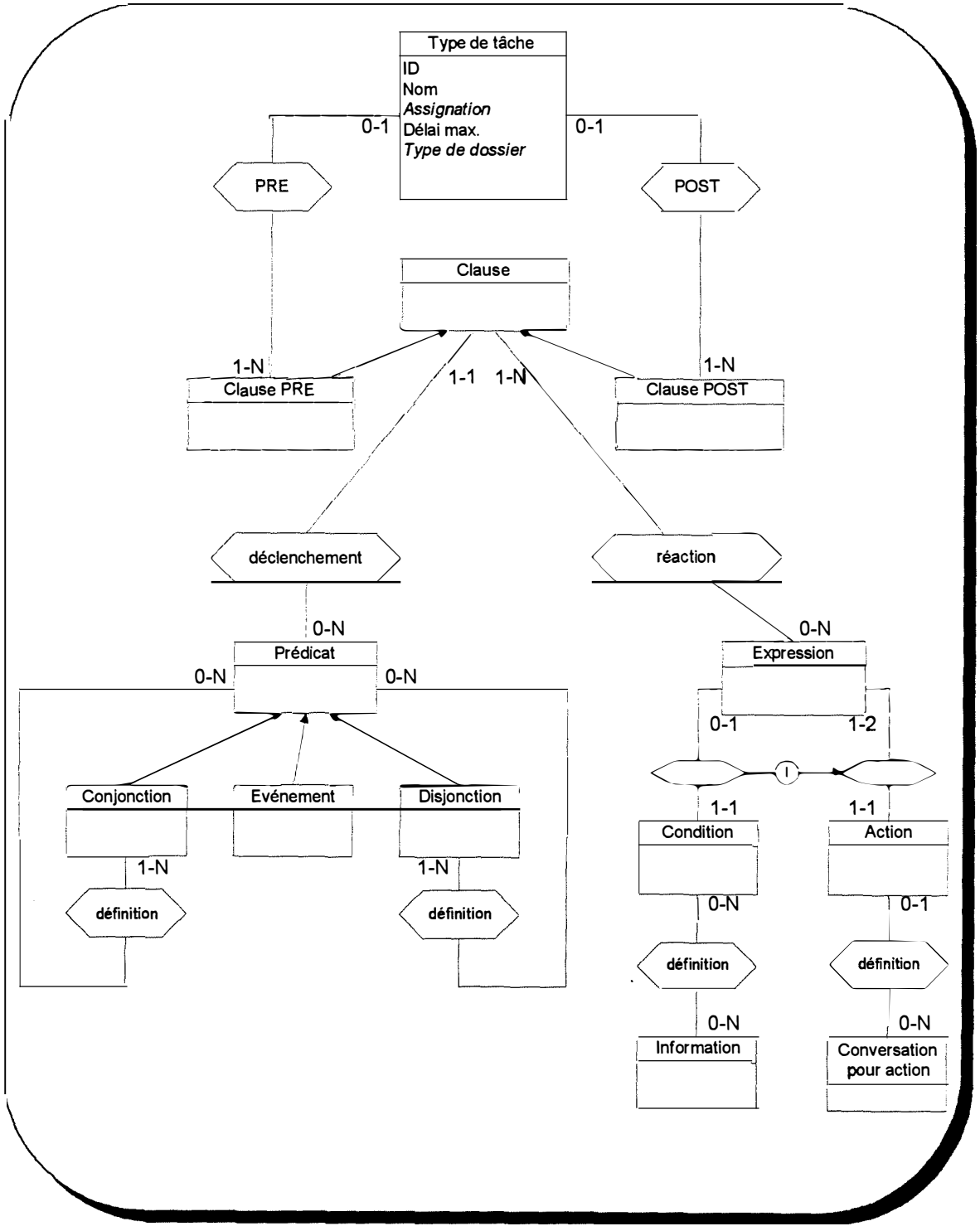


Figure IV-22: Schéma E/A de spécification des clauses pré/post d'une tâche

Sur ce schéma [figure IV-22], nous pouvons distinguer les éléments de spécification du flux, et plus particulièrement que :

- les clauses PRE et POST sont définies pour chaque type de tâche;
- chaque clause est composée d'un prédicat (stimulus déclencheur), et d'un ensemble de couples condition-action;
- un prédicat de clause est soit un événement, soit une conjonction ou disjonction définie sur un sous-prédicat;
- dans un couple condition-action, la condition est facultative, et l'action obligatoire;
- si une condition est définie, elle doit être associée à une action;
- une condition est définie sur un ensemble d'information;
- une action peut initier une conversation pour action (pour lancer une tâche).

Nous pouvons à présent esquisser la manière dont le gestionnaire de flux de PBFlow va exécuter un flux prédéfini, à l'exécution :

- lorsqu'un événement survient, PBFlow regardera si cet événement intervient dans un prédicat, c'est-à-dire s'il contribue à la réalisation d'un déclenchement de clause;
- lorsqu'un prédicat de clause est vrai, la condition de déclenchement est réalisée; PBFlow va alors regarder :
  - si une seule action est spécifiée, sans condition (cas typique d'une clause pré), PBFlow va lancer cette action;
  - sinon, PBFlow va évaluer la première condition;
    - si elle est vraie, il lancera l'action correspondante;
    - sinon, il lancera l'action alternative (si elle est définie);
  - le gestionnaire de flux de PBFlow procédera de même pour les autres couples condition-action.

Conclusion

Notre objectif était de contribuer à la conception d'un système de workflow inter-organisationnel, centré sur les documents, multimédia, et s'appliquant aux procédures administratives, largement prédéfinies mais avec cependant des situations ad-hoc.

Jusqu'à présent, la littérature n'avait consacré que peu de place à ce genre de système de collaboration. Nous avons comblé un vide en explorant les particularités de ce type de système : la gestion des documents et du flux. Nous avons également élaboré une proposition concrète.

Ce que nous aurions encore aimé aborder, c'est l'intégration et l'implémentation de notre proposition dans un système global de workflow, avec gestion des conversations. Nous avons cependant lancé un certain nombre de pistes en ce qui concerne les choix techniques et l'implémentation.

Acronymes

CGI	Common Gateway Interface
CSCW	Computer-Supported Collaborative Work
DTD	Document Type Definition
E/A	Entité/Association (schéma)
E/S	Entrées/Sorties
EDI	Electronic Data Interchange
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDA	Interactive Design Approach
IETF	Internet Engineering Task Force
MIME	Multipurpose Internet Mail Extensions
MCH	Milano Conversation Handler
MOH	Milano Organizational Handbook
MWMS	Milano Workflow Management System
OLE	Object Linking and Embedding
PBIA	Permis de Bâtir Inter-Administrations
PERL	Practical Extraction Report Language
PPA	Plan Particulier d'Aménagement
PGP	Pretty Good Privacy
SGML	Standard Generalized Markup Language
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
TI&C	Technologies de l'Information et de la Communication
TMS	The Milano System
WWW	World-Wide-Web

Bibliographie

Références sur les systèmes de workflow et assimilés

- AGOS95 - *The Milano System*, University of Milano, Alessandra Agostini, Giorgio De Michelis, Maria-Antonietta Grasso. Deliverable 1.3 of the COMIC project (ESPRIT BRA #6225), 1995.

- BOD89 - *Conception assistée des systèmes d'information (méthode, modèles, outils)*, François Bodart & Yves Pigneur, Seconde édition, MIPS Masson, 1989.

- DEMI94 - *From the analysis of cooperation within work-processes to the design of CSCW Systems*, University of Milano, Giorgio De Michelis, 1994.

- DEMI95 - *Work Process, Organizational structures and cooperation support: managing complexity*, University of Milano, and RSO, Milano, Italy. G. De Michelis, 1995.

- Konsynski93 - *Strategic Control in the Extended Entreprise*, IBM systems Journal, vol. 32, n°1, 1993, pg. 111-142.

- Lesuisse96 - *Gestion Stratégique de Systèmes d'Information*, Roland Lesuisse, 1996.

- LOBET95 - *EDI Adoption and Standard Choice : A Conceptual Model*, C. Lobet-Maris & R. Delhaye, ECIS 95, Third European Conference on Information Systems, Athens, June 95.

- ODESTA LiveLink (workflow system) - http://www.odestasys.com/

- POOS94 - *Prise en compte de situations de coopération non prédéfinies dans un système de workflow* (Mémoire), Institut d'informatique des FUNDP, Namur, William Poos, 1994.

- POOS96 - *W3Flow : An Intranet Based Inter-organizational Cooperation Management System Research, Perspectives Preliminary Draft*, William Poos & François Bodart, Research Paper RP-96-007, Avril-Mai 1996.

- SDW - *Schéma directeur pour une infrastructure en technologies d'information et de communication dans le namurois*, CIGER, François Bodart & al., Juillet 1995.

- TAES96 - *The Milano System : Agents & Enabled-mail*, Frédéric Taes - Adolphe Ayissi Eteme, Mai 1996.

- THP - *Contribution à la spécification de situations de cooperation ad hoc et à leur prise en compte dans les systèmes de Workflow*, Thèse de Doctorat. Facultés Universitaires Notre-Dame de la Paix, Institut d'Informatique, Namur. Thérèse Petitjean, 1994.

- TWI - *The Workflow Imperative*

Références techniques

- CMC n°60 - *Lotus intègre Internet à son Notes 4*, CM Corporate n°60, 24 Novembre 1995, Fred van der Molen.

- DELPHI95 - *Teach Yourself Delphi in 21 days*, Wozniewicz & Shammas, SAMS Publishing, 1995.

- RFC 1521-1522 - MIME, *Multipurpose Internet Mail Extensions*

- SGML86 - *Your Multi-Platform Publishing and Information Management Solution*, A Basic Introduction, 1986. http://www.sq.com/htmlsgml/s-tutor.htm

- SYR96 - *Etude préliminaire de l'architecture et de l'environnement technique, Syrecos*, Frédéric Taes, Juillet 1996.

- TCL - *TCL - An Internet Scripting Language*, http://www.sun.com/960710/cover/index.html

- Transit - *HTML Transit* ™, http://www.infoaccess.com/products/transit

Les modèles de formulaires pour la création automatique des documents

MODELES	LIBELLE	TYPE	CODE
MODELE-A	Formulaire A : Habitation	B	FOA
ACCRECPB	Accusé de réception d'une demande de bâtir.	B	ARC
ART192	Délibération application de l'art. 192 du CWATU.	B	192
ATTESARC	Attestation architecte.	B	ARI
AVIS-URB	Réclamation de document auprès de l'URBANISME.	B	URB
DEMBATIR	Demande d'un permis de bâtir.	B	DEM
ENQ-INDI	Modèle individuel pour enquête.	B	ENQ
ENQ-OUAV	Ouverture d'enquête.	B	ENQ
ENQ-PUBL	Clôture de la publication d'une enquête.	B	ENQ
ENQAFFIC	Formulaire d'affiche.	B	ENQ
ENQTRANS	Transmis avis d'enquête aux autres communes.	B	ENQ
ENQU-PV	P.V. d'ouverture d'enquête.	B	ENQ
ENSEI-LT	Lettre stipulant qu'il faut un P.B. (enseigne).	B	ENS
ENSEIGNE	Complément modèle A pour les enseignes.	B	ENS
INCIDENC	P.B. notice d'évaluation des incidences.	B	INC
MODELE-B	P.B. Modèle B - Lotissement.	B	BAT
PROLPBAT	Prolongation d'un permis de bâtir.	B	BAT
REFUS-C	Refus d'un permis de bâtir.	B	BAT
TRANS-CV	Transmis au Commissaire Voyer.	B	VOY
TRANS-PB	Transmis d'un dossier de permis de bâtir.	B	TRA
TRS-ENQ	Transmis de l'enquête - Urbanisme Pollutions.	B	TRS
VISARCHI	Modèle visa ordre des architectes.	B	VIS
ACCRECPL	Avis de Réception d'un Permis de Lotir	L	REC
DELPERPL	Délibération permis de lotir.	L	LOT
DEMLOT	Demande d'un permis de lotir.	L	LOT
ENQ-INDI	Modèle individuel pour enquête.	L	ENQ
ENQ-OUAV	Ouverture d'enquête.	L	ENQ
ENQ-PUBL	Clôture de la publication d'une enquête.	L	ENQ
ENQAFFIC	Formulaire d'affiche.	L	ENQ
ENQTRANS	Transmis avis d'enquête à d'autres communes.	L	ENQ
ENQU-PV	P.V. d'enquête.	L	ENQ
MODLOTIR	Modification d'un permis de lotir.	L	LOT
PERMLOT	Permis de lotir.	L	LOT
REFMODPL	Refus de modification d'un permis de lotir.	L	LOT
TRS-ENQ	Transmis de l'enquête: Urbanisme - Pollutions.	L	TRS
AVISCU1C	Certificat d'Urbanisme n° 1	U	CER
ALECABLE	Délibération pour la pose de cable par l'A.L.E.	U	ALE
AUTEXPAV	Avis de délivrance d'une autorisation d'exploiter.	U	AEX
AUTEXPLO	Délivrance d'une autorisation d'exploiter générale	U	EXP

MODELES	LIBELLE	TYPE	CODE
BELGACOM	Lettre pour la pose de câbles par BELGACOM.	U	RTT
BELGADEL	Délibération de pose de câbles par BELGACOM.	U	RTT
BOUCHEXP	Autorisation d'exploiter une boucherie.	U	BOU
CADASTRE	Tableau de relevé du Cadastre.	U	CAD
CAMPAGRA	Avis sur la demande d'agrandissement d'un camping.	U	CAM
CAMPAUT	Permis de camping (Annnexe 2).	U	CAM
CAMPAVAG	Demande d'agrandissement d'un camping (annexe 7).	U	CAM
CAMPAVDE	Demande de permis de camping (annexe 6).	U	CAM
CAMPAVIS	Demande de permis de camping (annexe 4).	U	CAM
CAMPREGL	Règlement d'ordre intér. d'un camping (annexe 1).	U	CAM
CAMPTRA1	Transmis d'un dossier de camping à l'URBANISME.	U	CAM
CAMPTRA2	Transmis d'un dossier camping au Com. du Tourisme.	U	CAM
CERTIF-2	Certificat d'urbanisme numéro 2.	U	CER
CERTPUBL	Certificat de publication.	U	PUB
CERTURB1	Certificat d'urbanisme numéro 1.	U	CER
DEMCERT1	Demande de certificat d'urbanisme numéro 1.	U	CER
DEMCERT2	Demande de certificat d'urbanisme numéro 2.	U	CER
ENQ-INDI	Modèle individuel pour enquête.	U	ENQ
ENQ-OUAV	Ouverture d'enquête.	U	ENQ
ENQ-PUBL	Clôture de la publication d'une enquête.	U	ENQ
ENQAFFIC	Formulaire d'affiche.	U	ENQ
ENTTRANS	Transmis avis d'enquête aux autres communes.	U	ENQ
ENQU-PV	P.V. d'enquête.	U	ENQ
EXPGAZ	Autorisation de dépôt de gaz.	U	GAZ
EXPLGARA	Autorisation d'exploitation d'un garage.	U	GAR
INCIDEN2	Etablissements classés N.E. incidences.	U	INC
MISEDISP	Mise à disposition à titre précaire.	U	PRE
PRI-REHA	Demande de prime communale à la réhabilitation.	U	REH
PRIMECON	Rappelle prime communale à la construction.	U	PRI
PRIMREHA	Rappelle prime communale à la réhabilitation.	U	REA
TRANS-CB	Transmis au Commissaire Voyer.	U	VOY
TRANSMIS	Transmis en général.	U	URB
TRS-ENQ	Transmis de l'enquête : Urbanisme - Pollutions.	U	TRS

Transmis

Internes

- S.A. Urbanisme
- S.T. Urbanisme
- Aménagement du Territoire
- Infrastructure
  - Eaux et Forêts
  - Voirie
  - Parcs et Jardins
- Environnement - Permis d'Exploiter
- Service Incendie

Commissions

- C.C.A.T.
  - Groupe Plan particulier d'Aménagement
  - Groupe Circulation
  - Schéma Directeur
  - Sites Mosans et Champêtres
  - Groupe Patrimoine et Architecture

Région

- Ministère de l'Equipement et des Transports
- Ministère de l'Agriculture
- Ministère des Affaires Economiques
- Inspection Générale de l'Eau
- Direction générale des voies hydrauliques
- Administration de l'Urbanisme provincial

Service des Fouilles

Service technique provincial