Chapitre 8. Parcours de lecture

When the user is building a trail, he names it, inserts the name in his code book, and taps it out on his keyboard. Before him are the two items to be joined, projected onto adjacent viewing positions. […]Thereafter, at any time, when one of these items is in view, the other can be instantly recalled […]. Moreover, when numerous items have been thus joined together to form a trail, they can be reviewed in turn, rapidly or slowly, by deflecting a lever like that used for turning the pages of a book. It is exactly as though the physical items had been gathered together from widely separated sources and bound together to form a new book. It is more than this, for any item can be joined into numerous trails.
[…] Tapping a few keys projects the head of the trail. A lever runs through it at will, stopping at interesting items, going off on side excursions.
Vannevar Bush, As we may think, 7.

Comme nous l'avons vu dans le chapitre précédent, les réseaux de description permettent une navigation de corpus en sous-corpus (et inversement). Si cette navigation permet d'atteindre l'ensemble des corpus décrits, c'est souvent par une ennuyeuse série de « déselections » et de sélections de descripteurs. Ne serait-il pas utile d'offrir des « raccourcis » entre corpus, transversalement aux relations d'inclusion ? C'est pour répondre à cette attente, que nous proposons les *parcours de lecture*.

En tant que *trace,* nos parcours de lecture pourront être définis à la fois par les auteurs et les lecteurs des contenus documentaires. Dans le premier cas, ils pourront représenter, par exemple, la séquence des pages ou des illustrations. Dans le deuxième cas, ils s'apparenteront à un historique de lecture lié à une tâche donnée. Ils pourront dans les deux cas être édités, stockés, publiés et réutilisés ultérieurement.

Au cours de ce chapitre, nous présenterons, tout d'abord de manière informelle, nos choix de modélisation ainsi que les raisons qui les ont dictées. Dans un second temps, nous en donnerons une spécification algébrique. Ensuite, nous donnerons un aperçu des possibilités offertes par le modèle en déroulant un petit scénario d'utilisation. Enfin, nous montrerons en quoi notre modèle se distingue des travaux apparentés.

1. Principe

Nous définirons un parcours de lecture comme un *historique* parmi des *étapes de lecture*, permettant une *navigation.* Précisons maitenant chacun de ces aspects.

a. Historique

Le premier choix que nous devons faire porte sur l'*historique*. Dans le domaine de l'hypermédia, on distingue en général trois modèles [BieberEtWan94] : le modèle de la *pile* (utilisé dans les clients Web pour le « retour arrière »), le modèle *chronologique* (utilisé dans les serveurs Web comme « log »[74]) et le modèle de la *visite guidée*. Pour comprendre les différences entre les trois types d'historique, nous suivrons un exemple pas à pas (cf. Figure 8.1).

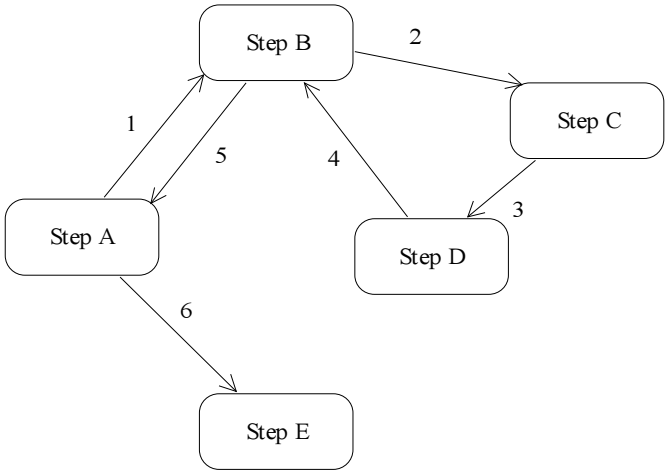

Figure 8.1 : Exemple de navigation entre des étapes de lecture (Diagramme d'état UML).

Supposons qu'un utilisateur passe par les étapes de lecture *A*, *B, C* puis *D*. Dans les trois modèles, l'historique sera [*A*, *B*, *C*, *D*].

Lorsque l'utilisateur retournera à l'étape *B*, le modèle de la pile indiquera [*A*, *B*], le modèle chronologique, [*A*, *B*, *C*, *D*, *B*] et le modèle de la visite guidée, [*A*, *B*, *C*, *D*].

[74] En supposant que le client Web n'ait pas de « cache ».

En retournant à l'étape *A*, l'utilisateur changera le premier historique en [*A*], le second en [*A*, *B*, *C*, *D*, *B*, *A*], et le troisième en [*A*, *B*, *C*, *D*].

Enfin en se rendant à l'étape *E*, l'utilisateur obtiendra pour historique :

- [*A*, *E*] dans le modèle de la pile,
- [*A, B, C, D, B, A, E*] dans le modèle chronologique,
- et [*A, B, C, D, E*] dans le modèle de la visite guidée.

Le modèle de la visite guidée, en stockant l'ensemble des étapes de lecture dans l'ordre de leur première visite et en négligeant les retours en arrière, nous semble préférable du point de vue de la charge cognitive et de la charge computationnelle.

b. Etape de lecture

Ayant défini pour les parcours de lecture ce que nous appelions un historique, nous avons maintenant à préciser ce que nous appellerons une *étape de lecture*.

On serait sans doute tenté d'assimiler une étape de lecture à un objet documentaire (source, fragment, note). Cependant une telle définition nous priverait du contexte documentaire de lecture[75] – l'un des aspects les plus intéressants des réseaux de description.

Pour autant, prendre comme étape le contexte documentaire de lecture serait assez peu judicieux. En effet, si l'on considère les opérations définies pour la navigation dans un réseau de description comme autant de systèmes, ce contexte serait une « sortie » mais jamais une « entrée ». Dit autrement, il serait possible de poursuivre une navigation dans les réseaux de description par une navigation dans les parcours de lecture mais pas l'inverse.

Tenant compte de l'objection précédente, on serait amené à définir une étape de lecture comme un corpus de documents. Cependant, rechercher une étape parmi les *N* étapes disponibles reviendrait à effectuer *N* comparaisons d'ensembles ! En outre, ne considérer que le corpus, néglige le fait qu'à un instant donné l'utilisateur concentre son

[75] Ensemble des objets documentaires affichés à un instant donné par le client de *Porphyre*.

attention sur un seul des objets documentaires présents à l'écran (ce qui se traduit par une activation de la fenêtre correspondante).

Les remarques précédentes, nous conduisent à définir une étape dans un parcours de lecture comme un objet documentaire parmi un corpus. Ainsi, rechercher une étape revient à comparer des « localisations d'objets documentaires[76] » entre elles. Une fois l'étape trouvée, on peut récupérer le corpus dans lequel l'élément documentaire doit être consulté.

Reste à préciser comment le corpus sera désigné : en intension (par ce que l'on a appelé une sélection) ou en extension (par la liste des objets documentaires le composant). Si la première est beaucoup plus concise, elle présente un inconvénient majeur, celui d'avoir un résultat dynamique. Après modification du réseau de description, le corpus obtenu pourrait même ne plus contenir l'élément documentaire cherché ! Les corpus seront donc notés en extension.

c. Navigation

Dans le modèle ainsi défini, plusieurs parcours de lecture pourront se croiser en un même objet documentaire. Par conséquent, le système devra indiquer pour l'objet documentaire activé la liste de ses parcours (pour les facettes connectées).

Après sélection de l'un de ces parcours, étant donné qu'un objet documentaire ne peut apparaître qu'une fois dans un parcours de lecture, le lecteur pourra choisir sans ambiguïté l'étape précédente, l'étape suivante ou l'origine du parcours.

2. Spécification

La Figure 8.2 nous permet de préciser notre modèle. A l'intèrieur d'une facette, il sera possible de définir des *Parcours.* Un parcours correspondra à la séquence de plusieurs objets documentaires. Inversement un objet documentaire pourra apparaître dans plusieurs parcours. Les objets documentaires référencés par les parcours pouvant

[76] Cf. Chapitre 6.

être distants[77], il n'y aura aucune contrainte d'intégrité référentielle sur eux. On appellera « *Etape* » le couple unissant un parcours et un objet documentaire. Chaque étape correspondant à contexte de lecture, elle comportera un *corpus* d'objets documentaires (pouvant eux aussi être distants).

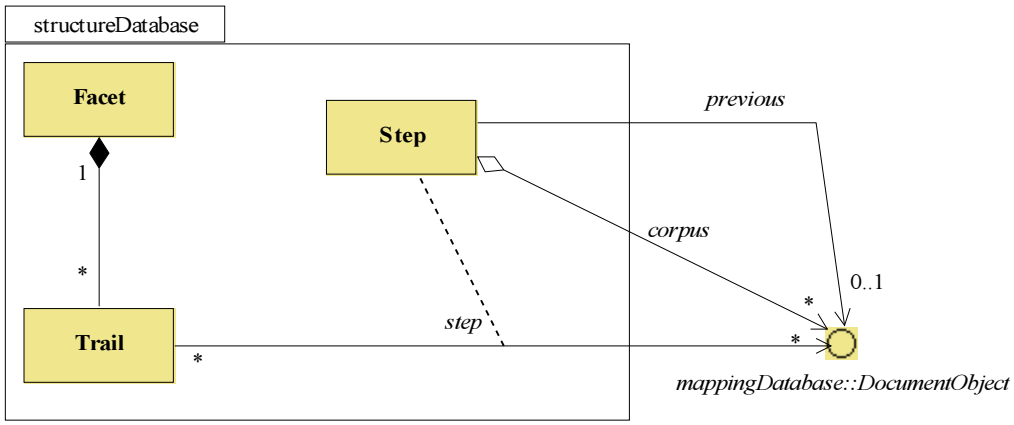

Figure 8.2 : Parcours de lecture (diagramme de classe UML)

Plus formellement, nous aurons affaire aux ensembles suivants : *Facet, Trail, Step, DocumentObject*. De même que dans le chapitre précédent, nous utiliserons des spécifications algébriques[78] pour préciser :

- le schéma des données à stocker (primitives),
- les contraintes supplémentaires que ces données doivent respecter (contraintes),
- les requêtes complexes qui seront effectués sur ces données (définitions).

Primitive : $step(\_,\_) : Trail \times DocumentObject \rightarrow Boolean$

[77] Grâce à l'utilisation des « localisations d'objets documentaires (cf. Chapitre 6).

[78] Voir en annexe l'explication de la syntaxe utilisée.

Axiome :

On appellera *Step* l'ensemble des couples (*t,d*) pour laquelle la relation homonyme sera vraie.

$Step = \{(t,d) | \ step(t,d)\}$

Primitive : *_.getPrevious* : $Step \rightarrow DocumentObject \cup \{NULL\}$

Le fait qu'il s'agisse d'une fonction et non d'une relation traduit la contrainte d'intégrité fonctionnelle suivante : pour une étape donnée, il existe au plus un *DocumentObject* précédent.

Contrainte : Unicité du prochain

Deux étapes d'un même parcours ne peuvent avoir le même prédécesseur.

$\bot \leftarrow (t,d_1).getPrevious = d_0 \wedge (t,d_2).getPrevious = d_0 \ \wedge d_1 \neq d_2$

Contrainte : Intégrité référentielle dans le parcours

Si une étape a un prédécesseur, celui-ci est forcément issu du même parcours.

$\bot \leftarrow (t,d_1).getPrevious = d_0 \wedge \neg \ step(t,d_0)$

Définition : *_.getHome* : $Trail \rightarrow \ DocumentObject$

On appelle *origine* d'un parcours, l'étape de ce parcours ayant pour prédécesseur *NULL*.

$t.getHome = d \leftarrow (t,d).getPrevious = NULL$

Définition : $\_.getNext : Step \rightarrow DocumentObject \cup \{NULL\}$

La fonction de succession est l'inverse de celle de précédence. Dans le cas, où il n'existe aucune étape de t ayant d_0 pour predecesseur, on dira que le successeur de l'étape (t,d_0) est *NULL*.

$(t,d_0).getNext = d_1 \leftarrow (t,d_1).getPrevious = d_0$

$(t,d_0).getNext = NULL \leftarrow \neg ((t,\_).getPrevious = d_0)$

Primitive : $\_.getFacet : Trail \rightarrow Facet$

Le fait qu'il s'agisse d'une fonction traduit la contrainte d'intégrité fonctionnelle suivante : un parcours de lecture appartient à une seule facette.

Définition : $\_.getTrails(\_) : Facet \times DocumentObject \rightarrow Trail^n$

Pour une facette donnée, permet d'obtenir tous les parcours ayant une étape passant par un objet documentaire donné.

$f.getTrails(d) = \{t |\ t.getFacet = f \wedge (t,d).getPrevious = \_ \}$

Primitive : $\_.getCorpus : Step \rightarrow DocumentObject^n$

Cette fonction permet d'obtenir le corpus correspondant à une étape de lecture. On peut alors appliquer le *getFilter* défini dans les réseaux de description. Ainsi, les deux types de navigation peuvent s'enchaîner.

3. Scénario : Feuilleter un ouvrage

Dans l'exemple de la figure 8.3, nous disposons de deux parcours de lecture pour feuilleter un ouvrage (le n°12) : le premier donnant l'enchaînement des paragraphes et le second celui des figures. Un troisième parcours de lecture correspond à l'enchaînement des références bibliographiques d'un autre ouvrage (le n°30) citant le premier. Dans le premier parcours, chaque paragraphe est lu en contexte avec les figures qui y sont référencées. Dans le second, c'est l'inverse. Dans le troisième, chaque passage référencé est lu en contexte avec ceux qui s'y réfèrent.

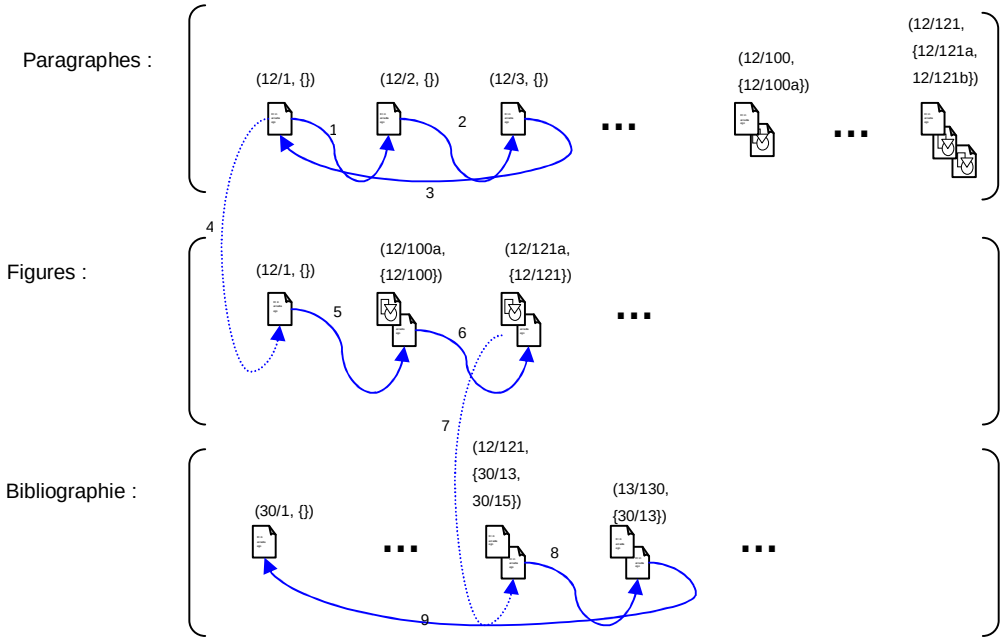

Figure 8.3 : Exemple de parcours de lecture

Supposons que le lecteur choisisse de parcourir l'ouvrage n°12, il se positionne sur le premier objet documentaire (12/1). Il apprend qu'il existe deux parcours documentaires qui y passent : un premier nommé « paragraphes » et un second « figures ». Il choisit paragraphe et passe à l'étape suivante.

A la lecture du nouveau paragraphe (12/2) et de son successeur (12/3) dans le parcours, le lecteur constate que l'ouvrage est susceptible de l'intéresser. Il décide de feuilleter les figures de l'ouvrage. Pour ce faire, il revient à l'origine du parcours « paragraphes » et choisit le parcours « figures ».

En se rendant à l'étape suivante, le lecteur découvre la figure 12/100a, puis 12/121a, toutes deux accompagnées de leur paragraphe explicatif. Intéressé par la figure 12/121a, il sélectionne le paragraphe 12/121 pour le lire.

Il apprend alors qu'un autre parcours, nommé « bibliographie », passe par ce paragraphe. Intéressé par les commentaires 30/13 et 30/15 qui sont faits du 12/121, il continue le parcours. Un autre ouvrage (n°13) très pertinent y est référencé et commenté. Intrigué par la bibliographie de l'ouvrage n°30, le lecteur décide de lire ce dernier *in extenso*, et va donc à l'origine du parcours de lecture.

Notons qu'en feuilletant ainsi le corpus, notre lecteur a tracé un nouveau parcours. S'il le souhaite, il pourra lui donner un nom et le conserver pour un usage ultérieur.

4. Originalité du modèle

Nombreux furent les systèmes hypermédia à proposer une implémentation de la notion de parcours introduite par Vannever Bush [Bush45] : une séquence d'objets documentaires créée par le lecteur, nommée, sauvegardée et publiée.

Certains, conscients du danger de désorientation que représentait une lecture « objet documentaire » par « objet documentaire », proposèrent comme étape de lecture non pas un objet documentaire isolé mais un ensemble de documents [TriggEtWeiser86, Trigg88, Maurer96]. Cependant, leurs parcours ne pouvaient se croiser que s'ils utilisaient explicitement la même référence[79] pour désigner cet ensemble.

Au contraire, la notion de parcours de lecture nous semblant fortement liée à celle du point de vue, il nous semblait indispensable de permettre le croisement de parcours issus de deux points de vue différents (donc ne dépendant l'un de l'autre que par l'intermédiaire du corpus). Le modèle proposé par nos soins permet d'y parvenir tout en gardant une complexité algorithmique très raisonnable.

[79] Cette référence est appelée, suivant les modèles, « noeud table-des-matières » [TriggEtWeiser86], « dessus-de-table » [Trigg88] ou « grappe » [Maurer96].

3ème partie : Études de cas