SARL MEZZONOMY

mezzonomy@orange.fr – (33|0)6.32.97.90.27

Programmation informatique (6201Z)
34 Place de Catalogne 31700 BLAGNAC
Toulouse B 504 641 473

Plate-forme d'édition de document

Pierre Gradit, Mai 2015

Ce document a vocation a être imprimé en couleurs

Introduction

La société à responsabilité limitée (SARL) mezzonomy développe depuis 2008 des plate-formes de coopération basées sur les technologies à données semi-structurées où ce sont les usages qui définissent le sens des données, plus que l'intention de celui qui les saisit. La volonté de fonder une société sur ce thème que son fondateur développe depuis 1997, fait suite à la découverte d'une théorie algébrique de la donnée, appelée « *formulation modale* ». Cette théorie permet de donner une expression graphique à ce présupposé initial sur les rapports entre usage et intention.

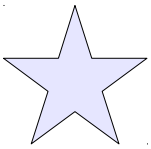
L'objet de ce document est d'expliquer comment peut fonctionner une plate-forme sécurisée et monétisée de partage de connaissance basés sur nos principes. Principes étonnamment proche des conceptions exposées par Jaron Lanier au Personal Democracy Forum de 2012 – y compris dans sa référence au modèle hypertexte symétrique de Ted Nelson et du projet Xanadu. Notre histoire retiendra que c'est sur une page relatant cette conférence que nous nous sommes rencontrés avec Michel Vandenbergue, le porteur du projet de plate-forme qui sert de toile de fond à ce document.

Dans la première partie, nous survolerons un état de l'art de la pratique d'édition de documents. Dans la deuxième partie, nous présenterons la formulation modale qui nous servira par la suite à exprimer nos vues. Dans la troisième partie nous allons exprimer l'état de l'art en formulation modale, puis cela nous permettra de dévoiler le fonctionnement de la plate-forme existante.

La dernière partie sera consacrée à l'étude d'un mode d'édition sans duplication à la manière de Xanadu. La conclusion présentera la méthodologie générale de développement de notre société et détaillera les perspectives de ce travail.

État de l'art

L'informatique moderne est née pendant la deuxième guerre mondiale. Les deux premières applications des **machines** sont des applications de **cryptage** de communications militaires et de **calcul** d'armements.



Dans tous les cas, les machines permettent de transformer des documents, qu'ils soient des textes à vocation exécutive ou des idéalisations de phénomènes physiques :

Machine : Document → Document

Les deux classes initiales d'usage des *machines* sont profondément différentes :

- le *cryptage* est **réversible** : les contraintes posées à cette réversibilité sont même le cœur du sujet.
- Le *calcul* est **irréversible** : il n'est pas possible de déduire les prémisses d'un calcul de ses résultats.

1. Dans l'après-guerre, les progrès autour des *machines* programmables amène à la possibilité pour une classe de *documents*, les **programmes**, d'être eux mêmes des *machines*, :

Machine : Programme → Machine

2. Dans les années 1970, l'apparition des machines personnelles à ouvert la voie au remplacement des documents papier par leurs équivalents numériques : c'est l'avènement de la **bureautique**. Les suites bureautiques comportent deux composants-clé :

- Un **éditeur** de document, décliné en éditeur de document « à imprimer » et en éditeur de document « à projeter ».
- Un **tableur** : cet outil de calcul ouvert et générique peut se trouver étendu par l'usage d'une base de données.

Or l'opération-clé qui rend ces applications utilisables est l'Undo/Redo (Défaire/Refaire). Pour réaliser cette fonctionnalité, il faut concevoir toute **action** comme un *programme réversible*.

Action : Document → Document

Sans la capacité de définir des programmes, pas de document.

La dernière évolution majeure de l'informatique concerne la mise en réseau de toutes les machines. Cette mise en réseau permet deux évolutions en apparence contradictoires :

- La possibilité de centraliser l'information et d'en accorder l'accès à distance ce qui ouvre a voie à une industrialisation des pratiques artisanales des utilisateurs de machines personnelles.
- La possibilité de créer des documents distribués, infalsifiables, des « *blockchains* », dont la première application est une sorte de livre de compte planétaire écrit dans une monnaie complémentaire : le bitcoin.

Après ce rapide tour d'horizon de l'informatique contemporaine, nous allons développer les notions propres à notre approche et voir comment construire une plate-forme de partage de la connaissance sécurisée et monétisée.

Formulation modale

La formulation modale est à l'origine un théorie manipulant indifféremment textes et graphes.

Pour aller plus loin

Aussi inouï que cela puisse paraître, cela n'existe pas. La théorie des graphes standard est antérieure à l'informatique et repose sur la théorie des ensembles. Cette théorie n'est pas informatisable car un ensemble n'a pas d'ordre et tout document est un écrit, ordonné *sui generis*.

Considérons le comportement d'un stylet sur une surface. Lorsque le stylet touche la surface, un **chemin** commence à être dessiné, jusqu'au moment où le stylet se détache de la surface. Ce que nous écrivons :

$$(\text{MoveTo } x_{Contact},y_{Contact} \ (\text{LineTo } \ x_{Déplacement},y_{Deplacement} \ )^* \ )^*$$

L'ensemble des chemins définit un ensemble de points d'intersection, appelés **Noeud**. Ce que nous écrivons :

$$(\text{MoveTo } x_{Contact},y_{Contact} \ (\textbf{Nœud}? \ \text{LineTo } \ x_{Déplacement},y_{Deplacement} \ )^* \ )^*$$

Avec la propriété suivante : chaque **Nœud** est présent deux fois et seulement deux fois.

La formulation modale est extrêmement proche de cette situation canonique. Elle ferme les chemins et dilate les points d'intersection en deux nœuds distincts. La Figure 1 détaille la transformation de l'intersection de chemins quelconque en un diagramme de formulation modale.

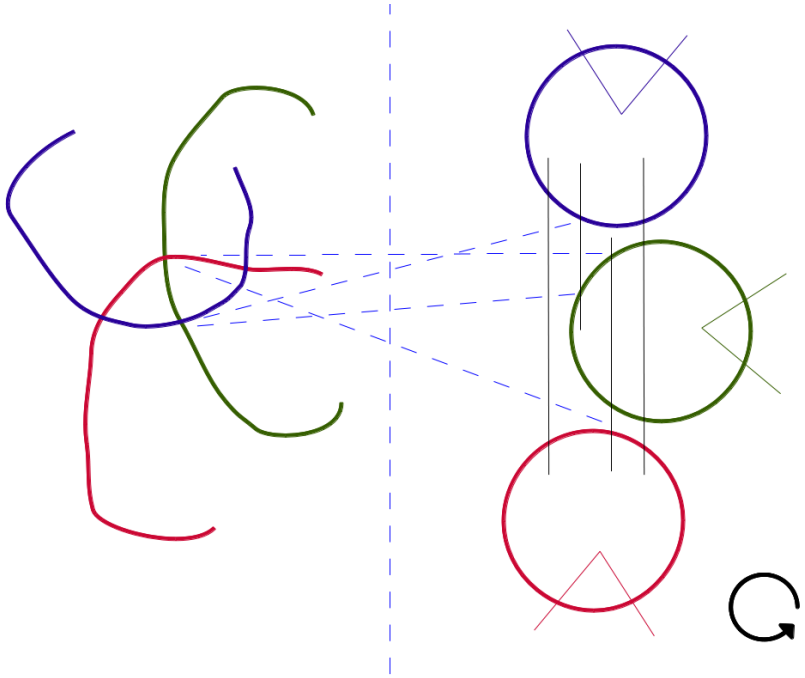


Figure 1 : Écriture modale d'intersections de chemins

A gauche nous avons 3 courbes et 4 intersections, et à droite 3 « *bulles* » et 7 « *liens* » dont quatre correspondent aux intersection initiales et 3 à des « *vues* ». Nous avons indiqué les correspondances pour les nœuds médians situé en haut et en bas. Pour interpréter les « *liens* », il ne faut considérer que les intersections **extrémales**, forcément au nombre de 2. Le diagramme de formulation modale peut s'écrire de façon textuelle par le document 1, où chaque item est cité deux fois.

```
Diagramme(figure-1)
      Chemin(rouge) Nœud(gauche) Nœud(haut) Nœud(droit) chemin(rouge)
      Chemin(vert) Nœud(bas) Nœud(haut) Chemin(vert)
      Chemin(bleu) Nœud(droit) Nœud(bas) Nœud(gauche) chemin(bleu)
Diagramme(figure-1)
```

Document 1 : Forme textuelle du diagramme de la figure 1

Ce qui forme aussi une interprétation satisfaisante du diagramme de gauche. Le nom des items est arbitraire, ici pour faciliter l'interprétation humaine.

L'ordre des lignes conserve une information perdue par la représentation graphique : la séquence d'événements du dessin initial.

> Pour aller plus loin
>
> Notre forme textuelle fait abstraction de la réalité géométrique du dessin de la Figure 1. Cette réalité peut être adjointe au document, selon les modalités de la représentation graphique choisie. Si le dessin a été réalisée de façon brute au stylet, il est possible d'intégrer les MoveTo, LineTo dans le document. D'autres méthodes peuvent être utilisées. La dernière, la plus sophistiquée, utilise une triangulation où chaque item est un triangle.

Après les diagramme et les documents, Il existe une autre forme des diagrammes de formulation modale : les tables. Pour cela, il suffit de remarquer qu'un diagramme de formulation modale est un « *binary decision diagram* » sur les **nœuds** : les intersections (extrémales) entre **bulles** et **liens**. A chaque nœud, nous pouvons d'une part associer son **pair** – l'autre intersection sur le même *lien*, et son **prochain** – l'intersection suivante sur la même *bulle*.

Figure-1			Diagramme	
id	**next**	**peer**	**class**	**attributes**
start-red-path	*red-left*	*end-red-path*	Chemin	color="rouge"
red-left	*red-top*	*blue-left*	Nœud	
red-top	*red-right*	*green-top*	Nœud	
red-right	*end-red-path*	*blue-right*	Nœud	
end-red-path	*start-red-path*	*start-red-path*		
start-green-path	*green-bottom*	*end-green-path*	Chemin	color="vert"
green-bottom	*green-top*	*blue-bottom*	Nœud	
green-top	*end-green-path*	*red-top*		
end-green-path	*start-green-path*	*start-green-path*		
start-blue-path	*blue-right*	*end-red-path*	Chemin	color="bleu"
blue-right	*blue-bottom*	*red-right*		
blue-bottom	*blue-left*	*green-bottom*		
blue-left	*end-blue-path*	*red-left*		
end-blue-path	*start-blue-path*	*start-blue-path*		

Table 1 : Forme tabulaire du diagramme de la figure 1

Encore une fois les identités données au nœuds sont arbitraires et données ici pour faciliter la lecture par un humain. Cette forme est celle manipulée par la machine en mémoire.

- Cette représentation compatible « *Document Object model* » (DOM) où le nœud « *firstChild* » est remplacé par « *peer* ». Ce n'est pas la seule différence : dans le modèle DOM les fonctions **first** et **next** sont des **fonctions partielles** complétées par **None** (sans chaînes infinies) vérifiant que tout nœud (sauf le nœud document) est soit *first* soit *next* d'un seul autre nœud.

- En formulation modale, les fonctions **peer** et **next** sont des **bijections** (sans chaînes infinies), ce qui signifie que tout nœud à un pair et un suivant et est lui même pair d'un nœud et suivant d'un nœud. En outre, La relation de pair est une convolution, le pair d'un nœud est *le* nœud qui a pour pair ce nœud.

Pour aller plus loin

L'expression ensembliste des diagrammes modaux est rédigée ainsi :

(next : Q → Q, peer : Q → Q) est un diagramme modal si et seulement si

- *Q is a denombrable set*
- *$next^{-1}$ is a function: $\forall x, |\{y/\ next(y)=x\}|=1$*
- *$peer^2$ is identity: $\forall x, peer(peer(x))=x$*
- *next has no infinite chains: $\forall f: ℵ \to Q, \exists (i,j) \in Q^2, next(f(n))=f(n+1) \Rightarrow f(i)=f(j)$*

Q est un ensemble de travail, le support du diagramme est la partie de Q pour laquelle *next* ou *peer* n'est pas l'identité, nous notons ce support *{next, peer}*. Nous appelons Q l'ensemble des nœuds par analogie avec l'ensemble des fractions, pour lequel *next* = *λx:x+1*, et p*eer* = *λx:1/x*, Q n'est pas un diagramme de formulation modale car *next* a un chaîne infinie, l'identité sur l'ensemble des entiers naturels ℵ.

L'utilisation de bijections en lieu et en place de fonctions partielle ouvre deux perspectives, l'une théorique et l'une pratique.

Pour aller plus loin

La perspective théorique est une définition de la dualité de diagramme bien plus robuste que la définition classique :

$$\overline{(next, peer)} = (next \circ peer, peer) \approx (peer \circ next, peer)$$

Cet outil nous a aidé très utile pour explorer les capacités descriptives de la formulation modale, en particulier la nécessité de mettre les données sur les paires de pairs (d'où les cases grisées dansa la vue tabulaire) et la volonté de trouver une méthode de calcul graphique de la dualité qui soit en temps constant motive largement la représentation par triangulation.

Nous allons nous attarder sur la perspective pratique, bien plus importante pour l'objet de ce document. Nous pouvons faire opérer des diagrammes les un sur les autres par composition :

$$(next\ ,\ peer).(next',\ peer') = (next \circ next',\ peer \circ peer')$$

Pour que *(next o next', peer o peer')* soit un diagramme modal, il faut vérifier que *(peer o peer')* soit une convolution. Pour cela, il suffit d'imposer à *peer'* d'être l'identité sur tout élément du support {*next* , *peer*}. Dans la figure 2, nous avons représenté la combinaison du diagramme bleu et du diagramme rouge qui donne le diagramme vert. L'endroit où réside la convention graphique est lors d'intersection entre trait et bulle de couleur différentes, ici entre bulles bleues et liens rouges. Ces intersections ne sont pas des nœuds, elles signifient l'identification entre le pair (rouge) de cet intersection et son

prédécesseur (bleu). Quelques identifications ont été fournies par des traits noirs en pointillés pour faciliter la lecture.

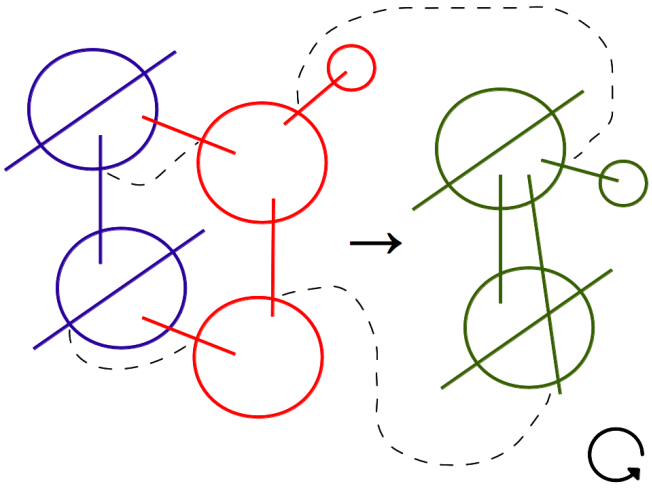


Figure 2 : Transformation de diagramme

Dans notre exemple, le diagramme rouge agit comme un programme sur le diagramme bleu, le résultat étant le diagramme vert. Nous avons donc une capacité essentielle pour prouver que nos diagrammes sont des documents, nous avons un vocabulaire d'action.

Application aux documents

Dans cette partie, nous allons nous intéresser aux documents, comme nous l'avons remarqué plus haut, notre modèle est très proche du « *Document Object Model* », qui structure la représentation des documents hypertexte et peut être utilisé pour les documents imprimés. Considérons le document très simple ci-dessous :

```
<html>
	<body>
		<a href="http://lenr-cities.com">LENR-Cities</a>
		to accelerate market transformation and foster LENR demand.
	</body>
</html>
```

Si la balise d'un nœuds est absente des balise de ses fils, l'ordre de lecture suffit pour déterminer quelle balise ouvre et quelle balise ferme.

Nous obtenons alors une forme textuelle :

```
html
	body
		a href="http://lenr-cities.com" '''LENR-Cities''' a
```

```
                '''to accelerate market transformation and foster LENR demand.'''
        body
html
```

Et de cette forme textuelle nous pouvons construire le diagramme modal de la Figure 3.

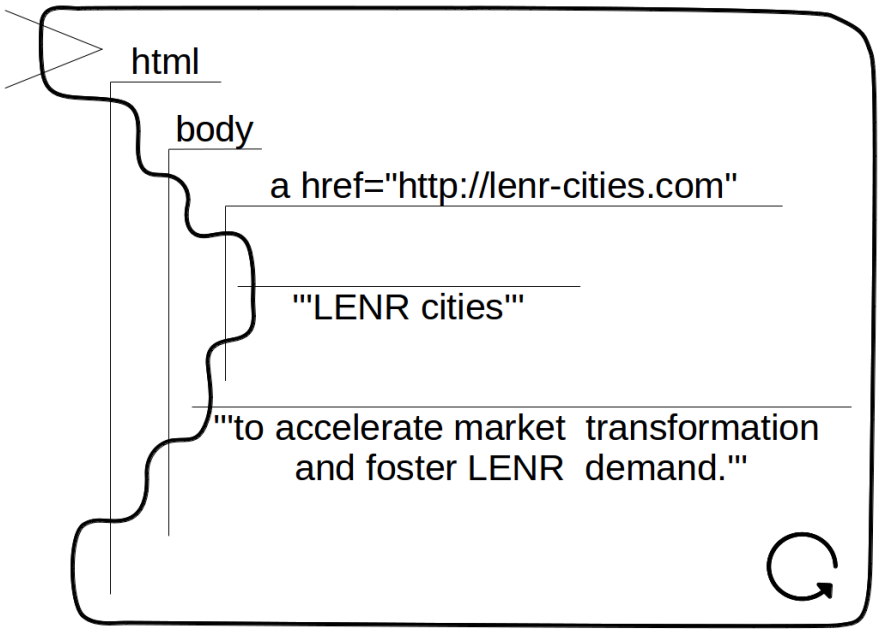


Figure 3 : Document Object Model

Le V en haut à gauche désigne l'endroit par lequel il faut lire le document, il correspond au nœud « *document* » du DOM. Nous l'appelons **point de lecture**.

Pour aller plus loin

Nous représentons les nœuds texte par des traits avec une seule intersection, tels que peer(x)=x, ce qui ne viole pas la condition de convolution. A l'appui de cette interprétation, dans l'API SAX, si les balises sont traitées par deux appels corrélés (startElement, endElement), les textes n'ont qu'un seul appel (characters). Dans cette même optique, les deux deux branches du V correspondent aux appels startDocument et endDocument. En outre, cette interprétation permet de positionner le point de lecture n'importe où sans incohérence. Avec des commentaires à la place des textes – exclus des nœuds fils du nœud document par une règle *ad hoc*, ceci produit des fichiers XML licite.

Nous allons maintenant faire abstraction du contenu précis d'un document pour traiter de la question des actions. Pour faire cette abstraction, nous allons introduire le concept de variable, concept qui ne peut avoir d'interprétation simple en théorie ensembliste des graphes et qui est tout à fait licite dans la formulation modale. Une variable est un trait comme les autres, lorsque de la valuation, ils se mappent sur les nœuds adjacents. Ceci est illustré dans la figure 4.

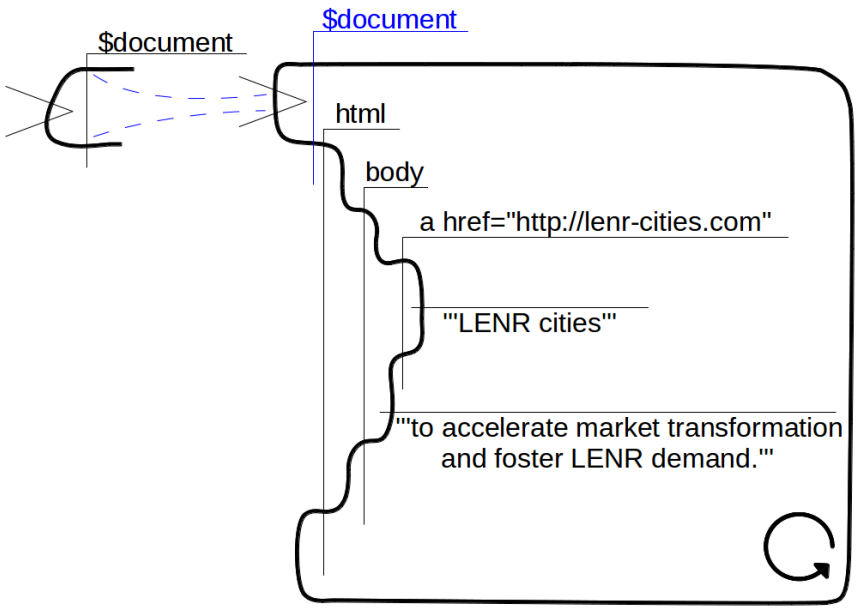


Figure 4 : Valuation d'une variable

Muni de cette capacité de définir des classes de diagrammes par des diagrammes munis de variables, nous allons représenter les actions. Les actions ont la propriété remarquables d'être réversibles, sans réversibilité pas d'édition possible, la capacité de pouvoir défaire et refaire est indispensable à tout document et fait partie de toute API permettant de les éditer. Cette réversibilité dans le schéma que nous définissons est représenté par un trait entre deux bulles.

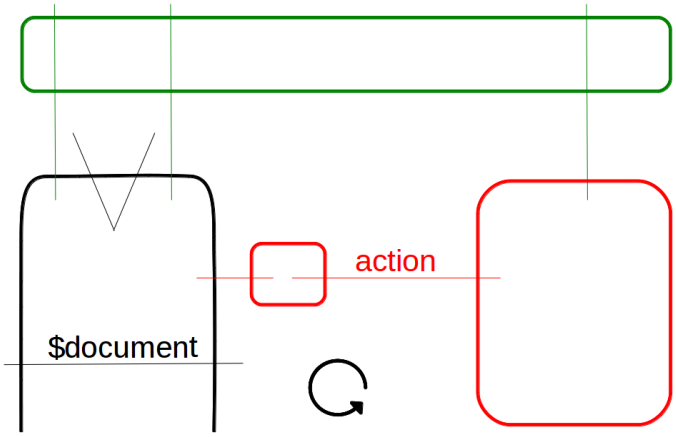


Figure 5 : Action sur un document

Cette action se décompose en deux mouvements, d'abord l'action sur le modèle en **rouge** puis la mise à jour de la vue en **vert**.

Dans le modèle sur lequel nous travaillons depuis l'été 2011, le document vu est lui-même le résultat d'un calcul, l'application d'une feuille de style XSL sur un document XML. La représentation dans le modèle du calcul d'une action est donnée par la figure 6. La partie noire du diagramme représente la situation initiale, le document *Vu* résulte du calcul de l'opération de *$style* sur *$modèle*.

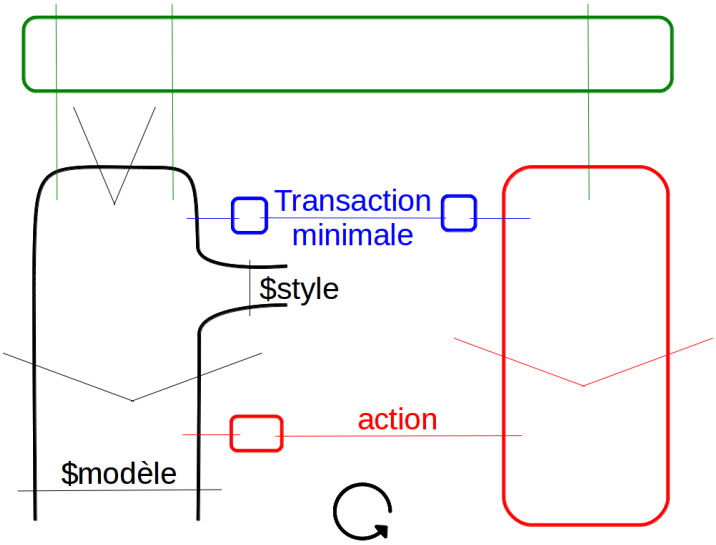


Figure 6 : Action sur un document calculé

Les autres couleurs détaillent le processus de calcul de l'action :

- **rouge** : une action a lieu sur le document
- **bleu** : la feuille de style est appliquée sur le document résultat, le résultat de ce calcul est transformé en une action minimale par l'utilisation d'un différentiateur.
- **vert** : le nouvelle vue est rendue sur la base du document amendé par la transaction différentielle.

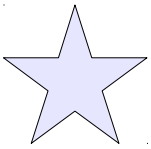
Pour aller plus loin

Nous avons profité de cet exemple pour montrer comment la formulation modale permet de raconter une transformation de diagramme étape par étape, sachant que tout ceci repose sur des calculs bien réels, et que tout ce qui détaillé ici peut être outillé et automatisé. Ce ne sont pas des dessins, mais d'authentiques documents, manipulables par des machines.

Nous avons réalisé une première implémentation de ce principe en 2011 en Java pour le compte de la société SCILAB ENTREPRISE qui voulait moderniser la gestion de ses préférences. Nous avons porté ce code en 2012 en Python/Qt en réalisant un forfait pour le compte de la SAS HYGIE, récupérant par là la propriété sur le code. Ce moteur a été baptisé CARGO et a servi à réaliser deux applications pour le compte de la société SILKAN, en conservant la propriété des codes clés.

En 2015, en prévision de notre collaboration avec LENR-cities, nous avons réalisée une version HTML5 appuyé sur un serveur CARGO à base de WebSocket. Cette dernière version a des capacités proprement stupéfiantes en terme de création d'applications web, rapide à développer, et simple à maintenir. Il suffit d'apprendre des bases l'HTML, mais en revanche de développer une connaissance approfondie en XSL et en CSS pour réaliser des applications en mode SaaS. En outre, la gestion du modèle reprenant tous les acquis de la formulation modale, il est possible de réaliser simplement des supports de collaboration très sophistiqués pour des coûts très limités, sans commune mesure avec les développements réalisés sans cette technique directement en Javascript.

En fait, la technique d'analyse différentielle fonctionne comme un générateur de programmes Javascript évalués à la volée par le client HTML5 (avec la commande *eval*). Toute la programmation Javascript est ainsi automatisée et réalisée par une machine.



Les économies d'exploitation et la simplicité de maintenance de nos systèmes à base d'analyse différentielle relèvent de cette délégation raisonné à la machine d'un travail répétitif et source d'erreurs.

Édition sans duplication

Nous allons maintenant nous préoccuper de réaliser un système d'édition de document sans copier/coller, où, plus exactement, cette opération est remplacée par une autre appelée *clip&view* qui remplace la duplication des données par un lien b-directionnel entre a source et la cible.

Soit deux textes sans formatage – les formatages apporté dans ce document sont à des fins d'éclairage. Dans la situation étudiée, l'auteur du deuxième texte veut recopier une partie digne d'intérêt dans le premier :

> « il était une fois un texte qui avait une partie digne d'intérêt »
>
> « Et un autre texte qui voudrait citer »

Le copier-coller standard fonctionne sur tous les éditeurs de texte, vous sélectionner « *une partie digne d'intérêt* », vous la copier, et la coller à la fin du deuxième texte :

« il était une fois un texte qui avait une partie digne d'intérêt »
« Et une autre texte qui voudrait citer une partie digne d'intérêt »

Le résultat est qu'aucun des textes ne conserve l'information que la partie digne d'intérêt provient du premier texte, privant l'auteur de la partie digne d'intérêt de tout droit sur le deuxième texte.

Pour réaliser le fonctionnement *clip&view*, la première opération-clé intervient au moment de la « *copie* », qui devient l'identification par deux balises identiques de la section digne d'intérêt :

« il était une fois un texte qui avait » **Clip** « une partie digne d'intérêt » **Clip**
« Et un autre texte qui voudrait citer »

La deuxième opération-clé intervient au moment du « *collage* », qui devient l'ajout de deux balises permettant de signifier le lien à double-sens :

« il était une fois un texte qui avait » **Clip** **View** « une partie digne d'intérêt » **Clip**
« Et un autre texte qui voudrait citer » **View**

La figure 7 détaille ces deux étapes avec notre notion de réécriture de graphe.

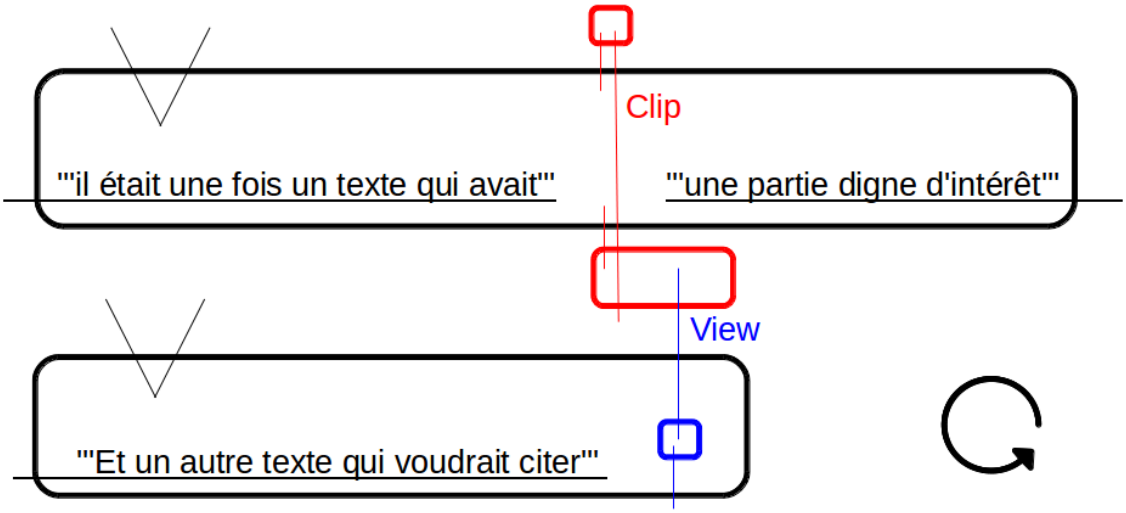


Figure 7 : Clip & View

Seule l'image de la partie digne d'intérêt est échangée entre la source et la cible mais la donnée est conservée par son propriétaire. Nous avons substitué au « copier-coller » défectueux notre opération « *Clip&View* ».

Mais que se passe-t-il si le deuxième auteur souhaite contribuer à la partie digne d'intérêt ?

Nous avons déposé un brevet sur cette question : ce brevet stipule que toute action sur la partie dupliquée a le même effet que s'il avait lieu directement sur l'original, sauf que l'original reste intègre. Repartons de la situation finale de l'encart précédent :

> « il était une fois un texte qui avait » **Clip View** « une partie digne d'intérêt » **Clip**
>
> « Et un autre texte qui voudrait citer » **View**

Le « *partage global* » fonctionne comme si l'auteur « *voyant* » le « *clip* » avait les mêmes droits que le détenteur initial, et s'il désire modifier la partie digne d'intérêt, il le peut – jusqu'à pouvoir lui ôter son intérêt :

> « il était une fois un texte qui avait » **Clip View** « une partie digne d'intérêt rendue caduque » **Clip**
>
> « Et un autre texte qui voudrait citer » **View**

Remarquez que la situation s'est arrangée par rapport au copier-coller car au moins le premier auteur pourrait *en théorie* détecter la destruction involontaire de son œuvre et prendre des mesures adéquates. Mais comme rien ne lui permet de détecter automatiquement la modification, et à moins de lire ou faire lire ses œuvres complètes à date régulière, la perte d'intérêt de son œuvre est irrémédiable.

Pour remédier à cette situation il faut marquer la modification et faire du « *partage partiel* » [Gradit-2010] :

> « il était une fois un texte qui avait » **Clip View** « une partie digne d'intérêt » **Insert** « rendue caduque » **Insert Clip**
>
> « Et un autre texte qui voudrait citer » **View**

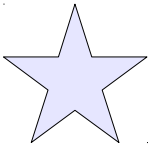
Ce qui signifie que d'une façon ou d'une autre, l'original conserve la trace de toutes les modifications réalisée sur l'œuvre partagée ce qui correspond au modèle d'édition de document présenté dans la section précédente.

Ce modèle, présenté ici de façon succincte permet d'envisager un mode d'édition partagé où l'ensemble des contributeur demeure propriétaire de l'ensemble de ses contributions, même si cette contribution est un collage intelligent d'autres contributions.

Méthodologie

Nous avons exposé les principes d'une plate-forme d'édition de document qui permet à tous les contributeur de demeurer propriétaires de leur contribution. Pour réaliser cette plate-forme nous disposons d'ores et déjà des éléments constituants suivants :

- **Un modèle document** capable de supporter une opération de citation sans duplication
- **Un modèle de gestion des modifications** du document
- **Un moteur informatique** permettant d'exploiter ces conceptions dans le cadre de l'utilisation via un navigateur web du commerce et un serveur dédié.

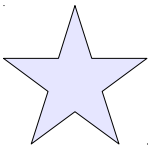


L'ensemble de ces composants partageant le même cadre algébrique de conception appelé « *formulation modale* » et largement introduit dans ce document.

Le développement d'une telle plate-forme se déroule en trois phases :

- **Prototypage/Spécification** : une première version du système est réalisé en même temps que la spécification du système. A l'issu de cette phase, un prototype plus être déployé auprès d'utilisateurs-clés qui peuvent émettre des demandes de correction ou de modification ou d'évolution.
- **Consolidation/Conception** : pendant cette phase, l'expérience des utilisateurs-clés permet de fixer la spécification dans sa forme définitive, et au prototype d'acquérir une certaine robustesse. Certaines fonctionnalités qui ont été écartées de la phase de prototypage sont développées. A l'issue de cette phase le produit est opérationnel et peut être déployé sans contrainte, seulement son usage n'est pas encore validé pour la production.
- **Industrialisation/Validation** : cette dernière phase permet de calibrer le système sur des problèmes à taille réelle avec toutes les problématiques de performances liées à cette situation. Le système est

validé selon les normes applicables. A l'issu de cette phase, le système est déclaré bon pour le service et entre en production.



Par expérience, chacune de ces phases dure et coûte autant. Si la prévision de charge sur la première phase se trouve invalidée par les faits, les partenaires peuvent ainsi ajuster les volumes des phases suivantes en bonne intelligence.

La documentation comporte en général cinq documents majeurs :

- **Une spécification du besoin** : décrit par un jeu d'exigences atomiques, précises et testables l'ensemble des fonctionnalités, interfaces et performances su système.
- **Une conception générale** : décrit l'ensemble des composants du système, leur organisation et décrit l'ensemble des codes sources mis en œuvre dans sa réalisation.
- **Un manuel d'utilisation** : décrit la prise en main par les utilisateurs finaux. Ce document est largement illustré par des copies d'écran, et sa consultation hypertexte est privilégiée.
- **Un manuel de référence** : décrit les modes d'action des auteurs (non-développeurs capable d'ajouter des fonctions simples) et des administrateurs du système.
- **Une main courante** : décrit l'ensemble des demandes de correction, de modification et d'évolution émis par les utilisateurs ou les experts (personnes garantes de l'intention du système) ainsi que des modes d'intégration de ces différents événements dans le système.

Par le présent document, nous affirmons détenir l'ensemble des connaissances nécessaires pour spécifier, établir et exécuter les plans d'une plate-forme d'édition collaborative avec titre de propriété sur les contributions. En outre, nous sommes en mesure de réaliser ce développement pour des coûts nettement inférieur au marché avec un volume total de réalisation largement inférieur à 100k€ jusqu'au déploiement de la première version industrielle. Enfin, nos processus qualité permettent d'assurer à chaque instants aux donneurs d'ordre et aux expert une visibilité optimale sur la tenue des

développement et l'avancement des travaux.

Depuis 2012, nous avons évalué avec Michel Vandanbergue les possibilité offertes par un système d'édition documentaire sans duplication. Le résultat de ces analyse tend à montrer qu'un donnée non duplicable est comparable à un capital et une citation à une transaction financière. Il s'avère, sans que cela forme encore une preuve, que les transactions financières et les actions sur les document sont les deux exemples connus de traits dans la formulation modale. Lors de la conférence de Milan, j'ai exposé la notion de « *private commons* » et la nécessité de disposer d'éléments de négociation non-orientés pour être en mesure de négocier lorsqu'un actif se présente dans le système sa répartition entre les ayants-droit. A ce jour, je pense que les éditions sans duplications forme un candidat à étudier comme base de négociation. Il demeure nécessaire de disposer d'un langage juridique permettant de regrouper les actions en transactions sans présager de l'occurrence de l'apparition d'actifs, mais les citations forment une base indiscutable de cette négociation.

Bibliographie

[Gradit-2014] Travailler sur les réseaux

[Gradit-2010] Brevet de partage partiel d'application