Nancy-Université

maison des sciences de l'homme lorraine

Loria Laboratoire lorrain de recherche en informatique et ses applications

Rapport de projet tutoré :

Interface et vérification pour la modélisation de conversations schizophréniques

Tuteurs : Amblard Maxime, Michel Musiol et Manuel Rebuschi

Réalisé par : Cédric Beuzit-Laboz & Joffrey Mougel

31 mai 2011

Master Sciences Cognitives et applications : 1ère année.

Table des matières condensée

Index des tables

Index des illustrations

Table des matières détaillée

Introduction

Ce projet s'inscrit dans le cadre du projet DiaRaFor de la MSH, traitant dialogues, rationalités et formalismes. Il se trouve au croisement de la logique, la psychologie, l'épistémologie, la linguistique et l'informatique. Il a pour but d'analyser et de modéliser des conversations schizophréniques et de permettre, à terme, un certain nombre de traitement et de vérification sémantique.

Suite aux recherches antérieures à ce projet sur les différents formalismes existant et les outils de modélisation adéquats dans l'étude de discours, il a été convenu d'utiliser une adaptation modifiée (augmentée) de la *Segmented Discourse Representation Theory* (SDRT) et des *Segmented Discourse Representation Structures* qui en découlent, afin de construire les représentations des conversations schizophréniques. En effet, aucun formalisme n'était parfaitement adapté puisqu'aucun jusqu'alors ne permettait de rendre compte des spécificités de ce type de discours (ruptures de références et problèmes d'anaphores non solvables notamment).

La tâche qui nous a été confiée lors de ce projet est de créer une interface visuelle permettant de construire un document dans un format permettant de manipuler informatiquement ces données, et d'en créer la représentation graphique.

Avec cet outil, il doit donc être possible de construire à partir d'un texte de référence un document informatique structuré, d'y effectuer un certain nombre de traitement, et enfin d'en générer une représentation graphique automatique, exportable sous divers format.

Après avoir présenté la SDRT, le formalisme qui sert de base à celui utilisé pour ce projet, nous décrivons dans ce rapport notre analyse de la demande, notre démarche de conception et de réalisation du système, puis concluons par l'état d'avancement de l'outil par rapport à a demande initiale, et quelques perspectives d'évolutions potentielles, à court terme comme à long terme.

Étude Préliminaire

« Discourse Representation »?

Dans le cadre de ce projet, nous avons commencé par étudier ce que sont les DRT et DRS (*Discourse Relation Theory / Structure*), SRDT et SDRS (*Segmented / Discourse Relation Theory / Structure*) qui sont les briques de base du formalisme développé pour répondre aux spécificités des conversations schizophréniques.

Segmented Discourse Representation Theory (SDRT)

C'est une théorie visant à analyser le discours au-delà d'un simple suite de phrases, elle permet de mettre en évidence la structure plus ou moins complexe de ce discours par le biais de relations rhétoriques : elles s'apparentent aux actes de langage (leurs forces illocutoires) que l'on retrouve en pragmatique. Elles relient les contenus sémantiques des différents segments de phrases de manière hiérarchisée. La SDRT se trouve à la rencontre de deux approches différentes du discours :

- La sémantique dynamique (issue de la philosophie du langage)
- L'analyse du discours (issue de la linguistique & l'informatique)

La sémantique dynamique, ayant pour signification une fonction de contexte à contexte, dépasse le cadre de la sémantique formelle, qui elle n'avait pour signification qu'une fonction entre mondes possibles.

La sémantique formelle distingue le sens et la structure de l'ensemble des entités composant une proposition grâce à l'analyse de ses conditions de vérité. La deuxième partie du traitement sémantique prend en compte les anaphores, la portée des quantificateurs et les références partagées par les deux interlocuteurs qui nous permettent d'interpréter le discours en contexte et nous apporte le coté dynamique.

L'analyse du discours, quand à elle distingue la structure d'un discours ainsi que les relations liant ses parties; elle nous sert à arranger les segments linguistiques qui le compose, d'une façon non linéaire, en un ensemble cohérent. La représentation crée avec cette méthode dépend de 4 axiomes, d'après Busquets, Vieu et Asher (2001) :

- Tout discours cohérent à une structure
- Il est possible de le formaliser
- Une structure formelle permet de rendre compte de différents problèmes discursifs
- Il existe un ensemble de relations de cohérence qui rend possible l'interprétation d'un discours

La structure est représentée sous la forme d'un arbre dans lequel les feuilles sont des segments minimaux et les nœuds des segments complexes. Les liens entre ces segments correspondent aux relations de sous-spécification rhétorique (par ex : narration, question, …). Les segments ouverts (tous les sites qui sont disponibles pour introduire une nouvelle relation) sont situé sur la frontière droite. Ils contiennent aussi les référents de discours nous permettant de résoudre les anaphores.

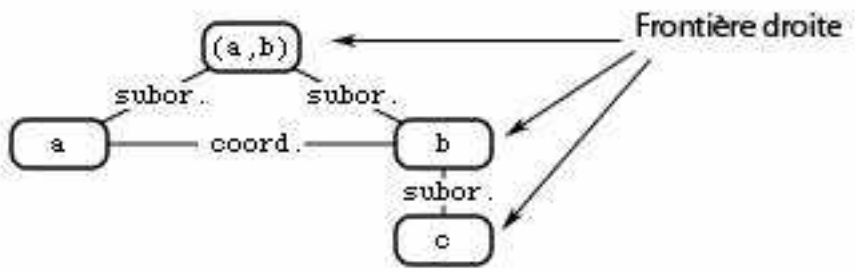


Illustration 1: Frontière droite d'une SDRS

Segmented / Discourse Representation Structures (DRS / SDRS)

Les Segmented Discourse Representation Structures (SDRS) correspondent à la structure même des représentations ; soit de la représentation globale du discours, soit d'une de ses parties (comme à l'intérieur d'un *topic* par exemple).

Les segments minimaux ne peuvent avoir comme structure qu'une DRS car ils ne peuvent contenir d'autres segments contrairement aux segments complexes, qui eux, ont une SDRS comme structure.

Une SDRS contient au minimum, une relation de discours reliant deux DRS.

Les différents types de relations sont "coordonnantes", repérable par des traits horizontaux, ou "subordonnantes", repérable par des traits verticaux ou oblique (dans le cadre d'une intervention de type « question », ou autre, amenant une réponse).

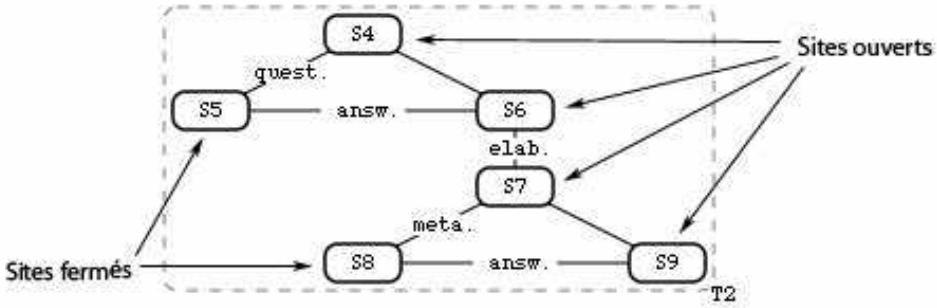


Illustration 2: SDRS : Sites ouverts, sites fermés

Cette théorie des représentations discursives segmentées nous permet donc de mettre en relation une analyse vériconditionnelle (DRS) de nos segments, afin de connaître le sens de chaque partie de notre dialogue, avec une analyse rhétorique distinguant la nature des relations entre toutes ces unités de sens de manière à en savoir plus sur la signification globale du discours.

Les « conversation schizophréniques »?

Il s'agit en réalité plus de discours que de véritables conversations : le psychologue étant plus là pour entretenir le discours que pour l'orienter.

Il n'est donc pas étonnant que la majeure partie des interventions du psychologues soient courtes, et souvent de type phatiques, ou des questions permettant de relancer la narration du patient.

Les propos tenus par un patient schizophrène peuvent s'avérer délicat à analyser, et d'autant plus à formaliser, dans la mesure où, contrairement à un discours conventionnel, les anaphores, ruptures référentielles et thématiques ne sont pas toujours signalées par le locuteur.

Par exemple, il est possible pour un patient schizophrène d'évoquer indifféremment deux références distinctes, mais présentant des similarités (par exemple le prénom d'une personne) sans le mentionner. Ce qui rend les anaphores toutes particulières à résoudres.

Afin de formaliser de tels actes de langages, il est nécessaire d'étendre les SDRS pour rendre compte de ces sauts référentiels et thématiques (ou des doutes que l'analyste aura sur la référence de telle ou telle partie du discours).

Analyse du besoin

Le but du projet est de développer un outil informatique permettant, dans un premier temps, de créer, manipuler et visualiser des relevés de conversations schizophréniques, dans ce formalisme dérivé des SDRS évoqué précédemment (c'est à dire permettant de rendre compte de ruptures référentielles et thématiques).

Idéalement, cet outil servirait de base de travail pour un système ultérieur plus avancé qui comprendrait du traitement et de l'analyse de conversations et discours à un niveau sémantique (à l'aide des DRS), voire pragmatique...

Par ailleurs, il semble attendu qu'outre la manipulation des données, il soit attendu des capacités de conversion de la part de l'application, vers des formats graphiques divers, ainsi que vers le format LaTeX, pour faciliter la publication des travaux des utilisateurs futurs.

Après avoir étudié en substance le fonctionnement de la SDRT augmentée pour les besoins dont il est question (les conversations schizophréniques), en étudiant des représentations de plusieurs exemples, en en réalisant des modélisations et en confrontant nos résultats, nous avons pu dégager plus clairement la problématique.

En premier lieu, il s'agit donc de définir un format pour manipuler de telles données. Le format XML semble s'imposer de lui même, sa structure par essence arborescente et son fonctionnement auto-validant faisant foi. C'est la partie centrale de la phase d'analyse, et de conception, car c'est sur le format définit que s'appuie le reste de nos travaux (y compris l'application développée).

Ensuite, il s'agit d'étudier les choix qui s'offrent à nous en matière de technologies, pour la réalisation pratique d'un tel projet. Les plateformes et langages informatiques ne manquant pas, c'est une phase délicate qui n'est pas à négliger... Plusieurs critères nous aideront à déterminer nos choix finaux, parmi lesquels s'ajoutent les contraintes d'ordre temporel, et bien sûr les connaissances que nous avons au préalable de telle ou telle plateforme de développement.

Formats de manipulation des données potentiels

- XML
- SGML
- Format binaire ou texte propriétaire

Les environnements de développement potentiels

- C/C++, associé à une bibliothèque d'interface graphique (par exemple wxWidgets ou GTK+)
- Java et sa bibliothèque d'interface graphique
- Python associé à une bibliothèque d'interfaces graphiques (TkInter ou WxPython)
- PHP / Javascript / HTML

Les solutions de rendu graphique potentielles

- Bibliothèque de dessin propre au langage choisi?
- Bibliothèque standardisée (OpenGL, DirectX, etc.)?
- Flash
- SVG
- Canvas HTML 5

Solutions conceptuelles

- Application locale
- Application client-serveur (distante, accessible par navigateur)
- Langage interprété? Compilé?
- Modélisation objet?

Nous allons maintenant faire part des choix que nous avons retenus et tenter de les expliquer.

Choix techniques

Technologies retenues

- Format de donnée XML
- Une application client-serveur en PHP / Javascript / HTML
- Pas d'objet (au début de la démarche, mais nous avons commencé à basculer sur un modèle objet)
- Rendu graphique SVG

Format des données

Nous avons choisi le format XML car c'est un format lisible par l'humain, extensible, qui permet l'élaboration, la définition, de langages structurés de type arborescent, ce qui correspond a priori aux besoins exprimés.

De plus, le XML permet la conception d'application validantes, c'est à dire qu'il est possible d'évaluer en temps réel, pendant la manipulation de données complexes, la validité de l'intégralité de la structure du document en confrontant celui-ci au fichiers de définitions disponibles (DTD, XSD, etc.)

Cela permet de réduire les risque de pertes de données dues à une utilisation non prévue du logiciel qui remettrait en cause l'intégrité de la structure.

Par ailleurs, il existe des langages XML de description graphiques (eg. SVG), ce qui devrait pouvoir permettre une conversion du document en graphique relativement simplifiée, ainsi que des possibilités assez avancées pour assigner des styles d'affichage personnalisé (CSS).

Format graphique

Nous avons choisit le SVG, car c'est un langage XML permettant de réaliser de graphiques vectoriels, car c'est un format ouvert qui n'est pas spécifique à une plateforme, et car travaillant en XML, l'intégration devrait en être facilitée.

De plus, le SVG est également personnalisable par l'intermédiaire de feuilles de styles CSS, ce qui pourrait permettre, à terme, d'autoriser l'utilisateur à personnaliser son rendu graphique via une interface spécialisée, sans avoir à toucher à la structure des données. Il semble par conséquent tout indiqué pour ce type de besoin.

Enfin, c'est un format potentiellement dynamique et évènementiel : il est possible d'introduire de l'interactivité à l'aide de code Javascript, ce qui pourrait être profitable dans un développement futur de l'application (par exemple pour afficher les détails d'un segment si l'utilisateur clique dessus, voire la modification dynamique de la structure).

Malheureusement, nous avons eu quelques soucis de compatibilité : le SVG n'étant pas si universellement reconnu qu'il ne le pourrait. Nous avons pu finalement pallier à ce défaut en utilisant une bibliothèque Javascript, « SVGWeb », permettant l'affichage du SVG sur les plateformes ne le supportant pas, par l'intermédiaire d'un visualisateur Flash, totalement transparent pour l'utilisateur.

Choix de l'environnement de développement

Ce choix a été le plus long car les possibilités sont infinies... Nous avons procédés à plusieurs essais avant de nous fixer sur une application PHP / Javascript.

- Ce qui a été déterminant dans ce choix, au final, est les connaissances que nous avons des différentes plateformes évoquées.
- Une application client-serveur à plusieurs avantages :
- Compatibilité universelle : un simple navigateur web est suffisant pour accéder à l'application, quelque soit la plateforme de l'utilisateur (Windows, UNIX-like, MacOS X, etc.)
- Accessible de partout, pour peu qu'elle soit installée sur un serveur distant, et que l'utilisateur dispose d'une connexion internet
- Possibilité d'être exécutée en locale à condition d'installer un serveur web local sur sa machine
- Accès multi-utilisateurs, et sauvegarde des fichiers à distance
- Manipulation avancée et simplifiée des formats XML
- Capacités dynamiques comparables à celles d'une application locale, pour peu que du code Javascript vienne en appui

Cependant, une application client-serveur possède également des inconvénients :

- Graphiquement, les possibilités sont moins souples qu'une application locale. Cet aspect devient négligeable si on considère l'utilisation complémentaire de Javascript pour redonner du dynamisme à l'application.
- Cela oblige l'utilisateur qui souhaite une utilisation locale à installer un serveur web sur son poste de travail.

C'est pourquoi le développement d'un application complémentaire, fonctionnant localement, n'est pas à exclure pour un développement futur du système.

Conception et Réalisation

Langage XML spécifique : SCML

« *Schizophrenic Conversation Meta Language* »

Ainsi qu'indiqué précédemment, l'une des première tâche dans le déroulement du projet a consisté à développer un format particulier pour pouvoir stocker et manipuler informatiquement les données discursives selon un formalisme dérivé de la SDRT, augmentée pour répondre aux besoins spécifiques du cas des conversations schizophréniques.

Ce travail de conception a abouti en un langage XML : le SCML (pour *Schizophrenic Conversation Meta Language*).

Application XML validante

Ceci est concrétisé par un fichier de définitions de type de document XML (DTD, pour *Document Type Definition*) qui sert d'articulation à tout le système : en effet, l'application Schisme est une application XML validante, c'est à dire qu'à chaque opération sur la structure des données, l'arbre XML sera *parsé* pour vérifier, outre une syntaxe correcte (*well-formness*) sa validité selon les définitions de la grammaire contenue dans la DTD.

Le fichier DTD est joint
dans la section « Annexes » du présent rapport

Voici un exemple de fichier SCML minimal valide:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<scml>
    <metadata/>
    <speakers/>
    <topics/>
    <segments/>
</scml>
```

Le fait que l'application soit validante permet de garantir en permanence l'intégrité de la structure des données. C'est fondamental dans un langage structuré de type arborescent, afin de prévenir de la perte de données et de comportements aberrants de l'application en cas de manipulation non désirée et non prévue lors du développement.

Par ailleurs, nous avons également développé un fichier de définition complémentaire de type XML Schema XSD, mais celui-ci et extrêmement permissif (il valide tout à partir du moment où le fichier est en XML bien formé et qu'il contient l'élément racine `<scml/>`). Il est cependant placé dans le projet pour des raisons structurelles (le test est déjà présent dans les fonctions de l'application et si

Le fichier XSD est joint
dans la section « Annexes » du présent rapport

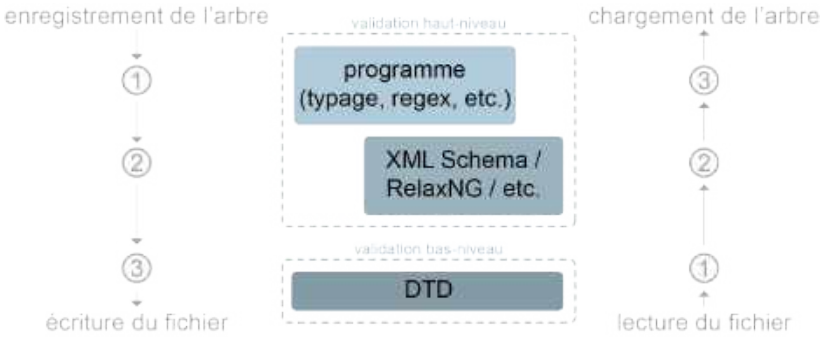


Illustration 3: Processus de valdation XML

Afin d'aboutir à ce résultat, nous avons reproduit plusieurs cas de modélisations déjà établies ou issues d'essais personnels, et ainsi constitué de manière empirique une définition quasiment définitive.

Le processus de construction pas-à-pas du fichier SCML, et de la représentation graphique correspondante, pour le fichier échantillon « Which Dead? »,que nous utiliser tout au long de ce présent rapport, est diponible en annexes

La définition DTD finale, présente dans ce rapport, à subit quelques modifications au long du projet, au fur et à mesure de l'avancement, ou nous avons parfois pu nous rendre compte qu'elle ne permettait pas de rendre compte de certains cas particuliers.

Des éléments spécifiques pour un formalisme spécifique

A titre de rappel, nous avons besoin, outre le fait de pouvoir segmenter un texte en briques minimales connectées les unes aux autres - comme nous l'aurions fait pour définir un langage purement SDRT - de rendre compte des concepts suivants:

- Thèmes conversationnels
- Ruptures (ou de doutes) référentielles
- Ruptures (ou de doutes) thématiques

Des segments « ubiquitaires »

Dans le cas d'une rupture référentielle, il faut pouvoir être en mesure de placer un segment à plusieurs endroits de la structure du discours; en d'autres termes, un segment doit pouvoir s'instancier, et éventuellement se dédoubler. Nous avons donc introduit pour cela les notions d'instances de segments, et de segments « ubiquitaires ».

Ainsi, ce n'est pas sur les segments en eux mêmes que les relations s'effectueront, mais sur des instances de segments.

Terminologie utilisée pour représenter les constituants d'un discours en SCML

Les types de segment:

- **Segments orphelins**: il s'agit des segments qui ne sont pas insérés dans la structure du discours, c'est à dire qui sont présent dans la segmentation du texte, mais pas [encore] instanciés, et qui ne sont par conséquent reliés à rien.
- **Segments instanciés**: Il s'agit des segment « normaux », instanciés une fois, présents à un seul endroit dans la structure du discours.
- **Segments ubiquitaires**: Il s'agit des segments qui contiennent au moins deux instances, et qui, pour rendre compte d'une rupture référentielle dans le discours, ou au mins d'un doute sur la référence perçu par l'utilisateur de l'application, sont insérés à plusieurs endroits du flux discursif.

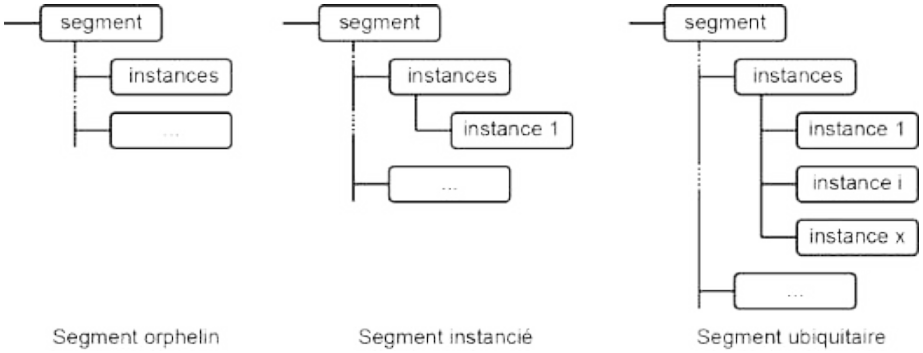


Illustration 4:Représentation des trois types de segments

Les types de relation:

Les relations lient les instances de segments entre elles de manière ascendante, c'est à dire que c'est le descendant qui contient la référence de la relation vers son descendant

- **Relation factice « root »**, *type 0* : c'est une relation factice présente uniquement pour des raisons techniques. Elle représente le commencement d'un nouvel arbre, l'instance de segment qui a une relation de *type 0* est représente donc une nouvelle racine.
- **Relation coordonnée**, *type 1* : sous-spécifications de type narration, continuation, réponses, etc.
- **Relation subordonnée droite / directe**, *type 2* : sous-spécification de type élaboration, explication, etc.
- **Relation subordonnée oblique**, *type 3* : sous-spécification de type question, méta-question, question oui-non, etc.

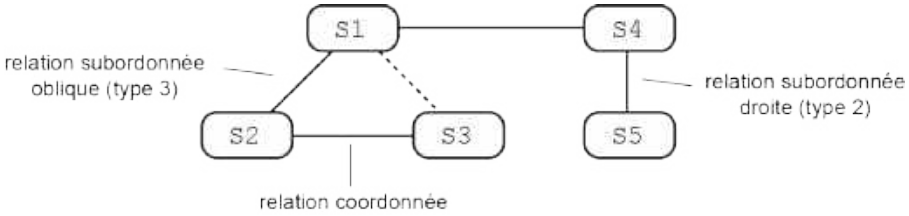


Illustration 5: Représentation de trois types de relations

Détail de la structure d'un document SCML

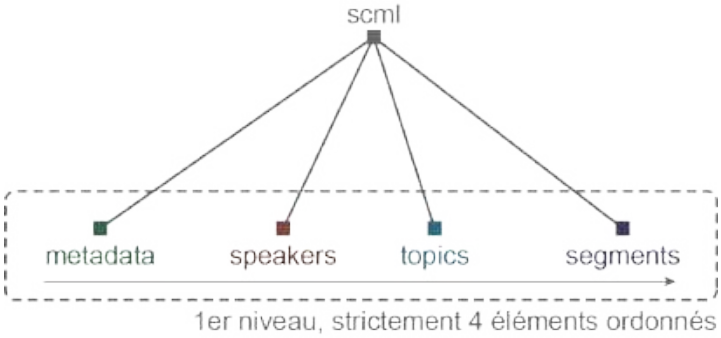


Illustration 6: Structure SCML : éléments racine et premier niveau

L'élément racine contient en séquence, les balises « conteneurs ». Ces quatre éléments doivent être présent impérativement pour que le document soit « DTD valid »

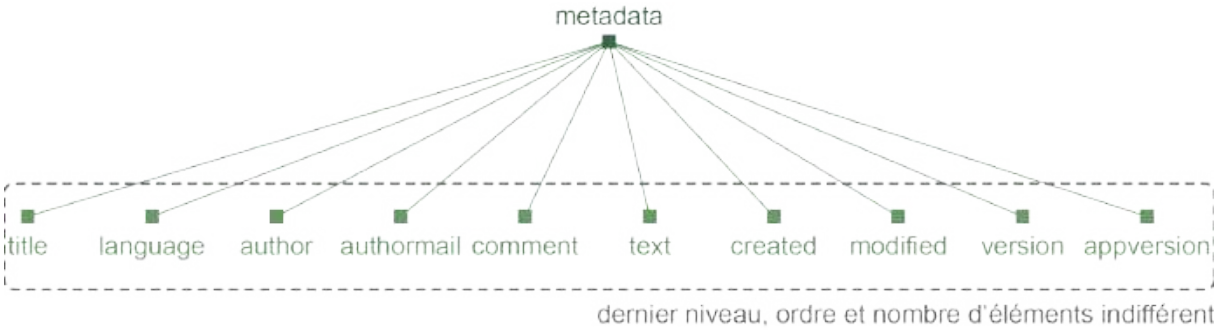


Illustration 7: Structure SCML : metadata

La balise metadata contient toutes les informations concernant le documents et son auteur. Elles sont paramétrables dans l'onglet correspondant de l'application.

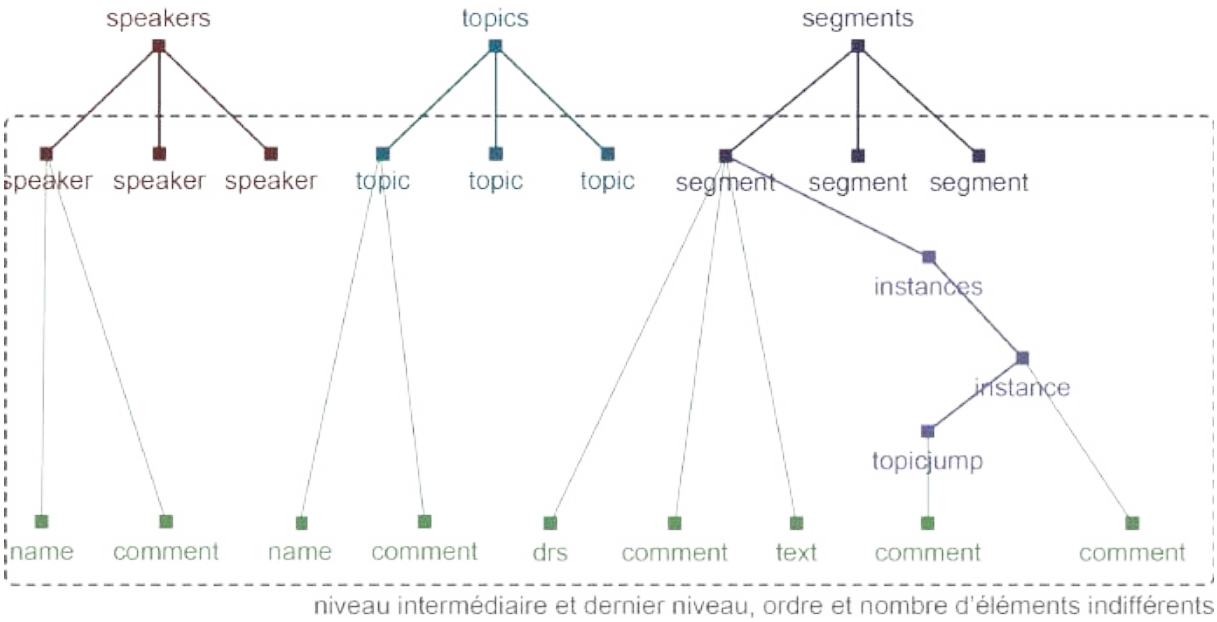


Illustration 8: Structure SCML : éléments du discours

Les conteneurs « speakers » et « topics » contiennent les balises servant à déclarer les locuteurs et les thèmes du document.

Les instances de segments y sont liées par attributs, en spécifiant l'id du locuteur et du thème auxquels elles sont rattachées.

Le format SCML tolère un nombre indéterminé de locuteurs, mais dans l'état actuel du développement, l'application ne sait en gérer que deux.

Les instances sont actuellement placée dans le conteneur « instances », situé lui même dans le conteneur « segments ». A l'origine, elles ne constituaient pas un élément à part entière, mais étaient des attributs des segments. Le fait est que l'ancienne configuration ne permettait pas les traitements désirés, et cette configuration est liée à un remaniement tardif. L'application a été modifiée en conséquence. Il est possible cependant que cette configuration ne soit pas la plus judicieuse, et, quitte à faire des instances un conteneur, il est possible qu'il soit plus intéressant de les remonter au même niveau que les éléments supérieurs (*metadata*, *speakers*, *topics* et *segments*). Ceci devrait être fait lors d'une prochaine étape de ce projet.

Description des attributs des éléments

speaker		
id	*ID*	id du locuteur
topic		
id	*ID*	id du locuteur
segment		
id	*ID*	id du segment
speaker	*IDREF*	id du locuteur associé
instance		
type	*(0|1|2|3)*	type de la relation
source	*IDREF*	id du segment ascendant
instance	*CDATA '1'*	n° d'instance de l'ascendant (1 par défaut)
name	*CDATA #IMPLIED*	nom de sous-spécification (optionnel)
topic	*IDREF #IMPLIED*	id du thème lié (optionnel)
topicjump		
topic	*ID*	id du thème lié

Illustration 9: Description des attributs des éléments SCML

Système développé : Schisme

L'application Schisme est un logiciel client-serveur développé en PHP, un peu de Javascript, et utilise une base de donnée MySQL pour stocker les données des utilisateurs et les types de relations gérées par l'administrateur.

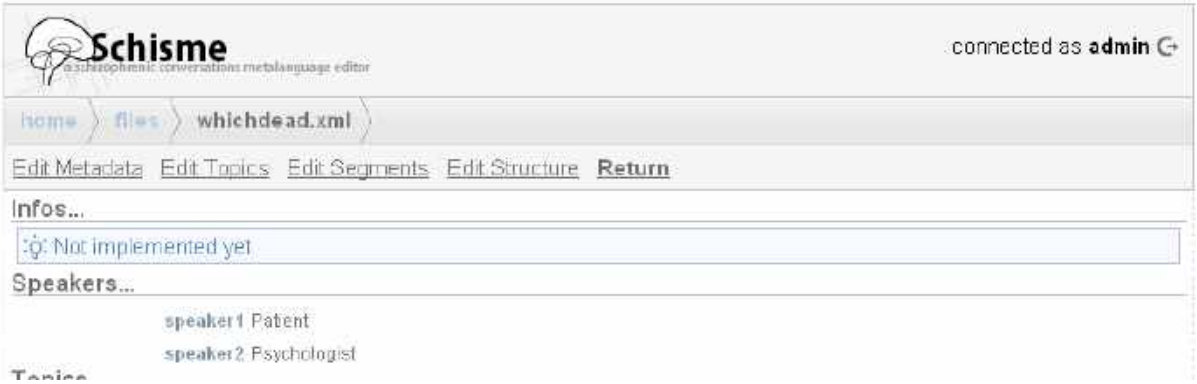


Illustration 10: Apperçu de 'interface graphique

Elle est basée sur un langage XML spécifique, développé pour les besoins du projet : le SCML (Schizophrenic Meta Language), et utilise le SVG pour générer la sortie graphique.

- Cette application permet d'effectuer les opérations sur des fichiers au format SCML:
- Création / upload / stockage d'un fichier SCML
- Téléchargement d'un fichier SCML créé ou modifié
- Affichage du contenu du fichier
- Edition visuelle de la segmentation
- Edition visuelle de la structure du discours
- Affichage d'une représentation schématique au format graphique SVG
- Téléchargement du graphique généré

Structure de l'application

Vue d'ensemble, schéma fonctionnel

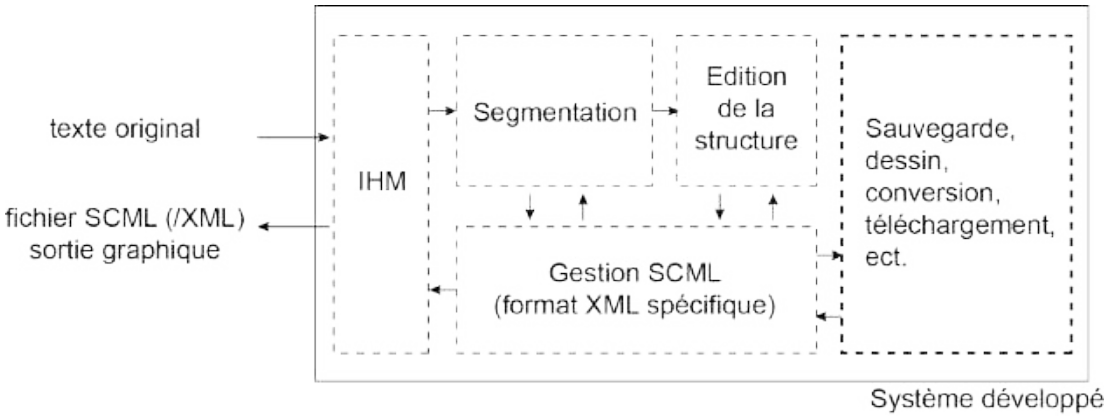


Illustration 11: Vue d'ensemble, schéma fonctionnel de l'application

Plan de l'application

Home (niveau d'accès : 0) : Permet de se connecter, de créer un compte

Fichiers (niveau d'accès 1) : gestion des fichiers du compte (ouvrir, télécharger, effacer, etc.)

→ **Créer un nouveau fichier** (niveau d'accès 1)

→ **Uploader un fichier existant** (niveau d'accès 1)

→ **Ouvrir** (niveau d'accès 1) : fichier en affichage (résumé, métadonnées, graphique)

→ → **Edition des métadonnées** (niveau d'accès 1) : page d'édition des métadonnées

→ → **Edition des thèmes** (niveau d'accès 1) : page d'édition des thèmes

→ → **Edition des segments** (niveau d'accès 1) : page d'édition de la segmentation du fichier

→ → **Edition de la structure** (niveau d'accès 1) : page d'édition de la structure du fichier

Gestion du nom des relations (niveau d'accès 3) : ajouter, supprimer des noms de relations

Administration (niveau d'accès 9) : gestion des utilisateurs

Gestion des droits d'accès

L'application étant multi-utilisateur, un système de droits d'accès est implémenté, et vérifie la légitimité d'un utilisateur sur chacune des pages de l'application.

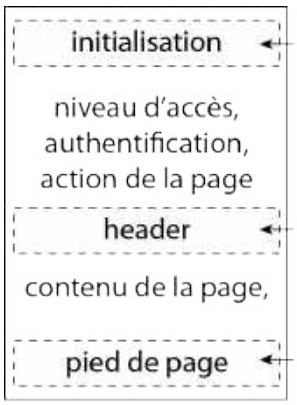
Voici les niveaux d'accès existants:

0	Non connecté
1	Droits 'editeur' (gestion et edition des fichiers SCML de son compte)
2	Droits 'auteurs' (1 + publication d'articles et de news)
3	Droits 'avancés' (1 + 2 + édition des sous-spécifs des relations globales)
9	Droits d'administration (1 + 2 + 3 + gestion des utilisateurs)

Tableau 1: récapitulatif des droits 'accès

Structure d'une page type de l'application

Chaque page de l'application est basée sur le modèle suivant:



`./lib/init.php`
(db connect,
+ authentification,
, etc.)

`./lib/html_header.php`
(en tête html
+ génération navbar
+ génération toolbar)

`./lib/html_footer.php`
(en tête html
+ génération navbar
+ génération toolbar)

Représentation graphique d'un document SCML

Préambule

Un document SCML étant une traduction XML du formalisme – basé sur la SDRT et les SDRS – imaginé pour représenter des conversations schizophréniques, il convient d'établir des définitions pour le représenter graphiquement.

L'illustration suivante donne un exemple de représentation graphique construite à partir d'un document SCML, utilisant tous les éléments existants. Il s'agit de la représentation du texte échantillon « Which dead? », que nous utilisons tout au long du présent rapport.

Le détail complet, détaillé et pas-à pas, de la construction de ce document et de sa représentation graphique est fournit en annexe.

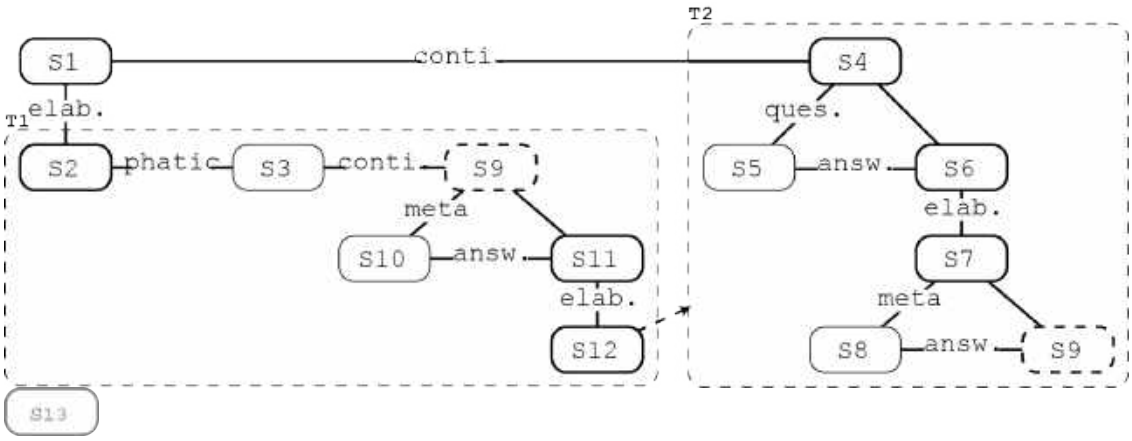


Illustration 12: Représentation SCML en cours d'édition (texte: 'Which dead?')

La suite du chapitre est consacrée au détail des définitions de la représentation graphique, et des principes qui sous-tendent sa construction.

Style graphique

Couleurs

D'une manière générale, en ce qui concerne la documentation technique, il est recommandé de ne pas assigner de sens aux couleurs, afin d'éviter de perdre de l'information en cas d'impression monochromatique. L'affichage de chacun des éléments de notre langage SCML est donc défini dans cette optique : les différences entre les locuteurs, les types et les états de l'élément représenté se font sur la variation de l'intensité du niveau de gris, l'épaisseur et le style du trait, ainsi que sur le remplissage.

Dans l'état actuel du développement, l'application ne gère que deux locuteurs, le patient (speaker1) et le psychologue (speaker2). Les styles graphiques sont donc fixés, codés dans l'application, pour chacun d'eux, en accord avec la règle ci-dessus. Cependant, le langage SCML est capable de gérer un nombre indéterminé de locuteurs. Par conséquent, il est imaginable, et même souhaitable, que l'utilisateur puisse, à l'avenir, définir ses propres styles graphiques, par exemple en intégrant un système d'édition et d'assignement de feuilles de style CSS à l'application.

Textes

Les polices de caractères utilisées sont non proportionnelles (monotype), et les légendes des éléments standardisées ce qui nous permet de maitriser la taille des textes dans la représentation.

Caractéristiques graphiques des éléments de la représentation

Le langage SCML définit un certain nombre d'éléments repris pour construire la représentation graphique. Voici l'inventaire de ces éléments :

- Segments *orphelins* / *instanciés* / *ubiquitaires*
- Relations de *type 0* (nouvel arbre / nouvelle racine)
- Relations de *type 1* (coordonnées) / *type 2* (subordonnées directes) / *type 3* (subordonnées obliques)
- Thèmes
- Ruptures thématiques

Les 3 types de segments sont disponibles pour chacun des locuteurs. Il est nécessaire de pouvoir aisément distinguer visuellement tant les différents locuteurs que les différents types.

Segments

Chaque segment est représenté par un rectangle aux bords arrondis, contenant un texte représentant son identifieur dans le discours segmenté. Voici les caractéristiques graphiques adoptées pour représenter les types possibles pour chaque locuteur:

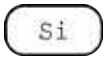
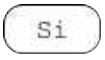
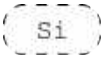
Identifieur du locuteur	Type	Représentation	Caractéristiques
speaker1	orphelin		Trait épais, plein, gris
	instancié		Trait épais, plein, noir
	ubitquitaire		Trait épais, pointillé, noir
speaker2	orphelin		Trait fin, plein, gris
	instancié		Trait fin, plein, noir
	ubiquitaire		Trait fin, pointillé, noir

Tableau 2: Caractéristiques graphiques pour la représentation des segments

Relations

Dans les définitions SCML, il existe 3 types de relations (plus une, factice, de *type 0* qui définit un nouvel arbre). Elles correspondent à un type d'intervention sur le plan rhétorique, et sont sous-spécifiées par un nom au format standardisé affiché au centre du trait.

Type	Nom	Représentation	Exemples de sous-spécifications
0	Nouvel arbre / nouvelle racine	N/A	N/A
1	Coordonnée		
2	Subordonnée directe		
3	Subordonnée oblique		

Tableau 3: Caractéristiques graphiques pour la représentation des relations

Boites thématiques

Un thème est un ensemble continu d'instances de segments liées au même sujet (cf. p.). Graphiquement, un thème est représenté par une boite rectangulaire pointillée, aux bords arrondis, avec une légende qui représente l'identifieur thématique, et qui englobe l'ensemble d'instances de segments qui lui sont liés.

Représentation	Caractéristiques
	Trait fin, pointillé, noir

Tableau 4: Caractéristiques graphiques pour la représentation des boites thématiques

Taille et disposition des éléments

Disposition générale des éléments

Chaque composante graphique est positionnée séquentiellement sur une grille virtuelle de placement (non affichée) à pas fixe, selon sa position dans la structure SCML. Les éléments orphelins sont affichés sous forme de pile en dessous de la représentation de la structure en cours d'édition.

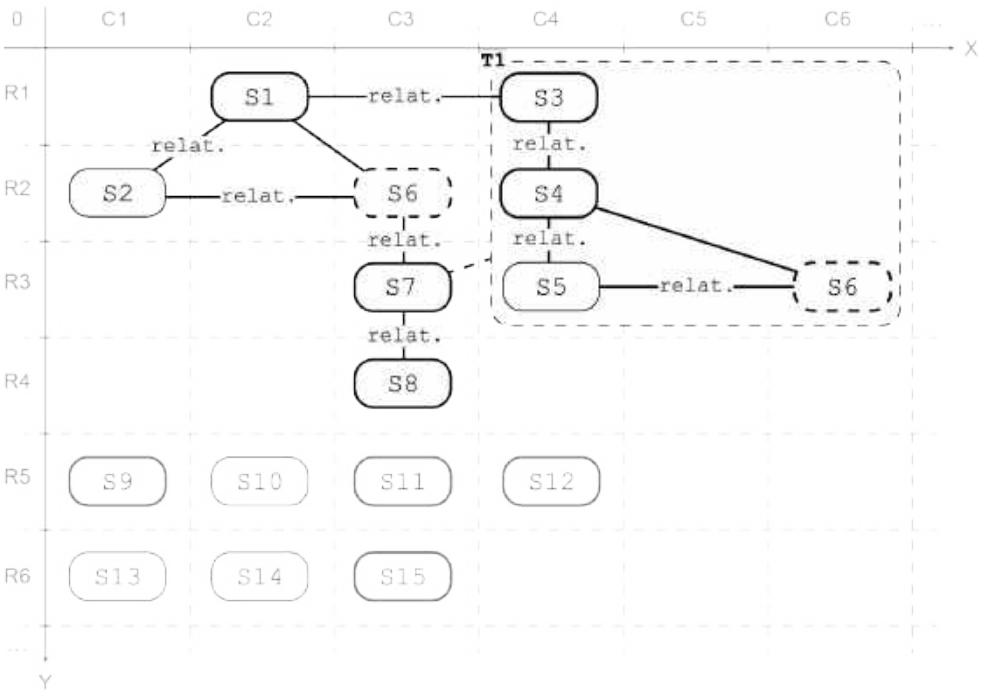


Illustration 13: Disposition générale des éléments de la représentation graphique

Taille des segments

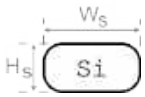
Dans l'état actuel du développement, la taille des segments est fixée 'en dur' dans le code de l'application, sous forme de constante.

A l'avenir, il semble judicieux de permettre à l'utilisateur d'ajuster cette taille pour tous ses graphiques, en proposant un paramètre « taille par défaut », et une fonction de zoom qui lui permettrait de la modifier temporairement en temps réel pour faire varier la taille globale de la sortie graphique.

Nous avons fixé cette taille en fonction des contraintes suivantes:

- Permettre l'affichage horizontal du plus de segments possible

- Afficher l'identifieur du segment de sorte qu'il soit lisible et sans dépassements, même pour un ID à 4 chiffre (*eg.* « S1037 »), avec une fonte monotype 10pts.
- La nécessité d'avoir une valeur multiple de 2 pour éviter les *glitches* graphiques



Largeur W_S (en pixels)	Hauteur H_S (en pixels)
40px	20px

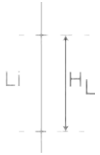
Tableau 5: Taille par défaut des segments

Taille et positions de la grille virtuelle de placement à pas fixe

La grille virtuelle de placement des objets est relative à la taille des segments, ainsi, la taille globale du graphique peut-être modifiée en ne changeant que les valeurs pour ces derniers.

Largeur W_C	Position X_{1Ci}	Position X_{2Ci}
1.5 x W_S	(i-1) x W_C	i x WC

Tableau 6: Taille et position des lignes de la grille virtuelle de placement (en pixels)



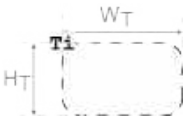
Largeur H_C	Position Y_{1Ci}	Position Y_{2Ci}
2 x H_S	(i-1) x H_C	i x H_C

Tableau 7: Taille et position des lignes de la grille virtuelle de placement (en pixels)

Taille des boites thématiques

Afin d'éviter tout recouvrement intempestif, la taille des boites thématique doit être ajustée en fonction de la taille des segments et de la grille virtuelle de placement : en largeur comme en hauteur, une boite thématique doit être plus grande que l'ensemble des segments qu'elle englobe, et plus petite que les cases de la grille dans lesquelles elle prend place.

La taille des boites thématiques est fixée relativement à la taille des segments.



W_T	H_T
1.25 x W_S	1.75 x H_S

Tableau 8: Taille et position des boites thématiques (en pixels)

Principe de placement des éléments

Nous n'avons malheureusement pas été en mesure de finaliser l'algorithme permettant de dessiner convenablement le graphique d'un document SCML. De ce fait, cette version de l'application ne permet pas la visualisation graphique des documents. Nous espérons y remédier au plus vite.

Positionnement des composants relativement à la grille virtuelle

Chaque segment est centré dans sa case hôte de la grille virtuelle.

Chaque boite thématique est centré sur l'ensemble des cases qu'elle occupe.

Les relations démarrent du centre de l'instance source, et arrivent au centre de l'instance cible (ou de la boite thématique cible en ce qui concerne les ruptures thématiques.

Voici, à titre d'exemple, les formules qui permettent de connaître les coordonnées d'un segment donné, que nous voulons placer dans la case (Ci, Rj) de la grille de placement virtuelle:

X position of the segment S1 in the box (Ci, Rj):

$$X_{1S} = ((i - 1) \times W_C + (W_C / 2) - (W_S / 2)$$
$$X_{2S} = X_{1S} + W_S$$

Y position of the segment S2 in the box (Ci, Rj) :

$$Y_{1S} = ((j - 1) * H_C) + (H_C / 2) - (H_S / 2)$$
$$Y_{2S} = Y_{1S} + H_S$$

Ordre de dessin des composants

Les composants graphiques sont dessinés dans un ordre spécifique, en trois étapes (assimilables à des « calques »), pour obtenir le rendu final.

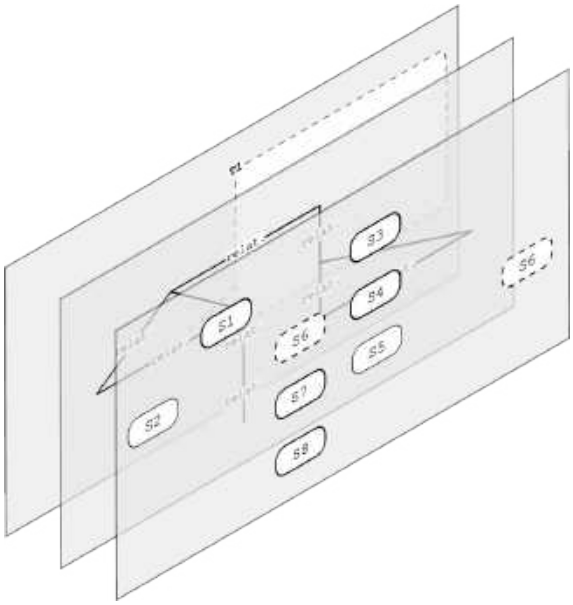


Illustration 14: Ordre de placement des composants graphiques

- **Etape 1**: dessin des rupture thématiques, puis des boites thématiques
- **Etape 2** : dessin des relations
- **Etape 3** : dessin des segments

En dehors de cela, les arbres distincts sont traités séparément (un graphique pour un arbre), et sont affichés côte à côte.

Enfin, les segments orphelins restants sont affichés sous le graphique structurel final sous la forme d'une pile.

Il semble judicieux à l'avenir de permettre à l'utilisateur de paramétrer l'affichage qu'il souhaite obtenir : afficher les arbres différents cote-à-cote ou l'un sur l'autre, afficher ou non les segments non insérés dans la structure (« orphelins »), etc.

Bilan

États des lieux

Nous avons conçu et développé un langage XML, et son fichier de définition, pour manipuler des conversations schizophréniques : le SCML (pour *Schizophrenic Conversations Meta Language*).

Nous avons développé une application client-serveur PHP / MySQL / Javascript, multi-utilisateurs, pour permettre la manipulation de données de conversations schizophréniques : Schisme. Cette application est de type validante (elle valide en temps réel la cohérence de la structure des données manipulées, pour éviter la corruption de données causée par d'éventuelles manipulations de l'utilisateur imprévues lors du développement), et est compatible avec tous les grands navigateurs internet du marché.

Cette application permet de créer un document à partir de rien (ou d'un texte brut), éditer la segmentation de ce document, et manipuler sa structure (les relations qui lient les unités de discours) par l'intermédiaire d'une interface graphique accessible par un navigateur internet. Ces documents sont par défaut stockés sur le serveur (qui peut-être l'ordinateur local de l'utilisateur), et il est possible de les télécharger en local, et d'en *uploader* des nouveaux.

Malheureusement, à ce stade, nous n'avons pas finalisé l'algorithme permettant l'affichage de la représentation graphique de notre formalisme basé sur la SDRT, à cause de difficultés imprévues sur le positionnement des éléments. Nous espérons parvenir à intégrer cette fonctionnalité, qui nous paraît fondamentale, avant la soutenance de notre projet afin d'avoir une application fonctionnelle à présenter.

De la même manière, nous n'avons pas trouvé le temps de finaliser le guide d'utilisation (la première partie, concernant l'installation de l'application, est dores-et-déjà présente dans les annexes. Nous les distribuerons lors de notre soutenance.

Perspectives

Techniques (court terme)

- Terminer / faire fonctionner l'algorithme de rendu graphique SVG.
- Edition / publication de news / articles.
- Administration des utilisateurs (activation et suppression de comptes)
- Edition des préférences utilisateurs, avec personnalisation des noms de sous-spécifications possible individuellement pour chaque utilisateur; une possibilité de pré-remplissage des métadonnées d'un nouveau fichiers avec les informations de l'utilisateur.
- Terminer la transition vers un programme complètement orienté objet, ce qui permettrait une meilleure clarté, structuration et maintenance ultérieure du code. Cela inclut la génération du SVG dans par le biais d'un arbre DOM, la conversion des fonctions de base de donnée au format PDO, et la finalisation des classes spécifiques à l'application.
- Ajout de formats graphiques d'export des fichiers SVG : LaTeX, PNG.
- Génération de rapport PDFs.
- Valider la pertinence du format définit (DTD) des données, par confrontation avec différents cas de figure
- Permettre plus de deux locuteurs, et permettre à l'utilisateur d'assigner et d'éditer des styles spécifiques personnalisés pour chacun d'entre eux (éditeur CSS, etc.)
- Améliorations ergonomiques diverses
- Application multilingue (internationalisation)

Conceptuelles (moyen terme)

- Intégration d'outil(s) de traitement à un niveau sémantique (eg. Boxer, http://svn.ask.it.usyd.edu.au/trac/candc/wiki/boxer, qui permet l'analyse sémantique DRS-based)
- Ajout au dynamisme / à l'interactivité de l'application (manipulation graphique des segments sur la représentation graphique, zoom, etc.)
- Création d'une application locale complémentaire

Théoriques (plus long terme)

- Extension au traitement de conversations du quotidiens? Moins caractérisables comme schizophréniques, bien que pas nécessairement répondant aux règles d'un discours parfaitement construit, parfaitement cohérent...?

Conclusion

Malgré certaines difficultés rencontrées en ce qui concerne sa finalisation, ce projet s'est avéré être d'une grande richesse pour les connaissances qu'il nous a amener a développer, tant sur le plan de l'analyse à partir d'un besoin spécifique à la réalisation d'un projet jusqu'à son terme.

Ce projet nous a fait prendre conscience de l'importance de l'analyse et de la conception dans tout développement, ce qui nous a occupé le plus de temps, notamment dans les cas de figure ou nous avions à corriger une erreur ou un oubli de conception à mi-chemin, ce qui nous faisait corriger la réalisation depuis le début.

L'outil que nous avons finalement développé est fonctionnel, et même si certaines fonctionnalités souhaitées ne sont pas finalisées à ce jour, nous continuons son développement pour présenter un outil le plus complet possible lors de notre soutenance.

Références bibliographiques

- *La SDRT : une approche de la cohérence du discours dans la tradition de la sémantique dynamique* (J. Busquets, L. Vieu, N. Asher, Verbum n°1, 2001)
- *Du rôle de la sous-spécification dans les interactions verbales pathologiques (formalisation à base de SDRT)* (Manuel Rebuschi, Michel Musiol, JSM10)
- *Une analyse des dialogues pathologiques basée sur la SDRT (*Maxime Amblard, Michel Musiol, Manuel Rebuschi, Décembre 2010)
- *Une approche sémantique et rhétorique du dialogue. (Un cas d'étude : l'explication d'un itinéraire.)* (Laurent Prévot, Philippe Muller, Pascal Denis, Laure Vieu)
- *Analyse du dialogue : aspects pragmatiques et rhétoriques* (Jean Caelen, Anne Xuereb, Les Prépublications de la MSH Lorraine n°4, 2010) http://www.msh-lorraine.fr/fileadmin/images/preprint/ppmshl04-2010-02-axe4-CaelenXuereb.pdf
- *Structures sémantiques et pragmatiques pour la modélisation de la cohérence dans des dialogues finalisés* (Laurent Prévot, Thèse, soutenue en janvier 2004. http://www.loa-cnr.it/Papers/laurentPhD.pdf)
- Relations subordonnantes et coordonnantes pour la désambiguïsation du discours (Laurence Delort, avril 2004. http://hal.archives-ouvertes.fr/docs/00/08/15/45/PDF/sdrt_delort.pdf)
- Wide-Coverage Semantic Analysis with Boxer (Johan Bos)

Autres

- SVGWeb : bibliothèque de prise en charge universelle du format graphique SVG http://code.google.com/p/svgweb/
- PHP.net : guide de référence de progrmmation PHP http://php.net/
- W3Schools : références des langages du web (XML, HTML, SVG, Javascript, PHP) http://www.w3schools.com/
- Boxer, bibliothèque d'analyse sémantique DRS-based http://svn.ask.it.usyd.edu.au/trac/candc/wiki/boxer

Annexes

Fiche de projet tutoré / Project Form

Commanditaire / Author

- Maxime Amblard
- Michel Musiol
- Manuel Rebuschi

Interlocuteur UFR (pas forcément le tuteur) / UFR interlocutor

- Maxime Amblard (maxime.amblard@univ-nancy2.fr)
- Michel Musiol (michel.musiol@univ-nancy2.fr)
- Manuel Rebuschi (manuel.rebuschi@univ-nancy2.fr)

Titre du projet / Title of the project

Interface et vérification pour la modélisation de conversations schizophréniques

Description / description

Ce projet veut prendre en compte deux thématiques :

- la modélisation de relations discursives et conversationnelles,
- l'analyse de la pathologie schizophrénique.

Plusieurs théories ont été développées pour modéliser la structure du discours. Le problème général est de définir le bon niveau de description. Dans ce projet, nous utiliserons une extension de la DRT (Discourse Representation Theory) la S-DRT qui construit un arbre de relations discursives et conversationnelles. L'avantage de cette théorie est qu'elle se base sur le calcul de la sémantique compositionnelle et qu'elle tente de rendre compte d'une pragmatique de manière simplifiée.

D'autre part, le laboratoire de psychologie étudie les troubles schizophréniques. On remarque que dans le cadre de l'entretien dirigé, une partie des patients schizophrènes utilise des stratégies discursives non usuelles mais récurrentes. Il s'agit donc de les répertorier et d'en saisir la nature.

Dans le cadre du projet DiaRaFor de la MSH, une étude est conduite sur l'utilisation de propriétés formelles pour modéliser le discours dans le cadre des entretiens entre patients schizophrènes et psychologues. Le but du projet tutoré est de s'inscrire dans ce travail en proposant un outil permettant de visualiser en quoi dans la représentation du discours, le schizophrène ne suit pas une stratégie autorisée. Cette interface doit être dynamique pour aider à la rédaction des structures discursives.

La deuxième partie de ce projet est de proposer des algorithmes de vérification de la bonne formation des discours construits. C'est-à-dire, à partir des structures autorisées dans le dialogue normal, de vérifier automatiquement la validité de la structure construite avec l'interface d'une part et d'autre part de mettre en avant ce qui ne fonctionne pas dans le dialogue du schizophrène.

La troisième partie de ce projet est d'utiliser cette interface pour modéliser des conversations de schizophrènes et de vérifier les algorithmes de bonne formation de la structure du discours afin de proposer une théorie sur la structure des conversations schizophréniques.

Informations diverses : matériel nécessaire, contexte de réalisation / Various information: material, context of realization

Ce projet s'inscrit dans le cadre du projet DiaRaFor de la MSH.

L'implémentation pourra être réalisée en Java ou en python en fonction des compétences du groupe.

Références:

- La SDRT : une approche de la cohérence du discours dans la tradition de la sémantique dynamique, J. Busquets, L. Vieu, N. Asher, Verbum n°1, 2001
- Du rôle de la sous-spécification dans les interactions verbales pathologiques (formalisation à base de SDRT). Manuel Rebuschi, Michel Musiol, JSM10

Livrable et échéancier / Deliverable and schedule

- Implémentation d'une interface graphique de visualisation de relation dans la conversation.
- Implémentation algorithmes naïfs pour la vérification de la structure de la conversation.
- Utilisation de l'interface pour modéliser les relations dans la conversation.
- Analyse des résultats obtenus.
- Rédaction d'un rapport synthétique.

DTD (XML Doc Type Definition)

```
<!-- ############################################################################
# SCHIZOPHRENIC CONVERSATIONS MARKUP LANGUAGE DOCTYPE DEFINITION
# M1 SCA 2010/2011 @CEDRIC BEUZIT-LABOZ @JOFFREY MOUGEL
# NANCY 2 UNIVERSITY - NANCY - FRANCE
# v1.0b (2011-05-15)
############################################################################ -->

<!--#########################################################################
# ROOT
######################################################################### -->
<!ELEMENT scml (metadata, speakers, topics, segments)>
 <!-- optional attributes in order to link an optional XML Schema -->
 <!ATTLIST scml xmlns:xsi CDATA #IMPLIED>
 <!ATTLIST scml xmlns CDATA #IMPLIED>
 <!ATTLIST scml xsi:noNamespaceSchemaLocation CDATA #IMPLIED>
 <!ATTLIST scml xsi:SchemaLocation CDATA #IMPLIED>

<!--#########################################################################
# HIGH LEVEL
#########################################################################-->
<!ELEMENT metadata (title|language|author|authormail|comment|text|created|modified|version|
appversion)*>
<!ELEMENT speakers (speaker)*>
<!ELEMENT topics (topic)*>
<!ELEMENT segments (segment)*>

<!--#########################################################################
# INTERMEDIATE LEVELS
#########################################################################-->
<!ELEMENT speaker (name|comment)*>
 <!ATTLIST speaker id ID #REQUIRED>
<!ELEMENT topic (name|comment)*>
 <!ATTLIST topic id ID #REQUIRED>
<!ELEMENT segment (drs|comment|text|instances)*>
 <!ATTLIST segment id ID #REQUIRED>
 <!ATTLIST segment speaker IDREF #REQUIRED>

<!ELEMENT instances (instance)*>

<!ELEMENT instance (comment|topicjump)*>
 <!ATTLIST instance type (0|1|2|3) #REQUIRED>
 <!ATTLIST instance source IDREF #IMPLIED>
 <!ATTLIST instance instance CDATA "1">
 <!ATTLIST instance name CDATA #IMPLIED>
 <!ATTLIST instance topic IDREF #IMPLIED>

<!ELEMENT topicjump (comment)*>
 <!ATTLIST topicjump topic IDREF #IMPLIED>

<!--#########################################################################
# LOW LEVEL (PCDATA)
#########################################################################-->
<!ELEMENT title (#PCDATA)>
<!ELEMENT language (#PCDATA)>
<!ELEMENT version (#PCDATA)>
<!ELEMENT appversion (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT authormail (#PCDATA)>
<!ELEMENT created (#PCDATA)>
<!ELEMENT modified (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT comment (#PCDATA)>
<!ELEMENT drs (#PCDATA)>
<!ELEMENT text (#PCDATA)>
```

XSD (XML Schema Definition)

```
<!-- ##############################################################################
# SCHIZOPHRENIC CONVERSATIONS MARKUP LANGUAGE XML SCHEMA
# M1 SCA 2010/2011 @CEDRIC BEUZIT-LABOZ @JOFFREY MOUGEL
# NANCY 2 UNIVERSITY - NANCY - FRANCE
# v1.0 (2011-05-02)
############################################################################## -->

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:element name="scml">

	</xs:element>
</xs:schema>
```

Conversions 'texte brut' → 'texte segmenté' → SCML

Which dead?

Voici un exemple détaillé qui suit le processus de conversion d'un texte de conversation brut en fichier XML au format SCML. Il s'agit du texte 'Which dead?'.

Nous partons du texte initial pour constituer pas-à-pas le fichier au format SCML. Nous commençons par définir les interlocuteurs et les thèmes, puis les segments à partir du texte initial ('raw text'). Enfin, nous ajoutons les relations qui lient les instances de segments entre elles.

Il est possible de définir d'autres balises, telles que `<drs>` ou `<comment>` pour chaque segment, mais dans la mesure ou aucun traitement n'est effectué dessus dans l'état actuel d'avancement du projet, nous nous contentons de détaillé les principaux éléments de la structure.

Texte initial

Patient: Oh yeah (↑) and complicated (↑) and it's really very very complicated (→) politics, it's really something when you get into it, have to win or else when you lose, well, you're finished (↓)
Psychologist: Yes
Patient: JCD is dead, L is dead, P is dead uh (…)
Psychologist: So you think they're dead because they lost (↑)
Patient: No they won but if they're dead, it's their disease well it's it's (→)
Psychologist: Yeah it's because they had a disease, it's not because they were in politics (↑)
Psychologist: Yes you think it's because they were in politics (↑)
Patient: Yes, so well yeah there was C too who committed murder, uh huh (→) he was there too, the one in B but well (→) it, that, it's because of politics again

Texte segmenté

Patient	s1	Oh yeah (↑) and complicated (↑) and it's really very very complicated (→)
	s2	politics, it's really something when you get into it, have to win or else when you lose, well, you're finished (↓)
Psychologist	s3	Yes
Patient	s4	JCD is dead, L is dead, P is dead uh (…)
Psychologist	s5	So you think they're dead because they lost (↑)
Patient	s6	No they won but if they're dead,
	s7	it's their disease well it's it's (→)
Psychologist	s8	Yeah it's because they had a disease, it's not because they were in politics (↑)
Patient	s9	Yes I mean (→)
Psychologist	s10	Yes you think it's because they were in politics (↑)
Patient	s11	Yes, so well yeah there was C too who committed murder, uh huh (→)
	s12	he was there too, the one in B but well (→)
	s13	it, that, it's because of politics again

1 → Initialisation XML, élément racine SCML, et metadata SCML

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<scml>
    <metadata>
        <title>Which Dead?</title>
        <language>English</language>
        <author>Amblard - Musiol - Rebuschi</author>
        <authormail>cedric@cblab.net</authormail>
        <comment></comment>
        <created>2011-05-03T13:47:13+00:00</created>
        <modified>2011-05-03T13:47:13+00:00</modified>
        <version>1</version>
        <appversion>1.0b</appversion>
```

2 → Texte initial (texte 'brut')

```
        <text>
            Patient: Oh yeah (↑) and complicated (↑) and it's really very very complicated (→)
            politics, it's really something when you get into it, have to win or else when you
            lose, well, you're finished (↓)
            Psychologist: Yes
            Patient: JCD is dead, L is dead, P is dead uh (…)
            Psychologist: So you think they're dead because they lost (↑)
            Patient: No they won but if they're dead, it's their disease well it's it's (→)
            Psychologist: Yeah it's because they had a disease, it's not because they were in
            politics (↑)
            Patient: Yes I mean (→)
            Psychologist: Yes you think it's because they were in politics (↑)
            Patient: Yes, so well yeah there was C too who committed murder, uh huh (→) he was
            there too, the one in B but well (→) it, that, it's because of politics again
        </text>
    </metadata>
```

3 → Speakers

```
    <speakers>
        <speaker id="speaker1">
            <name>Patient</name>
            <comment></comment>
        </speaker>
        <speaker id="speaker2">
            <name>Psychologist</name>
            <comment></comment>
        </speaker>
    </speakers>
```

4 → Topics

```
    <topics>
        <topic id="T1">
            <name>Political Dead</name>
            <comment></comment>
        </topic>
        <topic id="T2">
            <name>Physical Dead</name>
            <comment></comment>
        </topic>
    </topics>
```

5 → Segmentation initiale

```
<segments>
        <segment id="S1" speaker="speaker1">
            <text>Oh yeah (↑) and complicated (↑) and it's really very very complicated
            (→)</text>
        </segment>
        <segment id="S2" speaker="speaker1">
            <text>politics, it's really something when you get into it, have to win or else
            when you lose, well, you're finished (↓)</text>
        </segment>
        <segment id="S3" speaker="speaker2">
            <text>Yes</text>
        </segment>
        <segment id="S4" speaker="speaker1">
            <text>JCD is dead, L is dead, P is dead uh (…)</text>
        </segment>
        <segment id="S5" speaker="speaker2">
            <text>So you think they're dead because they lost (↑)</text>
        </segment>
        <segment id="S6" speaker="speaker1">
            <text>No they won but if they're dead,</text>
        </segment>
        <segment id="S7" speaker="speaker1">
            <text>it's their disease well it's it's (→)</text>
        </segment>
        <segment id="S8" speaker="speaker2">
            <text>Yeah it's because they had a disease, it's not because they were in politics
            (↑)</text>
        </segment>
        <segment id="S9" speaker="speaker1">
            <text>Yes I mean (→)</text>
        </segment>
        <segment id="S10" speaker="speaker2">
            <text>Yes you think it's because they were in politics (↑)</text>
        </segment>
        <segment id="S11" speaker="speaker1">
            <text>Yes, so well yeah there was C too who committed murder, uh huh (→)</text>
        </segment>
        <segment id="S12" speaker="speaker1">
            <text>he was there too, the one in B but well (→)</text>
        </segment>
        <segment id="S13" speaker="speaker1">
            <text>it, that, it's because of politics again</text>
        </segment>
    </segments>
```

Dans l'état actuel, nous n'avons définit que des segments; ils ne sont pas 'instanciés' ni liés par des relations. Ces segments sont dits 'orphelins'. Voici la représentation graphique correspondante à ce fichier ne contenant que des segments 'orphelins':

Illustration 15: représentation de 'which dead' segmenté, non instancié

6 → Instanciation des segments et ajout de relations

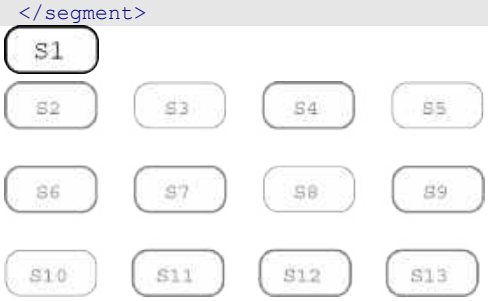
Nous allons maintenant créer des instances des chacun des segments, ce qui aura pour effet de les insérer dans la structure du discours / de la conversation. Ce ne sera plus des segments 'orphelins'.

Lorsqu'un segment se trouve potentiellement à plusieurs endroits de la structure (rupture), il est instancié plusieurs fois, et de vient un segment dit 'ubiquitaire'.

A noter que les choses intéressantes se passent surtout entre S9 et S10 (segment ubiquitaire) et S12 (rupture thématique).

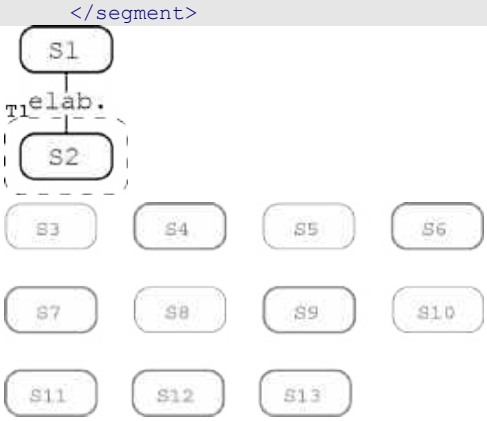
Instanciation de S1

```
<segment id="S1" speaker="speaker1">
    <text>Oh yeah (↑) and complicated (↑) and it's really very very complicated (→)</text>
    <instances><instance type="0"/></instances>
</segment>
```



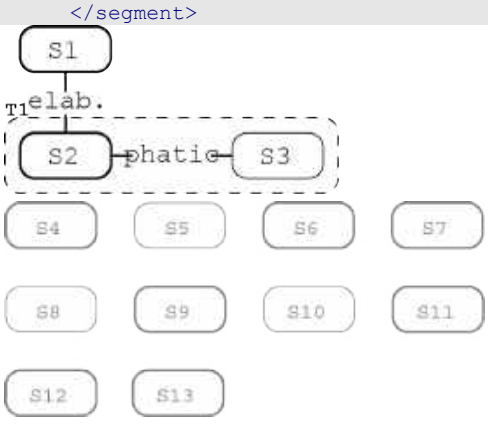
Instanciation de S2

```
<segment id="S2" speaker="speaker1">
    <text>politics, it's really something when you get into it, have to win or else
    when you lose, well, you're finished (↓)</text>
    <instances><instance type="2" source="S1" name="elab." topic="T1"/></instances>
</segment>
```



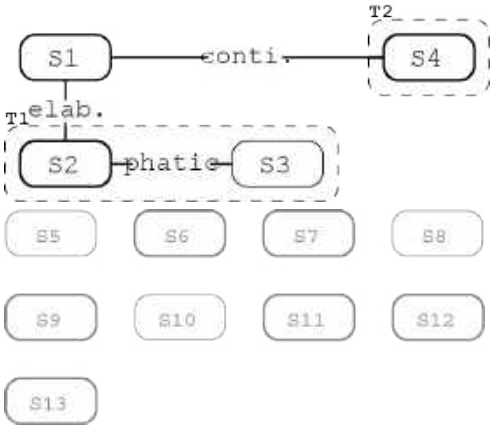
Instanciation de S3

```
<segment id="S3" speaker="speaker2">
    <text>Yes</text>
    <instances><instance type="1" source="S2" name="phatic" topic="T1"/></instances>
</segment>
```



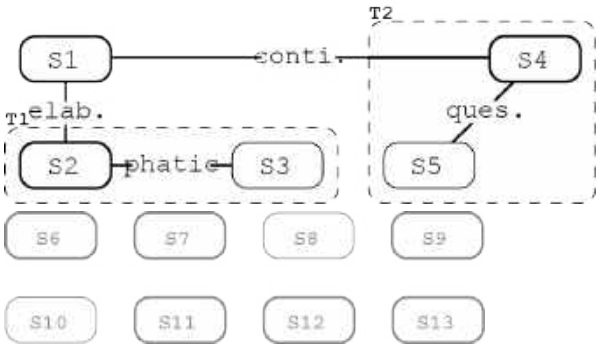
Instanciation de S4

```
        <segment id="S4" speaker="speaker1">
            <text>JCD is dead, L is dead, P is dead uh (…)</text>
            <instances><instance type="1" source="S1" name="cont." topic="T2"/></instances>
        </segment>
```



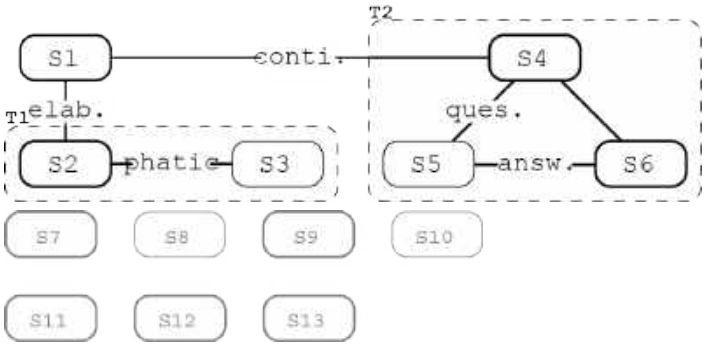
Instanciation de S5

```
        <segment id="S5" speaker="speaker2">
            <text>So you think they're dead because they lost (↑)</text>
            <instances><instance type="3" source="S4" name="quest." topic="T2"/></instances>
        </segment>
```



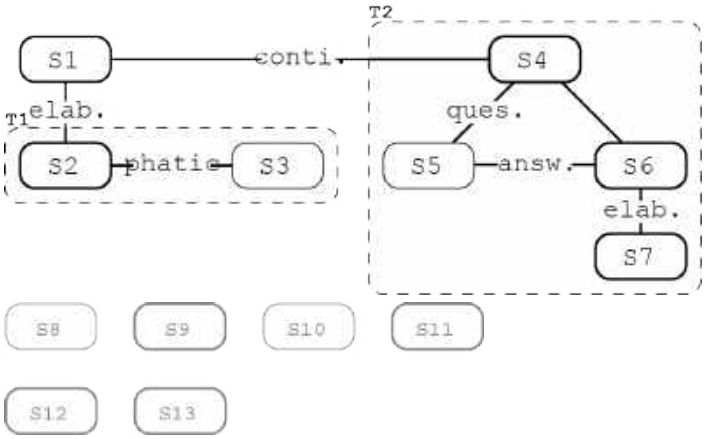
Instanciation de S6

```
        <segment id="S6" speaker="speaker1">
            <text>No they won but if they're dead,</text>
            <instances><instance type="1" source="S5" name="answ." topic="T2"/></instances>
        </segment>
```



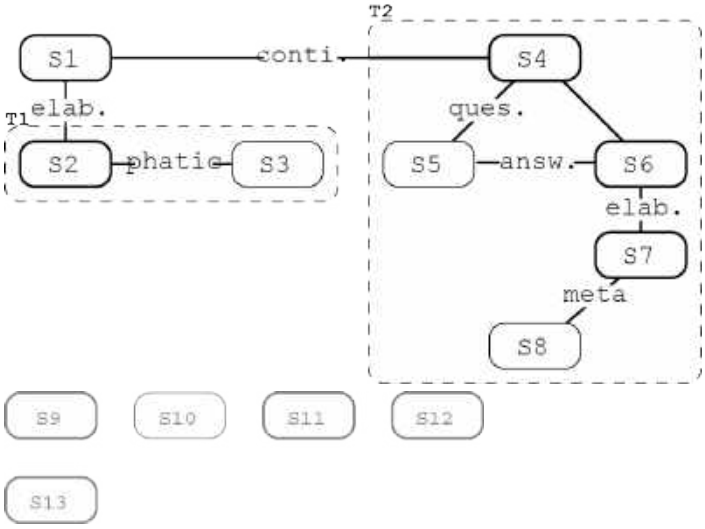
Instanciation de S7

```
        <segment id="S7" speaker="speaker1">
            <text>it's their disease well it's it's (→)</text>
            <instances><instance type="2" source="S6" name="elab." topic="T2"/></instances>
        </segment>
```



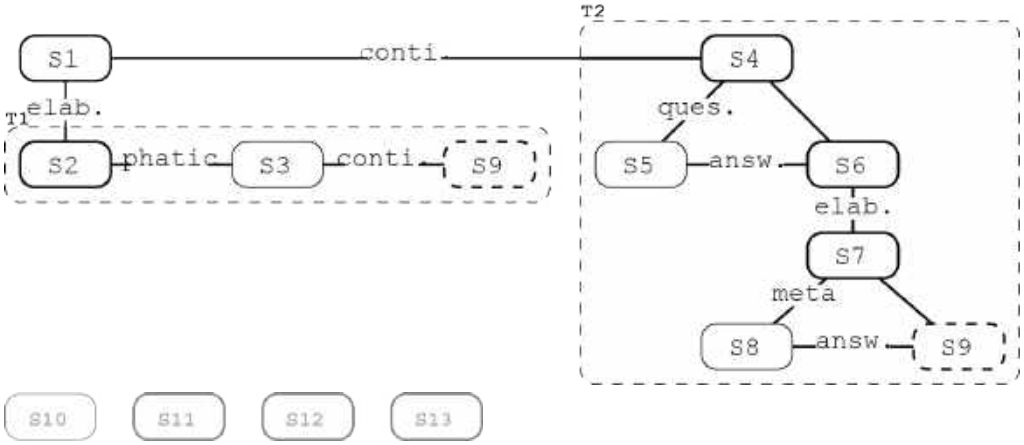
Instanciation de S8

```
        <segment id="S8" speaker="speaker2">
            <text>Yeah it's because they had a disease, it's not because they were in politics
            (↑)</text>
            <instances><instance type="3" source="S7" name="meta" topic="T2"/></instances>
        </segment>
```



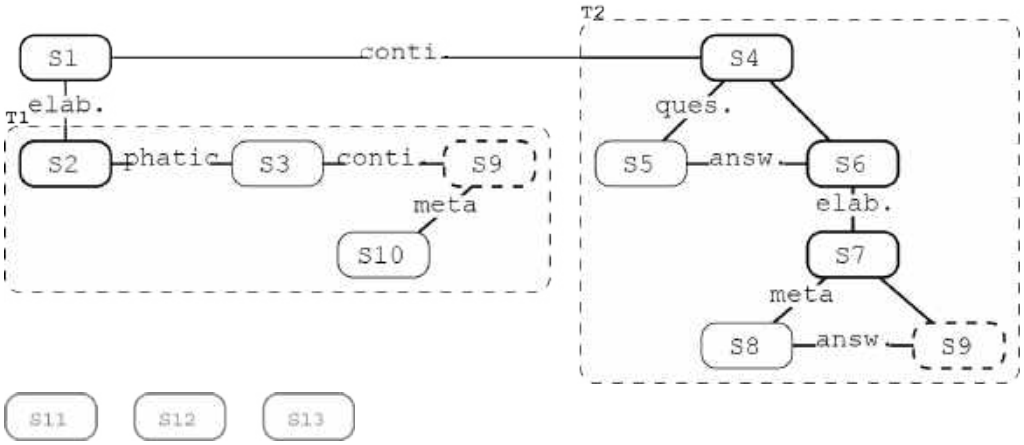
Instanciation de S9

```
        <segment id="S9" speaker="speaker1">
            <text>Yes I mean (→)</text>
            <instances>
                <instance type="2" source="S8" name="answ." topic="T2"/>
                <instance type="2" source="S3" name="cont." topic="T1"/>
            </instances>
        </segment>
```



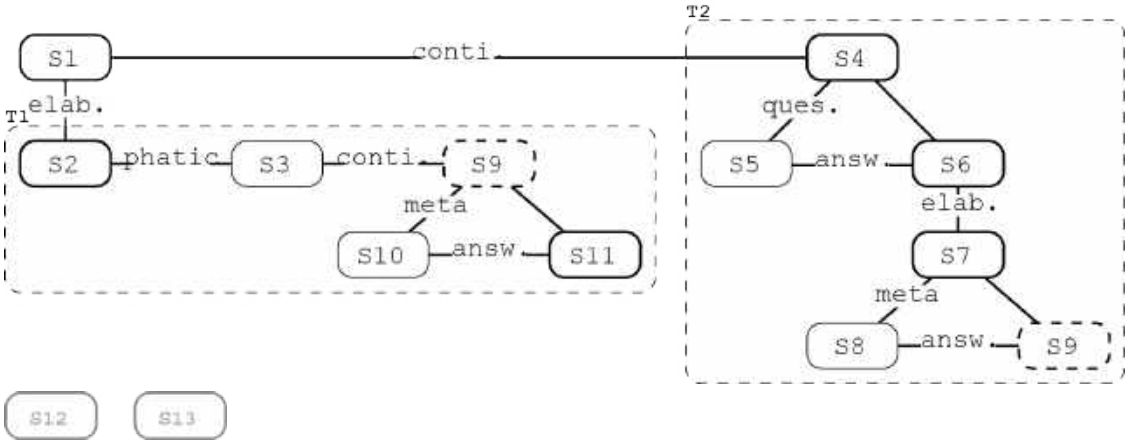
Instanciation de S10

```
        <segment id="S10" speaker="speaker2">
            <text>Yes you think it's because they were in politics (↑)</text>
            <instances>
                <instance type="3" source="S9" instance="2" name="meta" topic="T1"/>
            </instances>
        </segment>
```



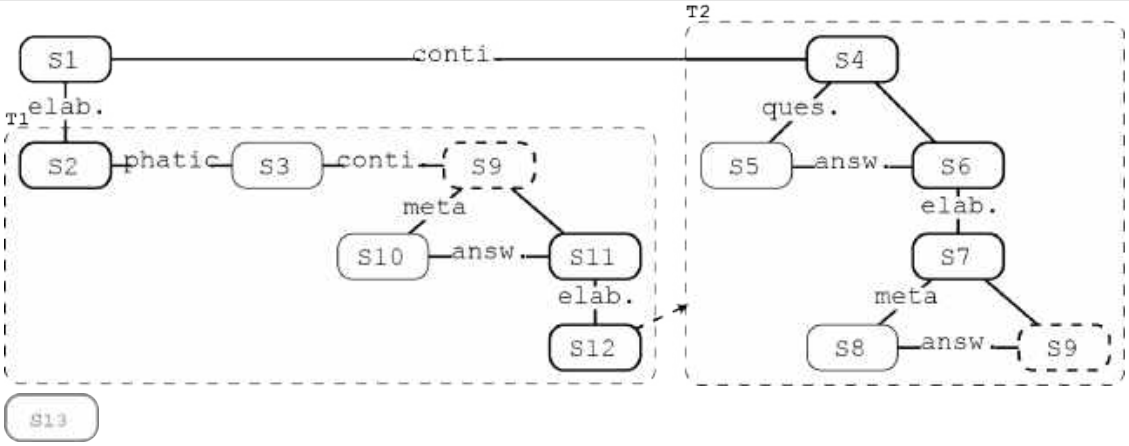
Instanciation de S11

```
        <segment id="S11" speaker="speaker1">
            <text>Yes, so well yeah there was C too who committed murder, uh huh (→)</text>
            <instances>
                <instance type="1" source="S10" name="answ." topic="T1"/>
            </instances>
        </segment>
```



Instanciation de S12

```
        <segment id="S12" speaker="speaker1">
            <text>he was there too, the one in B but well (→)</text>
            <instances>
                <instance type="2" source="S11" name="elab." topic="T1">
                    <topicjump topic="T2"/>
                </instance>
            </instances>
        </segment>
```



Instanciation de S13

```
        <segment id="S13" speaker="speaker1">
            <text>it, that, it's because of politics again</text>
            <instances>
                <instance type="2" source="S12" name="elab." topic="T1"/>
            </instances>
        </segment>
```

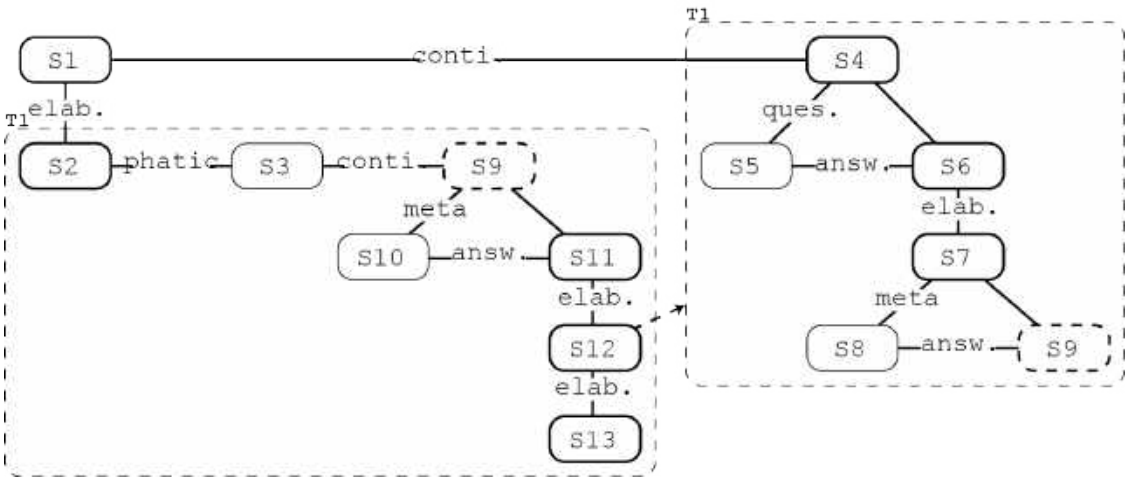


Illustration 16: représentation finale de 'which dead' segmenté et instancié

Clôture de l'arbre XML

```
    </segments>
</scml>
```

Procédure d'installation

La procédure ci-après indique le détail des opérations pour configurer l'environnement d'exécution de l'application Schisme sous les plateformes Microsoft Windows et Apple MacOSX. Les environnement UNIX-like contiennent généralement le nécessaire par défaut; si ce n'est pas le cas, se référer à la documentation de la distribution en question pour installer et configurer un serveur Apache et MySQL.

Configuration requise

Serveur

Schisme est une application de type client-serveur écrite en PHP et utilisant une base de donnée MySQL. Elle doit donc être exécutée par un serveur sur lequel sont installé les environnements PHP et MySQL.

Le serveur doit avoir les caractéristiques suivantes :

- PHP 5.2 et plus
- MySQL 5.1 et plus

Un serveur PHP - MySQL peut être installée aisément en local sur un micro-ordinateur personnel afin de pouvoir utiliser l'application sans connexion internet

Client

Un simple navigateur internet suffit pour pouvoir accéder à l'application. Schisme 1.0b a été conçu pour fonctionner sur toutes les plateformes :

- Un navigateur internet avec Javascript activé
- Plugin Adobe Flash Player (uniquement si le navigateur ne supporte pas le SVG)

Il a été validé, et se comporte identiquement sous les quatre grands navigateurs du marché :

- Chromium : http://www.chromium.org
- Google Chrome : http://www.google.com/chrome
- Safari : http://www.apple.com/safari
- Firefox : http://www.mozilla.com/firefox
- Internet Explorer : http://windows.microsoft.com/fr-FR/internet-explorer/downloads/ie
- Flash Player : http://get.adobe.com/flashplayer

	Microsoft IE 8 et inf.	**Microsoft IE9 et sup.**	**Apple Safari**	**Mozilla Firefox**	**Chromium / Google Chrome**
Sans Adobe Flash Player	-	✓	✓	✓	✓
Nécessite Flash Player	✓	-	-	-	-

Tableau 9: Compatibilité des navigateurs

Installation en local sous Microsoft Windows

Installation et configuration de Wampserver

Sous Windows, le logiciel Wampserveur permet d'installer et configurer en une seule fois l'environnement nécessaire à l'exécution locale d'applications PHP – MySQL.

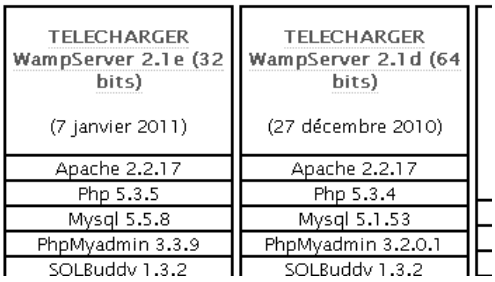
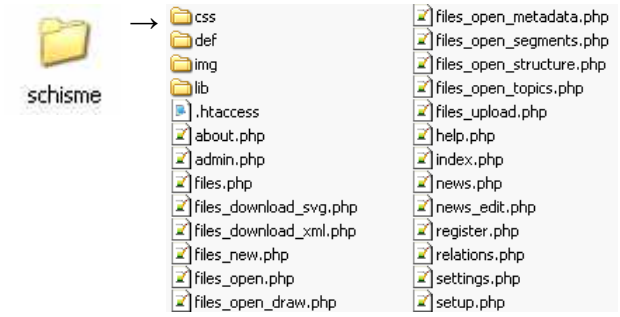
1	Télécharger Wampserver depuis le site officiel.	http://www.wampserver.com 
2	Lancer le programme d'installation en double-cliquant sur l'icône idoine.	
3	Valider les options par défaut et attendre la fin de la copie des fichiers. La fenêtre ci-contre apparaît alors. Cliquer sur '*finish*'. Wampserver se lance automatiquement.	
4	L'icône ci-contre apparaît dans la zone de notifications. Lorsqu'elle devient verte, cela signifie que Wampserver est démarré correctement.	

Tableau 10: Procédure d'installation de Wampserver (PHP - MySQL pour Windows)

A ce stade, Wampserver est installé et prêt à être utilisé. Il faut maintenant installer Schisme dans Wampserver et lancer la configuration automatique.

Installation de Schisme dans Wampserver

Lorsque Wampserver est démarré, l'ordinateur se comporte comme un serveur : il sert aux clients - qui s'y connectent via un navigateur - les fichiers situés dans son dossier de partage. Il peut exécuter des opérations via un langage de script 'côté serveur' tel que PHP, et générer dynamiquement des contenus.

1	Si Wampserver n'est pas démarré, le lancer depuis le menu 'Démarrer' de Windows en utilisant le chemin ci-contre.	`Menu 'Démarrer' → Wampserver → Start Wampserver` 
2	L'icône ci-contre apparaît dans la zone de notifications. Quand est verte, Wampserver est démarré correctement.	
3	Extraire le contenu de l'archive de l'application Schisme.	
4	Veiller à ce que le contenu du dossier ressemble bien à ceci. Vérifier notamment la présence du fichier `setup.php`	

5	Ouvrir le fichier ci-contre avec un éditeur de texte. Vérifier que le premier caractère de la première ligne est bien un dièse.	`.\schisme\.htaccess`
		`#php 1`
		`SetEnv PHP_VER 5`
		`SetEnv REGISTER_GLOBALS 0`
		`SetEnv SESSION_USE_TRANS_SID 0`
		`AddDefaultCharset utf-8`
		`Options -Indexes`
		`IndexIgnore *`

6	Ouvrir le fichier ci-contre avec un éditeur de texte. Vérifier que le contenu est bien le suivant. Il s'agit des informations de connexion à la base de donnée MySQL. Les valeurs par défaut de Wampserveur sont:
	`Login: root`
	`Password:`
		`.\schisme\lib\db_init.php`
		`$db_host="localhost";`
		`$db_login="root";`
		`$db_password="";`
		`$db_name = $db_login;`

7	Placer le dossier 'schisme' dans le dossier de partage 'www' de Wampserver.	 → `C:\wamp\www\`

Tableau 11: Procédure d'installation de Schisme dans Wampserver (Windows)

Schisme est maintenant prêt à être exécuté pour la première fois. Un script d'installation sera exécuté et le logiciel sera prêt à être utilisé.

Installation en local sous MacOS X

Installation et configuration de MAMP

Sous MacOS X, le logiciel MAMP permet d'installer et configurer en une seule fois l'environnement nécessaire à l'exécution locale d'applications PHP – MySQL.

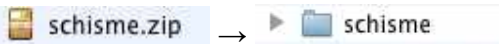
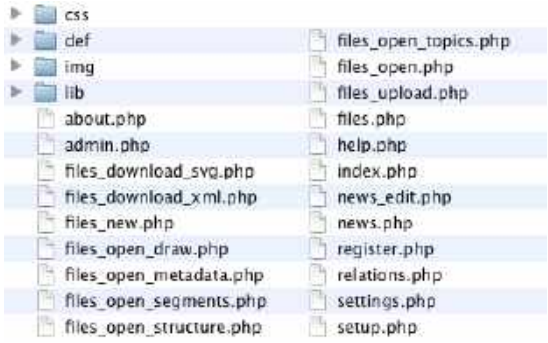
1	Télécharger MAMP depuis le site officiel.	http://www.mamp.info/en/index.html 
2	Lancer le programme d'installation en double-cliquant sur l'icône idoine.	
3	Valider	
4	Faire glisser le dossier MAMP vers le dossier Applications	
5	Ouvrez le dossier Applications → MAMP et ouvrez le programme en cliquant sur son icône	
7	Cliquer sur 'démarrer les serveurs'. Lorsque les deux icônes passent au vert, MAMP est démarré correctement.	

Tableau 12: Procédure d'installation de MAMP (PHP - MySQL pour MacOS X)

A ce stade, MAMP est installé et prêt à être utilisé. Il faut maintenant installer Schisme dans MAMP et lancer la configuration automatique.

Installation de Schisme dans MAMP

Lorsque MAMP est démarré, l'ordinateur se comporte comme un serveur : il sert aux clients - qui s'y connectent via un navigateur - les fichiers situés dans son dossier de partage. Il peut exécuter des opérations via un langage de script 'côté serveur' tel que PHP, et générer dynamiquement des contenus.

1	Si MAMP n'est pas démarré, ouvrez le via 'Applications → MAMP'	
2	Cliquer sur 'démarrer les serveurs'. Lorsque les deux icônes passent au vert, MAMP est démarré correctement.	
3	Extraire le contenu de l'archive de l'application Schisme.	
4	Veiller à ce que le contenu du dossier ressemble bien à ceci. Vérifier notamment la présence du fichier `setup.php`	

5	Ouvrir le fichier ci-contre avec un éditeur de texte. Vérifier que le contenu est bien le suivant. Il s'agit des informations de connexion à la base de donnée MySQL. Les valeurs par défaut de MAMP sont:
	`Login: root`
	`Password: root`
		`.\schisme\lib\db_init.php`
		
		`$db_host="localhost";`
		`$db_login="root";`
		`$db_password="root";`
		
		`$db_name = $db_login;`

6	Placer le dossier 'schisme' dans le dossier de partage 'htdocs' de MAMP.	`Applications → MAMP → htdocs`

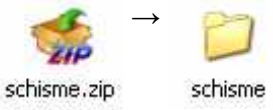
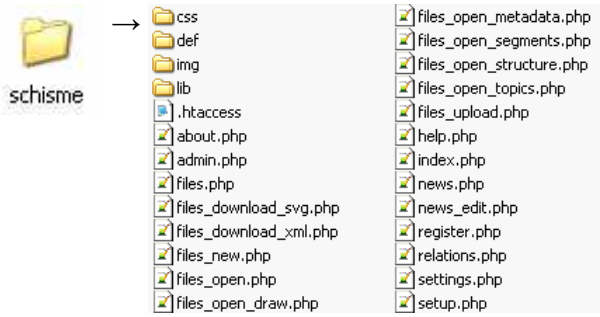
Tableau 13: Procédure d'installation de Schisme dans MAMP (MacOS X)

Schisme est maintenant prêt à être exécuté pour la première fois. Un script d'installation sera exécuté et le logiciel sera prêt à être utilisé.

Installation en distant sur un serveur internet

Le serveur doit disposer de l'environnement nécessaire pour exécuter du code PHP et héberger une base de donnée MySQL.

Schisme n'utilise que des fonctions compatibles avec un serveur Apache '*safe mode*', de ce fait, il est possible de l'installer sur des serveurs mutualisés gratuits du type Free (http://www.free.fr).

1	Extraire le contenu de l'archive de l'application Schisme.	
2	Veiller à ce que le contenu du dossier ressemble bien à ceci. Vérifier notamment la présence du fichier `setup.php`	

5	Ouvrir le fichier ci-contre avec un éditeur de texte.
	
	Sur un serveur normal, vérifier que le premier caractère de la première ligne est bien un dièse.
	
	Pour un serveur free.fr (http://www.free.fr) ou certains autres serveurs mutualisés en 'safe mode' Apache, enlever le dièse pour activer le PHP.
	
	Ceci dépend de la configuration du serveur.
		`.\schisme\.htaccess`
		
		Sur un serveur 'normal':
		`#php 1`
		`SetEnv PHP_VER 5`
		`SetEnv REGISTER_GLOBALS 0`
		`SetEnv SESSION_USE_TRANS_SID 0`
		
		`AddDefaultCharset utf-8`
		
		`Options -Indexes`
		`IndexIgnore *`
		
		Sur un serveur http://www.free.fr:
		`php 1`
		`SetEnv PHP_VER 5`
		`SetEnv REGISTER_GLOBALS 0`
		`SetEnv SESSION_USE_TRANS_SID 0`
		
		`AddDefaultCharset utf-8`
		
		`Options -Indexes`
		`IndexIgnore *`

6	Ouvrir le fichier ci-contre avec un éditeur de texte. Il s'agit des informations de connexion à la base de donnée MySQL. Entrez l'adresse de votre base de donnée (laisser 'localhost' pour un serveur mutualsé de type free.fr), votre login et votre mot de passe.	`.\schisme\lib\db_init.php`
		
		`<?php`
		`/*****`
		`* Cédric Beuzit-Laboz / Joffrey Mougel`
		`*****`
		`$db_host="localhost";`
		`$db_login="login";`
		`$db_password="password";`
		
		`$db_name = $db_login;`
		`?>`

7	Transférer le dossier 'schisme' sur votre serveur par FTP.	

Tableau 14: Procédure d'installation de Schisme pour un serveur distant

Premier démarrage et initialisation de l'application Schisme

Après avoir installé l'environnement d'exécution de Schisme, il est temps de l'exécuter une première fois afin que le script de configuration (`setup.php`) initialise automatiquement la base de donnée et les dossiers utilisateurs.

Exécution du script automatique d'initialisation / configuration

1	Veiller à ce que le serveur PHP – MySQL soit bien démarré, ouvrir un navigateur, et se connecter à l'une des adresses ci-contre (selon la plateforme utilisée).	En local avec Wampserver (Windows):
		`http://localhost/schisme`
		
		En local avec MAMP (MacOSX):
		`http://localhost:8888/schisme`
		
		Sur un serveur distant:
		`http://www.monserveur.com/schisme`

2	Au premier démarrage, Schisme propose d'exécuter la procédure d'initialisation (`setup.php`). Cliquer sur le lien 'Click here to run setup now'.	
3	Si tout s'est bien déroulé, voici la fenêtre qui s'affiche. En cas de problème, il est probable que l'environnement soit mal configuré. Dans ce cas, recommencer depuis le début ou contacter l'adinistrateur.	

Tableau 15: Premier démarrage de Schisme, exécution du script de configuration automatique

Schisme est maintenant prêt à être utilisé. Les tables de la base de données ont été initialisées, et les comptes utilisateurs suivants ont été créés lors de cette procédure:

Login	Password	Description / privilèges
`publicinactive`		Utilisateur inactif (pour tests). Compte publique (pas de mot de passe). Un administrateur doit activer ce compte pour qu'il soit utilisable.
`publiceditor`		Éditeur simple. Publique. Gère ses propres fichiers.
`publicpublisher`		Auteur. Publique. Droits d'éditeur + peut éditer / publier des news.
`publicadvanced`		Editeur avancé. Publique. Droits d'auteur + peut éditer les relations globales.
`admin`	`admin`	Administrateur. Privé. Tous les droits précédent + gestion des utilisateurs.

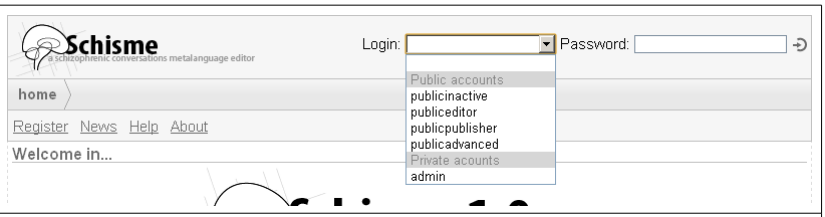
Tableau 16: Description des utilisateurs créés par défaut par le script de configuration

Suppression du fichier de configuration automatique 'setup.php'

Ainsi qu'indiqué à la fin de la procédure de configuration automatique ci dessus, il est recommandé d'effacer le fichier d'initialisation 'setup.php' du dossier de l'application. Pour ce faire, il faut se connecter en tant qu'administrateur.

Si le fichier '`setup.php`' n'est pas effacé, il demeure accessible à l'adresse suivante : `./schisme/setup.php`. Ce script **efface toutes les données** de la base de donnée et des dossiers utilisateur **sans vérification** de droits d'accès **ni confirmation**. C'est pourquoi il est **très dangereux** de le laisser dans le dossier de l'application.

Voici les étapes pour effacer le fichier de configuration automatique:

1	Se connecter en tant qu'administrateur.
	`Login: admin`
	`Password: admin`
		

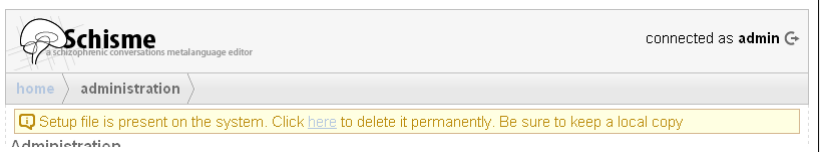
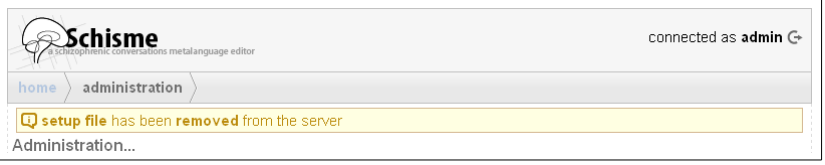
2	Cliquer sur le lien 'Administration' de la barre d'outil pour accéder à l'interface d'administration.	
3	Si le fichier '`setup.php`' est présent sur le système, un message propose de le supprimer. En cliquant sur le lien, le fichier sera supprimé sans demander confirmation.	
4	Un message vous confirme ensuite la suppression du fichier '`setup.php`'.	

Tableau 17: Suppression du fichier de configuration automatique via l'interface d'administration

Schisme est dorénavant prêt à l'emploi.

Accéder à Schisme

- En local avec Wampserver (Windows): `http://localhost/schisme`
- En local avec MAMP (MacOSX): `http://localhost:8888/schisme`
- Sur un serveur distant: `http://www.monserveur.com/schisme`