Exigences fonctionnelles

- **Partage partiel récursif (TED)**
  - [M] Transclusion (Ctrl+C/Ctrl+V)
  - [M] Texte riche
  - [M]
  - [O] Complétion en ligne
  - [O] Unification (Ctrl+C/Ctrl+U)
- **Interface programmable (SDK)**
  - [M] Clauses de parcours (Status – XSL)
    → [O] Datamining dans les agrégats
  - [M] Visualisation-Inspection du Tetris
    → [M] « *Squelletisation* »
    → alpha (chaîne de markov)
    → calque (contextualisation et structuration)
    → API « *Squelletisation* » → Smart-contract
  - [N] Traitement des langues par unification des structures de partage partiel
- **Interface calculante (MIRZA)**
  - [O] Valuation de parcours (eval)
  - [O] Traitement des unités
  - [N] Compilation de partage partiel

[M]andatory
[O]ptional
[N]ice-to-have

Contraintes de conception

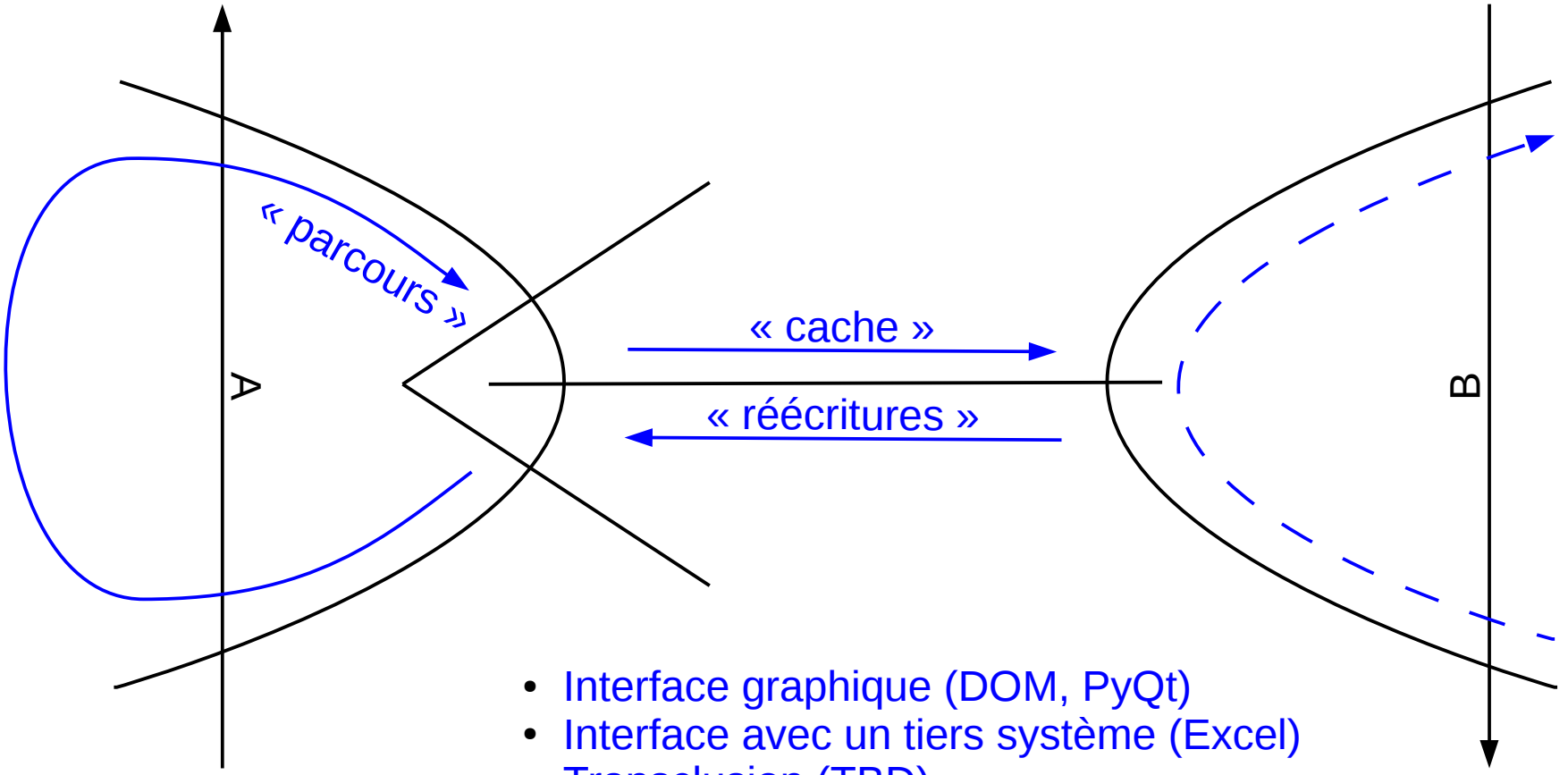
- **Système Décentralisé Interactif (SDI)**
  - [M] Chaque instance est autonome → Serveurs
    → [M] Chaque instance dispose de sa partition dans le flux de données temporel crypté
  - [M] Les termes sont les arcs fermés bien parenthésées liés par des arcs temporels au dual bien parenthésé
    → [M] Les termes s'accumulent dans le Tetris
  - [M] Les perspectives sont les ensembles d'arcs temporels associé à une position
  - [O] Les signes sont les caractères tapées au clavier
    →[N] biométrie comportementale
  - [O] Partition Termes / Signes / Perspectives (TSP) dans le flux de temporel crypté
  - [O] Présentation du buffer avec complétion en combo
- **Interface dynamique à différentiel de termes (CARGO)**
  → [M] Générateur de programme de mise à jour décentralisé
  → [M] réutilisation du moteur xDiff aux vues internes (EMBED)
  → [M] Texte riche par partage partiel de « *document* » et de « *paragraphe* »

S.A.R.L. Mezzònomy
34, place de Catalogne
31700 BLAGNAC
Logiciels programmables pour l'ingénieur
☎ 06 32 97 90 27
APE 6201 Z - TVA FR 35 504 641 473

Difficulté de l'expression fonctionnelle

- L'enrichissement fonctionnel « *métier* » de la plate-forme est la fonction des facilitateurs :
  - Partenaires désirant ouvrir à la plate-forme un domaine d'exploitation et finançant des « *interfaces* » programmables
  - Société de services (ou « services de sociétés ») spécialisées dans l'adaptation à un public donné par la programmation de ces « interfaces »
- Etablir la cahier des charges fonctionnel d'une ou plusieurs extensions
  - A la fois suffisamment générique pour pouvoir être adaptées pour faire des POC
  - Mais pas du niveau de « pureté » de l'industriel = avec encore des « *rustines* » par çi par là qui nuisent à la scalabilité
- Au moins trois :
  - Interface tierce graphique (HTML5 en mode SAAS ou PyQt au mode stand-alone)
  - Interface tierce banque (Excel)
  - Interface paire pour les parcours (statut = règles de parcours)
  - Interface paire pour les squelettes (→ gestion de la configuration)
  -

Principe général d'interface
(ou d' « *extension* »)

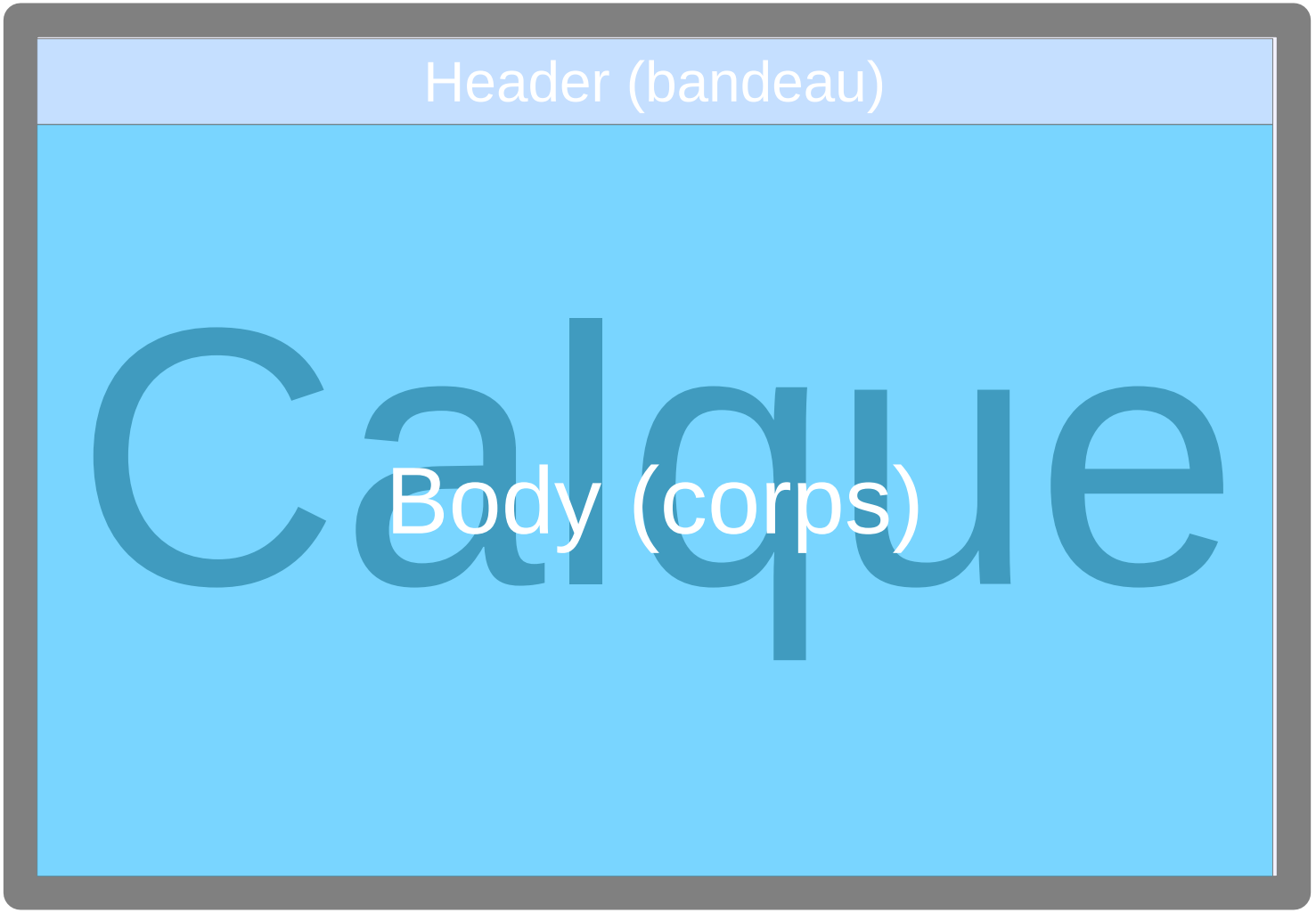


- Interface graphique (DOM, PyQt)
- Interface avec un tiers système (Excel)
- Transclusion (TBD)

Interface DOM

- L'interface DOM est une interface développée sur base CARGOWEB.
  - Chaque élément peut contenir des actions (= « *contrats* »)
    → les actions transforment les événements en « *termes* » (= « *actes* »)
  - L'interface est mise à jour par un programme différentiel
    → Algorithme CARGO (→ add/remove, set)
    → via une connexion WebSocket
  - Pas de limitation à l'HTML5,
    → conflits possible dans l'utilisation de Javascript, à vérifier avec soin
- Adjonction du calque de « *socialisation modale* » :
  - HTML 5 – Canvas avec des animations
    → identité graphique des applications de formulation modale depuis 2011
    → chemin vers l'édition sûre (Canvas n'a pas de donnée conservée sur le client)
- Bandeau du réseau social (identité, rôle, société, notifications)
  - Intégre le « *buffer* » TED hors des composants édités eux-mêmes
    → le clavier tactile s'intègre tout naturellement
    → l'input procède de l'identité (et inversement → biométrie comportementale)
    → embarque les actions TED dans les actions CARGO

Interface DOM : top-level



Interface asynchrone

- L'interface asynchrone est utilisée principalement par les développeurs :
  - Pour initier le développement des applications
  - Pour interfacer les systèmes tiers
- L'interface asynchrone permet de :
  - Récupérer un arbre XML associé à une vue
  - Pousser un arbre XML de remplacement
  - Seul le différentiel est stocké comme contribution
- L'interface asynchrone nécessite un diff plus sophistiqué que l'interface DOM
  - Le CARGO fonctionne tant que le nombre d'ayants-droit sur un nœud est proche de 1.
  - Pour être de qualité nettement supérieure, le diff doit suivre les évolutions du texte → API

Plan de réalisation

- Implémenter le TETRIS:
  - « *Blockchain de termes spatio-temporels* »
  - Réaliser un moteur d'empilage graphique (Cf. my-first-tetris-block.pdf)
  - Réaliser un diff asynchrone pour initier l'écriture des XSL et tester la base du système avec des cas de tests importants.
- Brancher CARGOWEB sur le TETRIS
  - quitte à reprendre le moteur, partir en mode SAAS dès le départ cette fois-ci.
  - Développer le composant d'édition de texte avec séparation du buffer (plus simple et plus proche de la réalité du modèle)
- Obtenir la non-regression des POC 2015
  (fonctionnelle ou pas, juste pour expressivité des statuts)
  - QAO : Qualité Assisté par Ordinateur :
    - éditeur de processus orienté objet
      → intégrer l'édition de description
  - QDF : Qualité Données Fournisseurs
    - filtre de données critiques sur des substances
      → intégrer l'édition des gabarits
  - TED : Tetris Editor
      → programmer les volets de navigation
  -