Implémentation d'un démonstrateur

Dans le chapitre précédent, nous avons réalisé une analyse fonctionnelle de CACDA et proposé un modèle de données adapté à son fonctionnement. Ce chapitre détaille le développement d'un démonstrateur, nécessaire à la validation empirique de notre approche (Figure 34). Ce démonstrateur est destiné à être testé dans le cadre de cas d'études industriels.

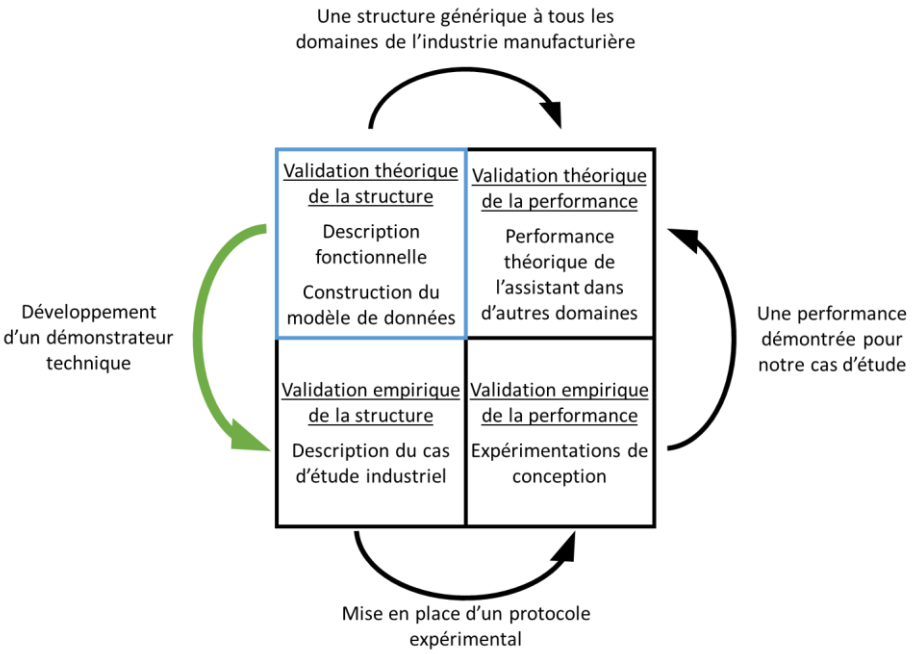


Figure 34 : Développement d'un démonstrateur technique

3.1 Structure logicielle du démonstrateur

Cette partie présente le démonstrateur de CACDA, notamment les choix de développement et l'architecture logicielle retenue. Il est à noter que ces choix sont avant tout techniques, d'autres outils ou langages de programmation pourraient être utilisés à des fins similaires.

Le démonstrateur est développé en Python[15], langage bien connu et disposant d'une large communauté et de nombreuses bibliothèques. L'utilisation de ces bibliothèques spécialisées facilite le développement, en particulier pour la communication du démonstrateur avec plusieurs ressources externes, notamment des outils de traitement du langage naturel et une base de données orientée graphe. Python sert donc de langage pivot entre les différentes couches spécialisées du logiciel (Figure 35).

[15] https://www.python.org/

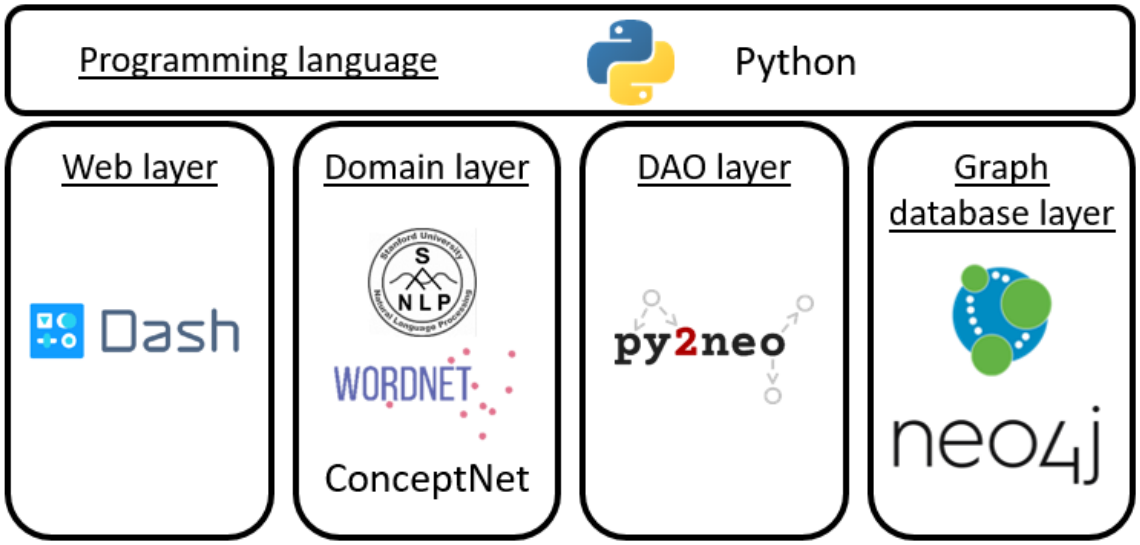


Figure 35 : Structure logicielle du démonstrateur

La première couche spécialisée est l'interface web de l'application, permettant l'interaction avec le concepteur. Elle est réalisée avec une bibliothèque Python nommée Dash[16] qui fournit une infrastructure pour créer des interfaces web. L'interface du démonstrateur est présentée dans la suite de ce chapitre (Figure 43).

La deuxième couche contient les outils spécialisés nécessaires au traitement du langage naturel. Il s'agit de Stanford CoreNLP [17] pour les algorithmes d'analyse, de l'ontologie ConceptNet[18] et du thésaurus WordNet[19] pour l'enrichissement sémantique. Ces outils sont utilisés lors de l'écriture du sous-contexte sémantique et de l'enrichissement sémantique du graphe. Ce processus est détaillé dans la partie 3.2.1.

La troisième couche est appelée DAO pour "*Direct Access Object*" et permet l'interaction de l'assistant avec la base de données. Nous utilisons py2neo, une API Python, pour manipuler directement les nœuds et relations de la base de données.

La dernière couche logicielle est celle de notre base de données. Nous utilisons la base de données orientée graphe NEO4J [97]. Comme mentionné dans l'état de l'art, NEO4J permet la modélisation d'un graphe de connaissances essentiel au raisonnement sur les données. Ce choix est tout d'abord justifié par la souplesse de NEO4J sur la structure des données, nous permettant d'améliorer notre modèle de données au fur et à mesure de l'avancée du développement du démonstrateur. L'outil permet également une interface simple avec des programmes externes, ce qui facilite le développement.

[16] https://dash.plotly.com/introduction
[17] https://stanfordnlp.github.io/CoreNLP/
[18] https://wordnet.princeton.edu/
[19] https://conceptnet.io/

3.2 Fonctionnalités du démonstrateur

Lors de l'analyse fonctionnelle de CACDA, nous avons identifié 3 services permettant d'améliorer la maîtrise des règles de conception. Notre démonstrateur apporte une réponse technique à chacun de ces services. Notre objectif est de réaliser un démonstrateur de CACDA afin que sa pertinence, son principe de fonctionnement et son utilisabilité puissent être testés.

3.2.1 Ecriture des sous-contextes sémantique et technique

Le premier service de l'assistant est de structurer les règles de conception et le contexte de conception dans un graphe de connaissances. Comme présenté dans l'état de l'art, plusieurs approches existent pour extraire des règles de conception de documents en langage naturel [43]. Cependant, cette technologie est encore complexe à mettre en part et ne constitue pas le cœur de notre apport. En conséquence, nous considérerons pour notre démonstrateur, que l'ensembles des règles utilisées sont stockées dans des documents semi-structurés, listant les documents sources de ces règles, les chapitres auxquels elles appartiennent ainsi que leurs énoncés principaux (Tableau 11).

Numéro de règle	Enoncé principal	Document source	Numéro de chapitre
1	*All milling element should be modeled with respect to cutter diameter. When possible, always select a standard dimension for milling cutters to minimize cost.*	Design Rules for Metallic Parts	4.2.2
2	*The maximum cutting height of a milling cutter determines the maximum depth of a pocket that can be machined. This value depends on the cutter diameter*	Design Rules for Metallic Parts	4.2.2
3	*It is necessary to have between wall corners a radius higher than the milling cutter radius*	Design Rules for Metallic Parts	4.2.3

Tableau 11 : Extrait du document source semi-structuré

Avec ces données d'entrée, le processus de création du graphe se déroule en 4 étapes (Figure 36) :

Etape 1 : L'ensemble des règles de conception sont extraites des documents semi-structurés. Le démonstrateur peut ainsi créer les nœuds qui correspondent aux règles, documents et chapitres. Les énoncés des règles sont récupérés afin d'en réaliser l'analyse.

Etape 2 : Les mots-clés issus de l'énoncé principal des règles de conception sont extraits et rajoutés au graphe. Ils passent ensuite par un processus de désambiguïsation et d'enrichissement sémantique. Ces procédés permettent d'ajouter des éléments sémantiques proches du mot d'origine (synonymes, concepts, etc.) afin de faciliter la recherche des règles. Les détails techniques de ces procédés ainsi qu'un exemple représentatif sont donnés ci-

dessous. Les nœuds contenant du texte (documents, chapitres, définitions etc.), à l'exception des règles, sont rattachés aux mots-clés contenus dans leurs énoncés.

Etape 3 : Une liste de mots-clés, représentatifs de notre domaine d'étude, est extraite du graphe de connaissances pour servir de référence au processus de désambiguïsation. Cette étape, où l'assistant récupère des données dans le graphe est simultanée avec l'étape 2.

Etape 4 : Les éléments techniques sont ensuite identifiés dans le graphe à partir d'un dictionnaire de termes techniques. L'assistant leur associe alors les labels appropriés et les relie entre eux pour former le sous-contexte technique.

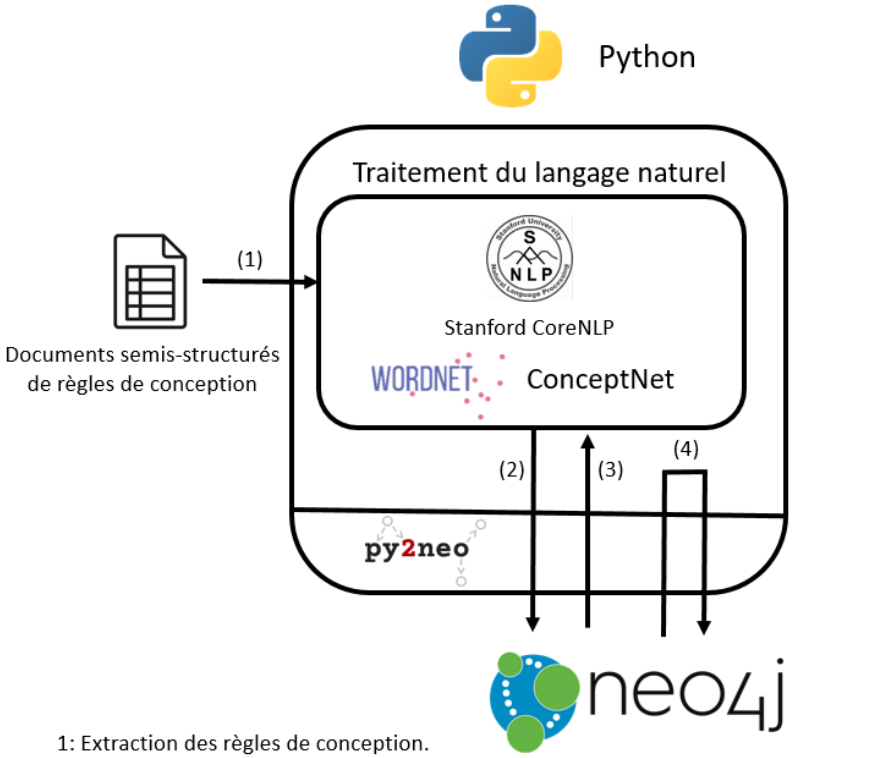


Figure 36 : Architecture logicielle pour l'écriture du graphe de connaissances

Afin de détailler les procédés de désambiguïsation et d'enrichissement sémantique évoqués plus tôt, considérons la règle suivante :

It is necessary to have between wall corners a radius a bit higher than the milling cutter radius to avoid an engagement of the tool in the part corner

Lors de la première étape, le démonstrateur lit la règle dans le document semi-structuré. Les outils de traitement du langage naturel de Stanford CoreNLP permettent au démonstrateur d'isoler les mots-clés (i.e. noms, verbes, adjectifs et adverbes) de la règle et de les réduire à leur base lexicale, appelée lemme. Ces lemmes sont ajoutés au graphe et reliés à la règle (Figure 37).

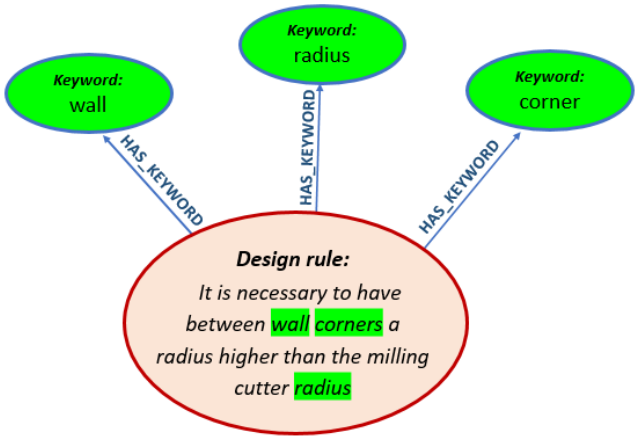


Figure 37: Nœud représentant une règle de conception reliée à trois de ses mots-clés

Considérons le mot-clé *radius*. En consultant le thésaurus WordNet, le démonstrateur extrait l'ensemble des significations possibles de ce mot (appelés synsets dans WordNet) et en déduit une liste de mots représentatifs issus des éléments sémantiques proches dans WordNet (**Tableau 12**).

Définition	Liste de mots représentatifs
The length of a line segment between the center and circumference of a circle or sphere.	[length, line, segment, center, circumference, circle, sphere, r, curvature, semidiameter, diameter]
A circular region whose area is indicated by the length of its radius.	[circular, region, area, indicate, length, part, location]
The outer and slightly shorter of the two bones of the human forearm.	[outer, short, bone, human, forearm, arm]

Tableau 12 : Exemple de trois significations du mot-clé radius

Le processus de désambiguïsation consiste à choisir le bon sens de *radius*. L'assistant va comparer les listes représentatives des synsets avec une liste de référence de 400 mots issus du graphe de connaissances. Cette liste, extraite dans l'étape 2 de notre procédé, est constituée des 400 mots les plus proches de la règle dans le graphe (distance en nombre de relations). Le démonstrateur mesure alors un score de similarité basé sur le nombre de mots communs entre les deux listes comparées. Cette méthode, appelée similarité cosinus est largement utilisée pour calculer des similarités linguistiques [154], [157], [172]. Dans notre exemple, la définition la plus similaire à notre contexte est la première. Les informations correspondant à cette signification sont écrites dans le graphe (Figure 38), c'est le processus d'enrichissement sémantique.

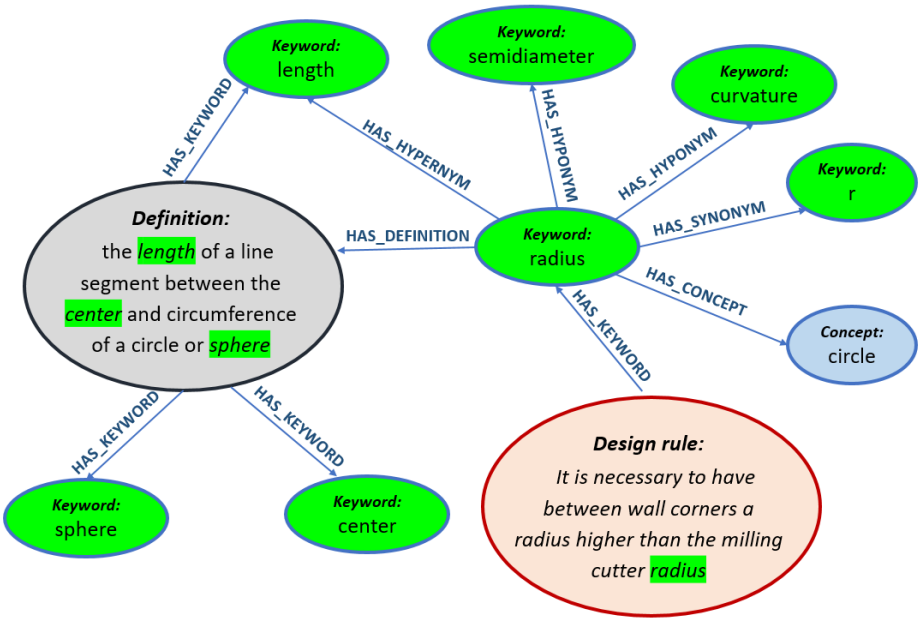


Figure 38 : enrichissement sémantique du mot-clé *"radius"*

L'interface du démonstrateur dispose d'une page dédiée à l'ajout de règles de conception (Figure 39). Elle permet à un utilisateur d'importer le document semi-structuré à analyser, et d'ajouter de nouvelles règles de conception en saisissant directement un énoncé.

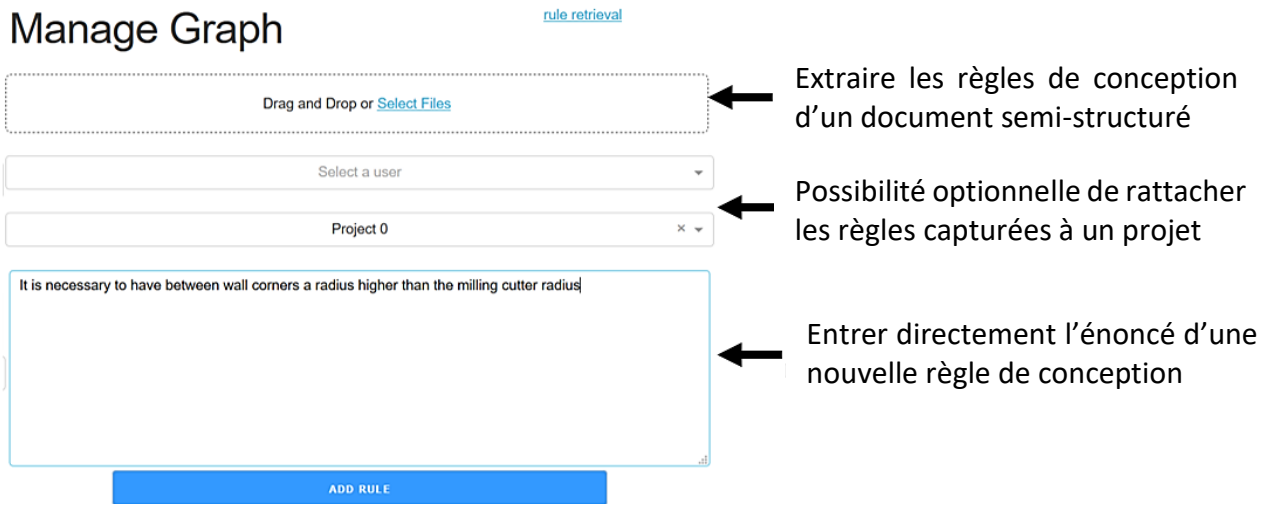


Figure 39 : Interface d'ajout de règles de conception dans le graphe

3.2.2 Recommandation des règles de conception

Le deuxième service rendu par l'assistant est de recommander des règles de conception adaptées au besoin de l'utilisateur. Pour réaliser ce service, notre démonstrateur s'appuie sur un système de recommandation piloté par une interface de recherche de règles de

conception. Cette partie présente en détail l'algorithme de recommandation utilisé puis le processus total de recherche des règles de conception.

3.2.2.1 Algorithme de recommandation

Comme présenté dans l'état de l'art (partie 1.4.3), il existe une grande variété de systèmes de recommandation. L'objectif n'est pas d'identifier le meilleur algorithme mais d'en sélectionner un qui est adapté à la recommandation contextuelle afin d'expérimenter notre approche.

L'algorithme ContextWalk [160] est basé sur un principe de marche aléatoire dans le graphe des connaissance. Son utilisation dans notre cas a plusieurs avantages :

- Comme souligné par son auteur, ce système est facilement adaptable à divers domaines d'application et modèles de données. Il est donc possible de l'adapter à la conception assistée par ordinateur tout en considérant les évolutions du modèle de données.

- L'algorithme est prévu pour de la recommandation contextuelle dans des graphes hétérogènes. Tous les sous-contextes du modèle de données sont explorés simultanément. Cela permet, d'une part, de prendre en compte l'ensemble du contexte de l'utilisateur lors des recommandations et d'autre part, d'identifier les éléments contextuels les plus pertinents afin de les utiliser comme filtres pour les prochaines recommandations.

- Le rajout de nouveaux sous-contextes est simple. Cela nous permet de conserver le même système de recommandation au cours du développement et des expérimentations successives.

- La marche aléatoire peut débuter avec un nombre de nœuds quelconque. Chaque nœud de départ peut avoir un poids différent. Ici, cela permet de sélectionner plusieurs mots-clés issus d'une requête du concepteur, mais aussi de sélectionner des éléments contextuels qui pourront influencer les recommandations.

Il est toutefois nécessaire d'adapter le système de recommandation à notre démonstrateur. Afin de construire une matrice d'adjacence cohérente, il faut lister l'ensemble des nœuds de notre graphe et de les classer par sous-contextes et labels. Cet ordre est utilisé pour construire le vecteur de position ainsi que la matrice probabiliste de transition. Ainsi, chaque nœud est associé à une dimension du vecteur de position et à une ligne de la matrice de transition. La Figure 40 présente l'ordre des sous-contextes et labels dans le vecteur de position.

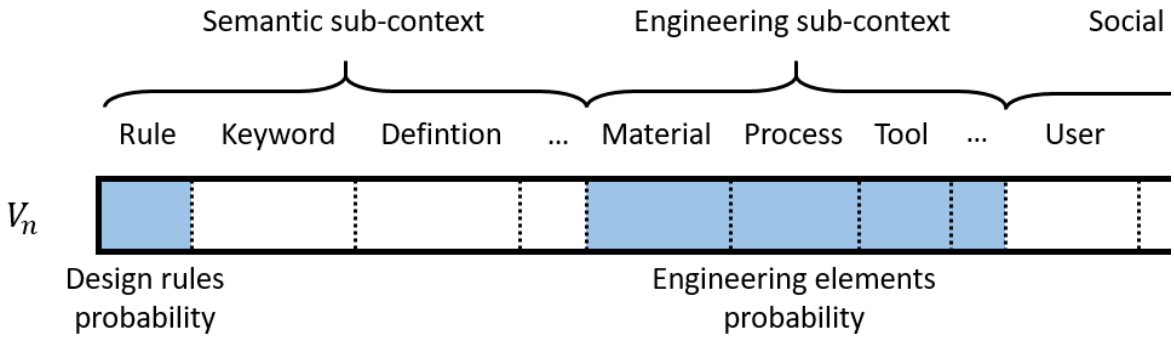


Figure 40 : Ordre des sous-contextes et labels dans le vecteur de position

Lorsqu'une recommandation est effectuée, l'assistant peut lire dans le vecteur résultat les règles dont le score est le plus élevé, mais également les éléments les plus pertinent de tout un sous-contexte, comme indiqué en bleu dans la Figure 40.

Afin de construire la matrice probabiliste de transition, nous construisons d'abord une matrice d'adjacence (A) de notre graphe de connaissances. Pour chaque nœuds i et j, $A(i,j)$ est égal à 1 si une relation relie les nœuds i et j, sinon $A(i,j)$ est égal à 0. Cette matrice est ensuite pondérée afin que la somme de chacune de ses lignes soit égale à 1. Un poids égal est donné à chacun des sous-contextes. En suivant les recommandations de Bogers, nous fixons également une probabilité d'auto-transition α égale à 0,7. Ce facteur a pour effet de ralentir la progression de la marche aléatoire dans le graphe et donc de favoriser les nœuds plus proches des points de départ.

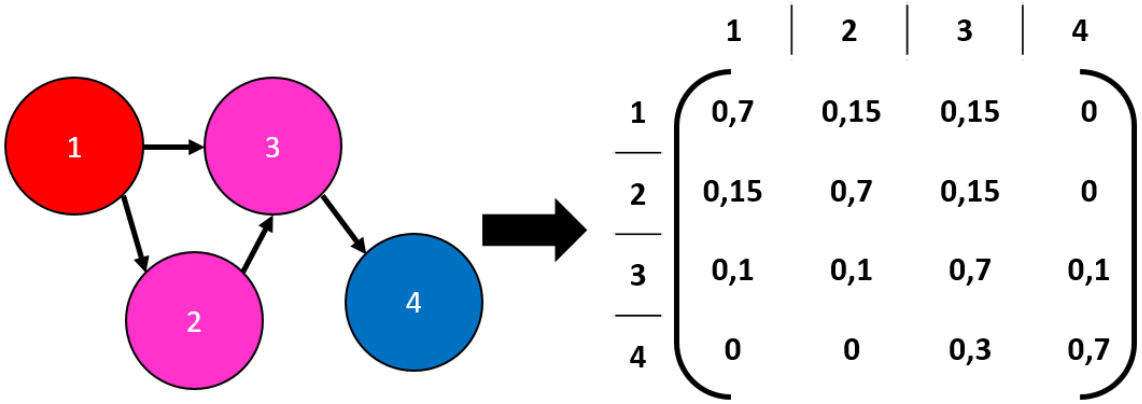


Figure 41 : Exemple de matrice probabiliste de transition

La Figure 41 présente un exemple de matrice probabiliste de transition pour un graphe de quatre nœuds. On remarque que, pour chaque nœud, la probabilité d'auto-transition, située sur la diagonale de la matrice est égale à 0,7. La somme de chaque ligne est bien égale à 1. Le nœud en rouge représente une règle, les nœuds roses des mots clés et le nœud bleu un concept technique. Supposons qu'un utilisateur sélectionne le nœud 4 pour initier une recommandation (Figure 42). Le vecteur de position initial V_0 indique donc une probabilité de position égale à 1 pour ce nœud. On remarque qu'à chaque pas dans le graphe, la somme des probabilités de position est toujours égale à 1. Chaque nœud du graphe est associé à une probabilité de position, ce qui permet à l'assistant de sélectionner les différents nœuds à recommander ou à utiliser en filtres contextuels.

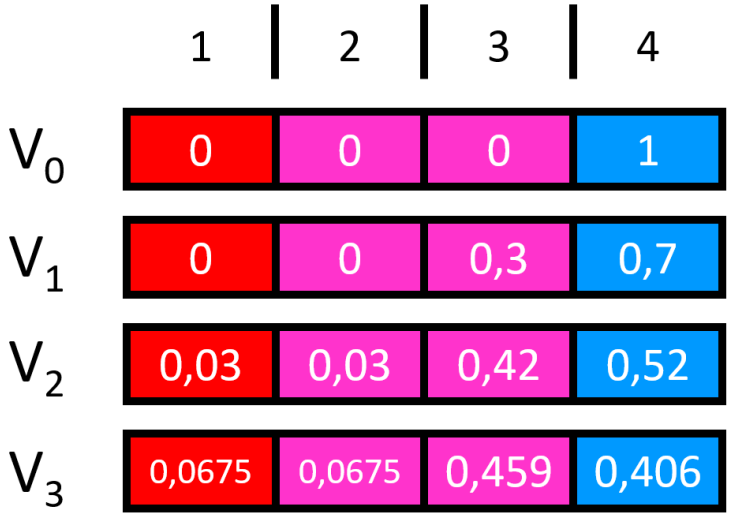
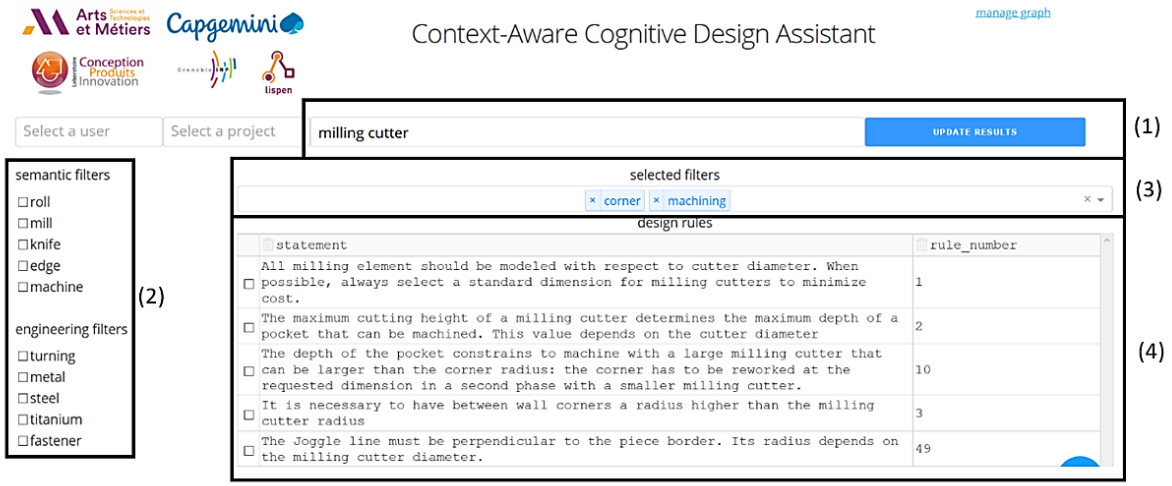


Figure 42 : Exemple trois pas de marche aléatoire dans un graphe de quatre nœuds

3.2.2.2 Processus de recherche de règles

Une fois l'algorithme de recommandation fonctionnel, le démonstrateur peut réaliser le deuxième service de l'assistant, recommander au concepteur des règles adaptées à son besoin.



1: Barre de recherche
2: Filtres contextuels proposés par le démonstrateur
3: Filtres contextuels sélectionnés par le concepteur
4: Liste des règles de conception recommandées

Figure 43 : Interface web pour la recherche de règles de conception

Afin d'interagir avec le concepteur et recueillir son besoin, le démonstrateur dispose d'une interface web (Figure 43). Celle-ci dispose d'une barre de recherche (1) permettant de recueillir les requêtes du concepteur. Les nœuds initiaux de la marche aléatoire sont issus des mots clés de cette requête ainsi que des filtres contextuels sélectionnés par le concepteur (3). Le bouton "Update Results", à droite de la barre de recherche, permet de lancer une recommandation. Les règles recommandées apparaissent dans la partie centrale de l'interface et les filtres contextuels proposés dans la partie gauche sont mis à jour. Pour des raisons expérimentales, lorsque le concepteur sélectionne une règle à appliquer, le démonstrateur ouvre un fichier PDF contenant les informations de cette règle.

Le processus logiciel de recommandation de règles est présenté dans la Figure 44. Ce processus se divise en 6 étapes :

Etape 1 : Le concepteur rentre une requête et peut également sélectionner des filtres contextuels. Quand le concepteur clique sur le bouton "Update Results", le démonstrateur sélectionne les nœuds initiaux pour la marche aléatoire à partir des mots-clés de la requête et des filtres sélectionnés. Cette sélection permet de construire le vecteur initial V_0.

Etape 2 : La marche aléatoire est lancée avec le vecteur V_0. Celui-ci est multiplié par la matrice probabiliste de transition pour réaliser des pas aléatoires dans le graphe.

Etape 3 : Le système effectue 7 pas dans le graphe. Cette distance est un compromis qui dépend de la taille du graphe. Un faible nombre de pas favorise les nœuds proches des nœuds initiaux et ne permet pas une exploration suffisante. Un grand nombre de pas avantage les nœuds les plus densément connectés et donne le même résultat indépendamment des nœuds initiaux. Une série de tests du système de recommandation nous a permis de fixer une

distance adaptée à notre graphe. Une fois la marche aléatoire terminée, le vecteur V_7 permet de lire les résultats.

Etape 4 : Le démonstrateur lit le vecteur V_7. Il sélectionne les règles et éléments contextuels avec le meilleur score. Ce score correspond à la probabilité que la position de la marche aléatoire après 7 pas soit sur ce nœud. Le système affiche alors à l'utilisateur les règles les plus probables ainsi que les filtres les plus pertinents pour chaque sous-contexte.

Etape 5 : Le concepteur peut alors choisir les règles de conception qu'il souhaite consulter dans l'interface web. Les PDF correspondant aux règles choisies sont ouverts par le démonstrateur.

Etape 6 : Le concepteur peut alors lire et analyser les règles ouvertes et se servir de cette information pour son travail de modélisation.

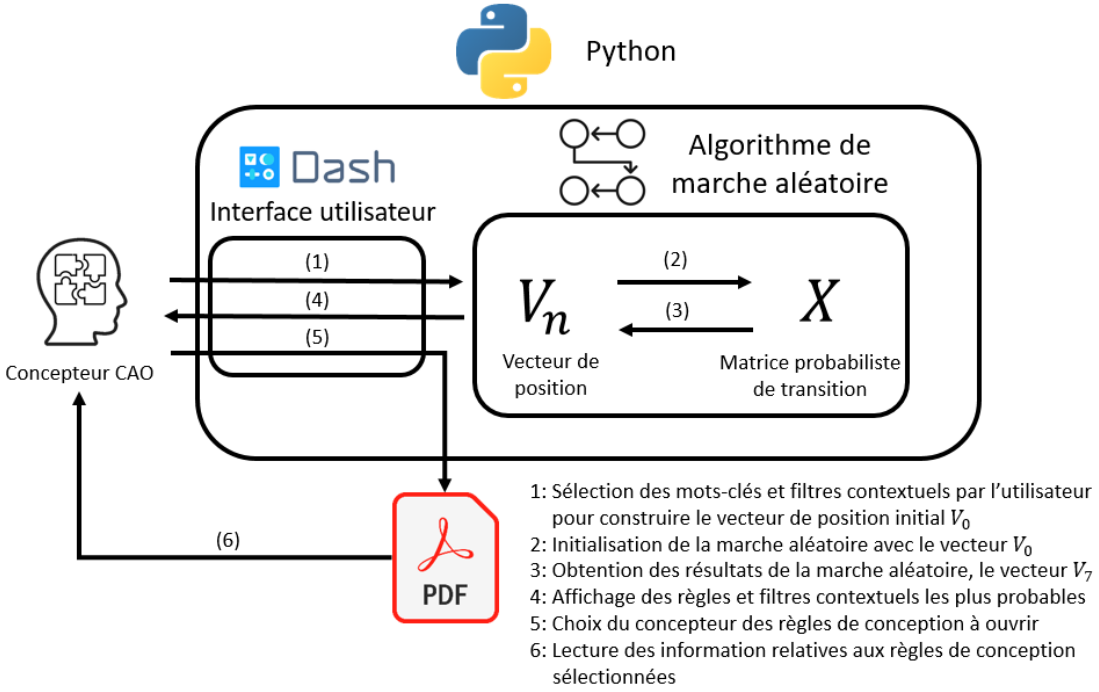


Figure 44 : Structure logicielle de la recommandation de règles de conception

3.2.3 Ecriture des sous-contextes social et numérique

Les sous-contextes social et numérique sont traités à part dans ce chapitre car ils sont les moins directs à mettre en place. En effet, le premier dépend fortement du domaine et le second requière une mise à jour en temps réel et le développement d'un connecteur avec un logiciel de CAO.

3.2.3.1 Sous-contexte social

Le sous-contexte social modélise les informations contextuelles relatives à l'environnement social du concepteur : ses collègues de bureau, son équipe, les experts de son entreprise, etc. Modéliser un tel sous-contexte de façon arbitraire à la main entrainerait de nombreux biais.

Notre approche pour représenter ce sous-contexte est donc d'utiliser les interactions des premiers participants à nos expérimentations avec les règles de conception. Ces participants ayant tous interagis avec un même set de règles pour réaliser la même tâche peuvent représenter les membres d'une même équipe de conception. Le sous-contexte social résultant alors d'interactions réelles avec le démonstrateur, n'est plus soumis au biais de l'arbitraire. Un participant pourra donc voir ses recommandations influencées par les participants précédents via le sous-contexte social.

3.2.3.2 Sous-contexte numérique

La modélisation du sous-contexte numérique nécessite un échange d'information entre notre démonstrateur et un logiciel de CAO. La majorité des logiciels de CAO disposent d'une API[20], laquelle permet d'interagir avec la géométrie et l'environnement de modélisation. Cependant, chaque logiciel disposant de sa propre API, il faut développer la capture du sous-contexte numérique pour une plateforme CAO spécifique. Nous avons choisi d'utiliser CATIA dans la 3DEXPERIENCE. L'API de ce logiciel est gérée par un langage spécialisé nommé EKL[21].

Une action EKL peut explorer toutes les informations relatives à une pièce dans la plateforme, des informations géométriques aux informations du systèmes PLM telle la maturité de la pièce. Ces informations peuvent ensuite être transmises de façon automatique à un serveur externe. Dans le cadre de notre démonstrateur, nous nous focaliserons sur la capture de l'arbre de construction qui structure l'historique de modélisation des features de la pièce en cours de conception.

Le modèle de données du sous-contexte numérique (2.2.3.4) est approfondi pour correspondre aux spécificités de notre environnement CAO (Figure 45). Pour des raisons pratiques, le logiciel (CATIA), l'application de modélisation (PartDesign) et le document CAO modifié sont fixes pour notre démonstrateur. Ils ne font donc pas parti des nœuds capturés par le système. Ce méta-modèle comporte donc des nœuds correspondant à des features de PartDesign (Fillet, Chamfer, Pad, Pocket, Remove et Sketch) et d'autres correspondant à des

[20] "Application Programming Interface" ou interface de programmation d'application.
[21] "Engineering Knowledge Language"

éléments structurels de CATIA (MechanicalRefPlane et GeometricalSet). Ce modèle peut évidemment être étoffé. Nous présentons ici la partie implémentée dans le démonstrateur.

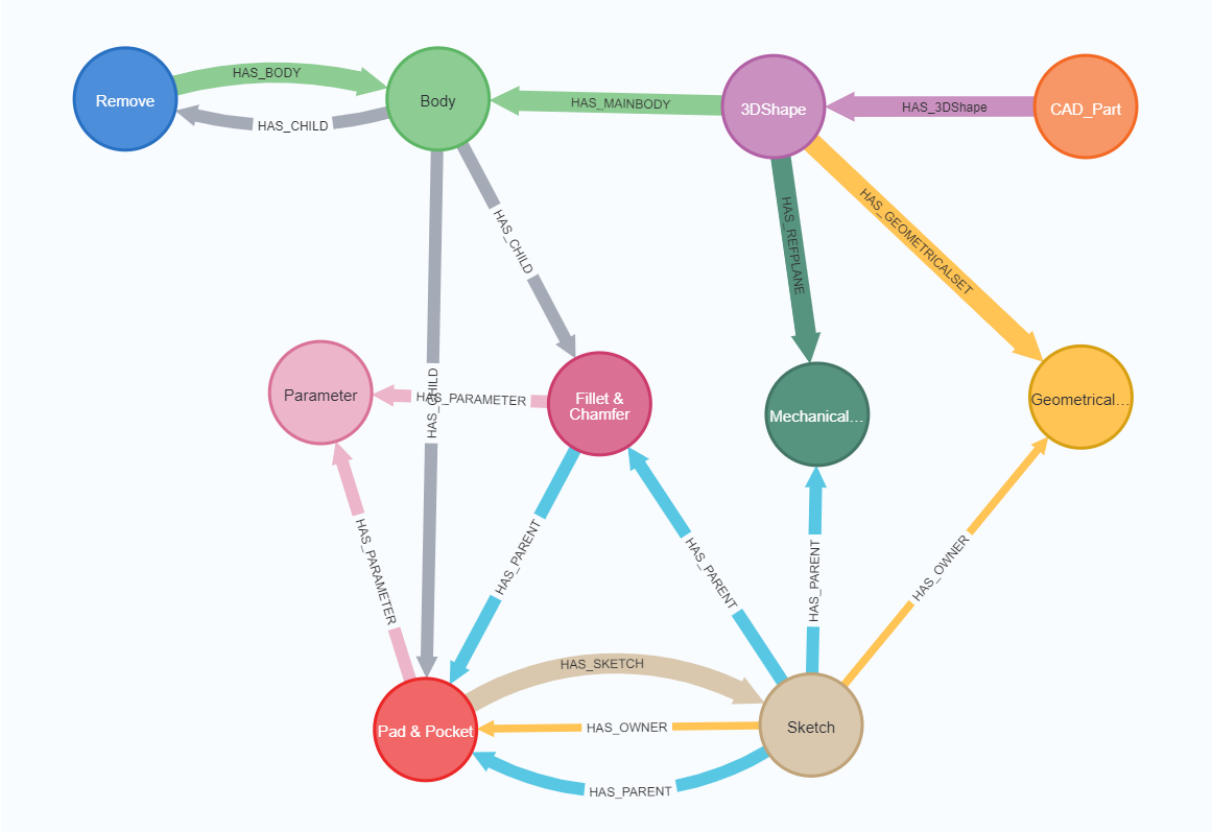


Figure 45 : Sous-contexte numérique capturé par notre démonstrateur

Le sous-contexte numérique est également enrichi des éléments de modélisation et de structure utilisés par CATIA. Les définitions des nœuds additionnels sont données par le **Tableau 13**. Les relations [:HAS_PARENT], [:HAS_CHILD] et [:HAS_OWNER] représentent des liens structurels entre les différentes données. La relation [:HAS_PARENT] indique la source des données nécessaires à une opération. La relation [:HAS_CHILD] pointe vers les sous-éléments constituant un ensemble. La relation [:HAS_OWNER] indique l'élément sous lequel un "Sketch" est placé. Des exemples de modèles simples et des graphes associés sont illustrés par les Figure 46 et Figure 47. On remarque que l'algorithme de capture s'adapte aux différentes stratégies de modélisation en représentant les relations entre les différentes features.

Remove	Opération booléenne de soustraction volumique entre deux corps
Pad / Pocket	Action de modélisation permettant d'ajouter ou de soustraire de la matière à un corps en fonction d'une distance et d'un contour fermé.
Fillet / Chamfer	Opération de finition réalisée sur l'arête d'une pièce
Sketch	Élément géométrique en deux dimensions
Geometrical Set	Elément de structure regroupant des données non volumiques
Mechanical Ref Plane	Plan de référence d'un corps permettant au logiciel de situer son orientation dans l'espace.

Tableau 13 : Définition des nœuds additionnels du cous-contexte numérique

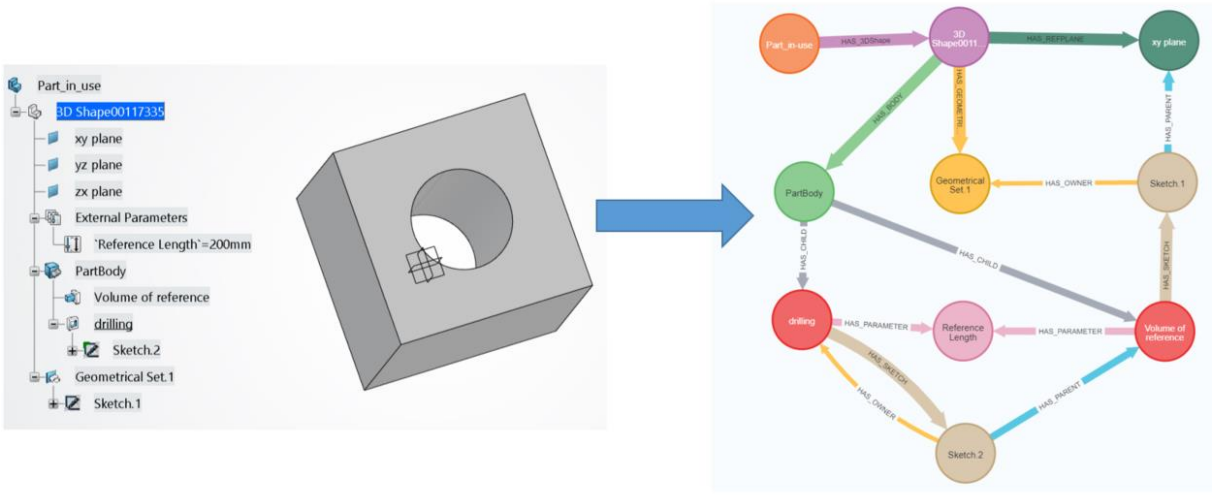


Figure 46 : Capture de l'arbre de conception exemple 1

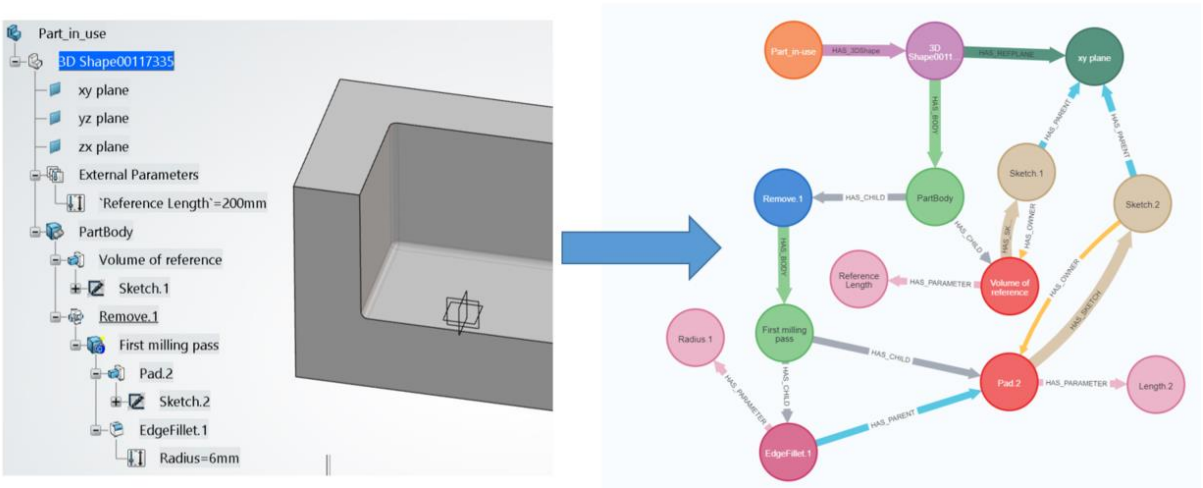


Figure 47 : Capture de l'arbre de conception exemple 2

Une fois l'arbre de conception capturé et modélisé dans le graphe, les nœuds du sous-contexte numérique sont reliés aux mots-clés présents dans leur nom. Le système met d'abord à jour la matrice d'adjacence avec une nouvelle dimension par nœud ajouté au graphe. Le processus de pondération de la matrice est ensuite réalisé afin d'obtenir une nouvelle matrice probabiliste de transition. Le processus de recherche de règles de conception reste similaire avec de nouveaux filtres contextuels correspondants au sous-contexte numérique.

L'implémentation de ce sous-contexte a un impact sur le processus de recommandation des règles de conception car il modifie de façon dynamique le système de recommandation. On ajoute donc une étape supplémentaire à la structure de recommandation des règles de conception (Figure 48).

Etape 7 : Le concepteur peut à tout moment de la modélisation lancer un scan de son modèle numérique. Cette action lance la capture du sous-contexte numérique dans la 3DEXPERIENCE et met à jour la matrice probabiliste de transition afin de prendre en compte les nouveaux nœuds et les nouvelles relations lors des prochaines recherches.

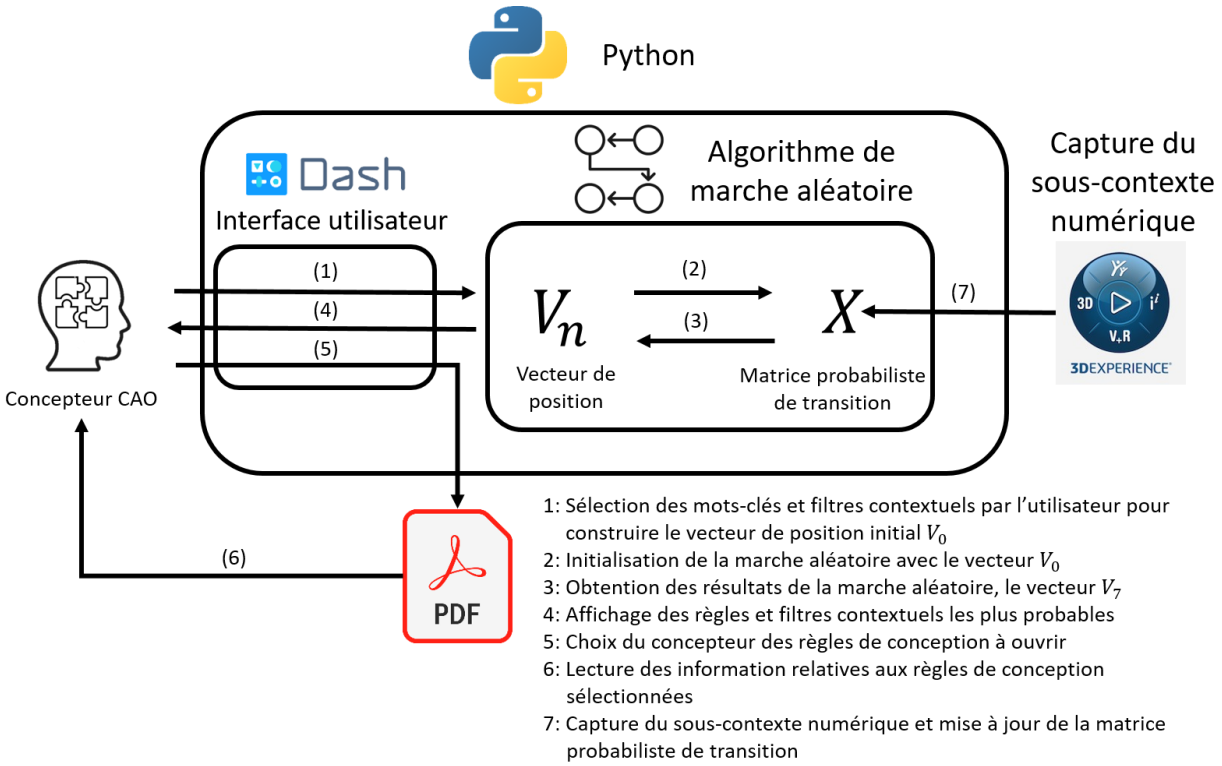


Figure 48 : Processus de recommandation des règles de conception avec capture du sous-contexte numérique

3.3 Synthèse de l'implémentation

Dans ce chapitre, nous avons présenté la démarche d'implémentation d'un démonstrateur de l'environnement de conception intelligent. Ce démonstrateur est une preuve de concept destinée à être utilisée dans le cadre d'expérimentations. La description de son implémentation est essentielle à la répétabilité de nos expérimentations et pourra servir de référence à des améliorations proposées par la communauté scientifique.

Dans un premier temps, ce chapitre détaille la structure logicielle du démonstrateur. Il est programmé en Python et structuré en quatre couches logicielles. La première couche constitue l'interface utilisateur. La deuxième couche regroupe toutes les bibliothèques et tous les outils spécialisés auxquels le démonstrateur a recours lors de son fonctionnement, notamment lors de l'analyse du langage naturel. La troisième couche représente l'interface entre le logiciel et la base de données orientée graphe. La quatrième couche est la base de données orientée graphe NEO4J [97].

La seconde partie de ce chapitre détaille l'implémentation des services identifiés lors de la description fonctionnelle de CACDA (Chapitre 2.1). L'écriture des quatre sous-contextes du processus de conception est présentée. Les sous-contextes sémantique et technique sont extraits de documents en langage naturel spécifiques au domaine d'application. Le sous-contexte social a été construit à partir de l'activité des sujets qui ont participé aux tests.

L’écriture du sous-contexte numérique a nécessité le développement d’un connecteur entre l’environnement de CAO 3DEXPERIENCE et le démonstrateur. Ainsi, notre démonstrateur représente, à une échelle expérimentale, l’ensemble des sous-contextes envisagés dans nos travaux.

Le choix du processus de recommandation et son implémentation sont aussi présentés. En adaptant le principe de l’algorithme ContextWalk proposé par Bogers [160], nous avons développé un système de recommandation prenant en compte l’ensemble de nos sous-contextes. Ce système prend en entrée une liste de mots-clés choisis par l’utilisateurs ainsi qu’une liste de filtres contextuels. Il en déduit les règles de conception les plus adaptés à ce besoin ainsi que des listes d’éléments contextuels d’intérêt, lesquels sont utilisés comme filtres contextuels pour la prochaine recherche.

La présentation de l’implémentation du démonstrateur de CACDA a fait l’objet d’un article publié dans la revue scientifique *Advanced Engineering Informatics* [176].

Notre démonstrateur technique représente donc toutes les fonctionnalités de l’assistant intelligent. Son utilisation lors de tests permettra de démontrer la faisabilité de cette approche. La mesure de son impact sur le processus de conception permettra de confirmer notre hypothèse de résolution.