Université de Montréal

ReLiS: un outil flexible pour réaliser des revues systématiques itératives et collaboratives

par
Bigendako Brice Michel

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

Février, 2018

RÉSUMÉ

Les Revues Systématiques (RS) offrent une méthode rigoureuse pour identifier et analyser les résultats dans la littérature relatifs à un sujet d'intérêt particulier. La réalisation d'une RS est connue pour être une tâche demandant beaucoup de temps et de travail qui nécessite un protocole bien documenté avec plusieurs itérations. Il suit un processus systématique pour atteindre des résultats reproductibles, objectifs et complets. Les outils qui permettent d'automatiser certaines tâches du processus sont d'une grande valeur pour les chercheurs. Cependant, d'importantes fonctionnalités liées à la réalisation de RS de manière collaborative et itérative font encore défaut dans les outils existants. Dans ce mémoire, nous présentons ReLiS, un outil pour installer et configurer automatiquement des projets RS à réaliser de manière collaborative et itérative en ligne. Le développement de ReLiS suit une approche de développement basée sur les modèles. Il dispose d'un éditeur de modèle spécifique au domaine adapté aux chercheurs qui réalisent des RS et d'une architecture qui permet l'installation progressive et la (re)configuration de plusieurs projets SR en cours de réalisation.

Mots clés : Revue littéraire systématique, étude systématique de cartographie de la littérature, développement dirigé par les modèles, installation automatique, génie logiciel.

ABSTRACT

Systematic Reviews (SR) provide a rigorous method to find and analyze the literature evidence relating to a particular topic of interest. Conducting SR is known to be an effort intensive and time-consuming endeavor that requires a properly documented protocol and several iterations to setup right. It follows a systematic process to achieve repeatable, unbiased and complete outcomes. Tools that help automate some tasks of the process are of tremendous value for researchers. However, important features related the conduction of SR in a collaborative and iterative way are still lacking in existing tools. In this thesis we present ReLiS, a tool to automatically install and configure SR projects to conduct them collaboratively and iteratively on the cloud. ReLiS is engineered following a model-driven development approach. It features a domain-specific modeling editor tailored for researchers who perform SR and an architecture that enables on-the-fly installation and (re)configuration of multiple concurrently running SR projects.

Keywords: Systematic literature review, systematic mapping study, model-driven development, automatic installation, software engineering.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

LISTE DES SIGLES

CMS	Content Management System
DSL	Domain Specific Language
DSM	Domain Specific Modeling
MDD	Model-Driven Development
MDE	Model-Driven Engineering
MVC	Model–View–Controller
QA	Quality Assessment
RS	Revue Systématique
SLR	Systematic Literature Review
SMS	Systematic Mapping Study

REMERCIEMENTS

Je tiens en premier lieu à remercier mon directeur de recherche Mr. Eugene Syriani pour le soutien continu, la confiance, la patience et les conseils qui ont constitué un apport considérable sans lequel ce travail n'aurait pas pu être mené au bon port.

J'aimerais également remercier tous les membres du laboratoire GEODES qui, avec leurs présences, ont permis de créer un environnement amical, convivial et très favorable à l'aboutissement de ce travail.

Je tiens aussi à remercier mes parents, frères et sœurs ainsi que la famille NAHIMANA Elie pour leur soutien et encouragements tout au long de mon parcours.

Enfin, je remercie tous ceux qui, de près ou de loin, ont contribué à la réalisation de ce travail.

CHAPITRE 1

INTRODUCTION

1.1 Contexte

Toute recherche commence d'une certaine manière par une revue de la littérature. Lorsqu'un chercheur veut se pencher sur un sujet de recherche dans un domaine quelconque, il commence par passer en revue la littérature sur le sujet. Cette revue lui permet d'identifier ce qui a été déjà réalisé sur le sujet, pour l'éclairer sur de nouvelles orientations de recherche. C'est également le cas dans le milieu des entreprises. Lorsqu'un professionnel veut adopter une nouvelle technologie ou une nouvelle pratique, il analyse les études qui ont été déjà réalisées en pratique. Il évalue les avantages et les inconvénients des solutions existantes afin de le guider dans ses choix. Cette revue rejoint par ce fait le Génie Logiciel Basé sur l'Évidence (Evidence-Based Software Engineering — EBSE) dont le but principal est de fournir les moyens de pouvoir intégrer les meilleurs résultats des recherches et l'expérience pratique dans le processus de prise de décisions concernant le développement et la maintenance de logiciels [14].

Dans le but de rendre ces revues complètes, objectives et reproductibles, quelques méthodologies existent [29, 42]. Elles proposent des processus à suivre avec des activités bien planifiées à exécuter de manière systématique. Une revue suivant ces processus devient donc *systématique* par opposition aux revues ad hoc qui ne suivent pas de stratégie bien planifiée afin de pouvoir retracer le protocole suivi [47]. Une Revue Systématique (RS) est donc une manière d'identifier, évaluer et interpréter toutes les recherches relatives à une question de recherche ou un phénomène en particulier [29]. Elle s'inscrit dans la catégorie d'*étude secondaire* et les différentes études concernées par la revue sont quant à elles des *études primaires*. Elle peut aussi être une *étude tertiaire* lorsque

les études concernées sont elles-mêmes des RS tel que [31].

Depuis leur introduction en génie logiciel, les RS ont attiré beaucoup l'attention des chercheurs dans le domaine. Cela est reflété par le nombre de RS publiées qui continue d'augmenter [21], sans oublier l'intérêt que les journaux et conférences leurs portent en ajoutant des types de publications et des sessions qui leurs sont dédiées [16]. Malgré cette importance, la réalisation d'une RS reste une tâche qui demande beaucoup de travail avec un fort risque d'introduction d'erreur [9, 11, 29]. Pour assurer la complétude de la revue, il faut identifier toutes les études candidates, ce qui peut passer par l'analyse de centaines voire de milliers de publications selon le domaine ciblé. Également, certaines activités du processus de réalisation d'une RS doivent être effectuées par deux ou plusieurs personnes, puis comparer les résultats pour éviter toute introduction de biais par le travail d'un seul chercheur. Cette exigence accroît la charge de travail de chaque participant dans l'étude et ajoute une complexité liée à la gestion de cette collaboration. Il faut également documenter tout le processus pour assurer la reproductibilité de l'étude et démontrer qu'elle a été réalisée correctement.

Pour ces raisons, des outils pour aider dans la gestion des références, la sélection des études primaires, la collectes des métadonnées et le rapport des résultats sont d'une importance capitale. Quelques outils sont déjà disponibles [38]. Une analyse de ces outils a montré qu'ils ne couvrent pas suffisamment les réels besoins des RS, surtout dans la gestion du processus itératif et la collaboration [1].

1.2 Problématique et solution proposée

La réalisation d'une RS est un processus composé de plusieurs activités à compléter de manière itérative. Les activités sont d'abord planifiées dans un protocole avant d'être exécutées pendant une phase de réalisation proprement-dite de la revue. Malgré cette exigence de planification, il est souvent impossible de planifier correctement la to-

talité du protocole à suivre dès le début du projet. Le protocole a souvent besoin d'être adapté à certains cas spéciaux qui peuvent apparaitre durant la réalisation. Également, pour s'assurer que le protocole corresponde à la revue à réaliser et qu'il est bien compris par les chercheurs participants dans l'étude, Staples et al. [47] recommandent de faire une étude pilote. L'issue de cette étude pilote peut conduire à une amélioration du protocole. Par conséquent, il est crucial qu'un outil de RS puisse modifier le protocole et s'adapter à la RS à réaliser. Les outils de RS doivent donc supporter le caractère itératif, faciliter la collaboration et permettre le pilotage des projets. Ils doivent également être très configurables pour être adaptés aux spécificités des différents RS.

Pour atteindre ces objectifs, le développement dirigé par les modèles (Model Driven Development — MDD) peut aider en générant automatiquement l'application à partir d'un modèle spécifique au domaine [23]. Dans ce mémoire, nous suivons l'approche MDD pour configurer—et reconfigurer, au besoin— un projet de RS afin d'atteindre un protocole adapté à l'étude à réaliser. Nous avons développé un outil nommé ReLiS *(abréviation de "Revue Littéraire Systématique")*, qui offre un environnement intégré basé sur un langage spécifique au domaine (DSL) permettant l'ajout et la configuration de projets de RS. Il permet à un groupe de chercheurs de réaliser une RS en collaboration avec une application personnalisée à leur sujet de recherche.

1.3 Contribution

L'objectif de ce mémoire est de faciliter le travail des chercheurs dans la réalisation de RS avec l'utilisation d'outils adaptés aux exigences des RS. La contribution de ce mémoire est la création d'un outil qui aide dans la réalisation de RS. L'outil supporte le caractère itératif de processus de réalisation de RS et permet le pilotage des projets RS. Il appuie la collaboration et est basé sur une architecture flexible permettant la reconfiguration de projets RS déployés sur le cloud.

Une partie des résultats de ce mémoire ont été publiés dans [6].

1.4 Organisation du mémoire

Ce mémoire est organisé comme suit. Dans le Chapitre 2 nous présentons les informations pertinentes sur les RS, le MDD et les outils utilisés pour réaliser les RS avec un aperçu sur les travaux connexes. Dans le Chapitre 3 nous présentons la DSL utilisée pour rendre plus flexible la configuration de projets de RS. Dans le Chapitre 4 nous décrivons l'architecture dynamique mise en place pour supporter les exigences du processus de RS. Dans le Chapitre 5 nous faisons une évaluation de l'outil développé sur son adaptation à la réalisation de RS et une comparaison avec les outils existants. Enfin, nous concluons dans le Chapitre 6.

CHAPITRE 2

ÉTAT DE L'ART

Pour introduire les notions principales abordées dans ce mémoire, ce Chapitre présente les RS, leur processus de réalisation et les outils qui sont utilisés pour aider dans leur réalisation. Nous présentons également le MDD avec son architecture pour la génération d'applications.

2.1 Revues systématiques

Une RS vise à faire une récapitulation des études existantes sur une technologie ou un traitement dans un domaine quelconque. Cette récapitulation permet d'identifier les lacunes et les forces dans ces études. Elle guide dans la prise de décisions et il met en évidence les domaines à approfondir en fournissant un cadre solide pour positionner de façon appropriée de nouvelles activités de recherche.

2.1.1 Type de revues systématiques

Suivant l'objectif de l'étude, une RS peut se présenter sous la forme d'une *Revue Littéraire Systématique* (Systematic Literature Review — SLR) ou d'une *Étude Systématique de Cartographie de la Littérature* (Systematic Mapping Study — SMS). Une *SLR* se concentre sur la collecte, la synthèse et l'analyse des résultats des études primaires. Elle étudie les articles en détail pour pouvoir extraire et analyser les résultats empiriques contenus dans ces études. Utilisée pour structurer un domaine de recherche, une *SMS* consiste à classifier les études primaires dans des catégories, pour montrer les tendances de recherches dans le domaine [43]. Elle fournit un large aperçu d'un domaine de recherche afin d'établir si des recherches existent sur un sujet et fournir une indication

sur la quantité de ces recherches. Elle n'étudie pas les articles en détail comme le fait une SLR, ce qui fait qu'elle demande moins de temps et de travail comparé à une SLR.

Malgré que ces deux types de RS diffèrent en termes d'objectifs, ils partagent les mêmes activités composant le processus de réalisation avec quelques nuances, surtout au niveau de l'identification et de la sélection d'études primaires, ainsi qu'au moment de la collecte et analyse des résultats. Une description détaillée des différences entre ces deux types est présentée dans [29, 43].

Dans le but d'aider les chercheurs à bien réaliser une RS, Kitchenham et Charters ont proposé une méthodologie [29] (avec sa version antérieure [26]). C'est une adaptation des méthodologies utilisées en médecine [2, 19, 24] où la pratique est mature, pour adresser des besoins spécifiques au génie logiciel. Cette méthodologie spécifie les différentes phases et activités à accomplir pour bien réaliser une SLR. Une autre méthodologie qui se concentre sur les SMS a été proposé par Petersen et al. [43] (avec sa version antérieure [42]). En plus d'aider dans la réalisation de SMS, cette méthodologie guide aussi les chercheurs à choisir entre réaliser une SLR ou une SMS suivant l'objectif visé par leur étude. A côté de ces deux méthodologies les plus populaires en génie logiciel [43], il existe d'autres méthodologies leur comparaison est présentée dans [28].

2.1.2 Processus de réalisation d'une revue systématique

Les méthodologies de réalisation de RS [29, 42] recommandent un processus composé d'activités à exécuter dans trois phases : *la planification*, *la réalisation* et *le rapport* de la revue.

2.1.2.1 Planification de la revue

Pour prévenir l'introduction de tout biais par les chercheurs lors de la réalisation d'une RS, le contexte, les questions de recherches et les décisions sur la stratégie à

adopter doivent être préalablement définis dans une phase de planification [9]. Toutes ces décisions sont rapportées dans un *protocole de la revue* qui sert de référence à la suite du processus. Dans cette partie, nous présentons les principales composantes de ce protocole.

Définition du besoin de réaliser une RS La réalisation d'une RS commence par la présentation de l'intérêt de l'étude et le contexte dans lequel l'étude se situe. Cet intérêt exprime le besoin à faire une récapitulation non biaisée de toutes les informations relatives à un sujet particulier de manière approfondie [26]. Les raisons pouvant motiver cet intérêt peuvent être, entre autres :

— le besoin de tirer des conclusions générales sur des études qui ont été réalisées sur un sujet particulier. Ces conclusions peuvent être utilisées en entreprise pour aider dans la prise de décision sur l'adoption de nouvelles solutions.

— la volonté d'évaluer ce qui a été réalisé dans un domaine en vue de positionner de nouvelles recherches. Par exemple, un étudiant en recherche qui fait un état de l'art pour son sujet de recherche.

Spécification des questions de recherche Lors de la réalisation d'une RS, il est important de spécifier des questions de recherche claires et adaptées à l'étude, car ce sont ces questions qui vont guider tout le processus [29] :

— le processus de recherche doit identifier les études primaires portants sur les questions de recherche ;

— le processus d'extraction des données doit être choisi dans le but d'extraire des informations permettant de répondre aux questions de recherche ;

— le processus d'analyse doit synthétiser les résultats de telle sorte qu'ils puissent répondre aux questions de recherche.

La forme des questions de recherche dépend fortement du type d'étude à réaliser. Les

SMS ont des questions beaucoup plus générales car l'objectif est de trouver les tendances des recherches dans le domaine. Par contre, pour les SLR, les questions de recherche sont très spécifiques car elles visent à bien analyser les résultats dans les études primaires et tirer des conclusions sur bases de ces analyses [43].

Stratégie d'identification des études candidates L'identification des études candidates cherche à trouver toutes les études relatives aux questions de recherche en utilisant une stratégie impartiale afin assurer la rigueur et la complétude de l'étude.

Dans la phase de planification, il faut décider sur la stratégie de recherche. Les chercheurs peuvent choisir entre une recherche manuelle, une recherche automatique dans les librairies électroniques ou la combinaison des deux.

Recherche automatique Il faut d'abord identifier les sources (librairies électroniques) adaptées à l'étude. Il en existe plusieurs en génie logiciel tels que : *IEEExplore* [1], *ACM Digital library* [2], *Google scholar* [3], *Citeseer library* [4], *Inspec* [5], *ScienceDirect* [6], *EI Compendex* [7], *SpringerLink* [8], *Engineering Village* [9], *et Scopus* [10].

Après le choix des sources, les chercheurs doivent formuler des requêtes de recherche à appliquer aux sources choisies. Une requête de recherche comprend des mots clés qui sont reliés entre eux avec les connecteurs binaires AND et OR. Ces mots clés sont choisis en se basant sur l'objectif de l'étude reflété dans les questions de recherche. Une bonne manière d'identifier ces mots clés est d'utiliser

1. https://ieeexplore.ieee.org
2. https://dl.acm.org
3. https://scholar.google.com
4. https://citeseer.ist.psu.edu
5. https://theiet.org/resources/inspec
6. https://sciencedirect.com
7. https://elsevier.com/solutions/engineering-village/content/compendex
8. https://link.springer.com
9. https://www.engineeringvillage.com
10. https://scopus.com

l'approche PICO, proposée par Kitchenham et Charters [29], qui les structure en termes de population, intervention, comparaison et résultats.

Recherche manuelle Cette stratégie peut être utilisée pour compléter la recherche automatique ou elle peut être choisie comme seule stratégie de recherche à utiliser. Elle consiste à parcourir les conférences et journaux du domaine concerné et identifier les études qui sont relatives à l'étude. La recherche manuelle inclut aussi le processus l'échantillonnage en boule de neige *(snowball sampling)* [22].

Le *snowballing* s'appuie sur un échantillon d'études déjà identifiées pour trouver d'autres études candidates. Il comprend : le *forward snowballing* qui cherche parmi les articles qui citent les études identifiées dans l'échantillon, le *backward snowballing* qui cherche parmi les articles cités par les études identifiées dans l'échantillon.

Stratégie de sélection des études primaires Une fois les études candidates identifiées lors de la recherche, on sélectionne celles qui sont pertinentes pour la RS. Des critères de sélection sont utilisés pour inclure ou exclure des études de la revue.

Dans la phase de planification, il faut décider sur la stratégie à adopter. Cette stratégie comprend la détermination de phases de sélection (sélection par titre, par abstract ou lecture complète), le nombre de personnes qui vont évaluer chacune des études candidates et comment les conflits de sélection vont être résolus. La planification comprend aussi, la définition des critères de sélection (critères d'inclusion et d'exclusion), qui sont généralement choisis sur base des questions de recherche.

Stratégie d'évaluation de la qualité des études primaires L'évaluation de la qualité est utilisée pour déterminer la pertinence des études sélectionnées en appliquant un score suivant leur niveau de qualité vis à vis de la RS et les questions de recherche. Cette activité concerne particulièrement les SLR, mais n'est pas nécessaire pour les SMS car

ces derniers s'intéressent beaucoup plus aux tendances des publications. Elles ont un faible intérêt dans l'analyse de la rigueur et de la pertinence des résultats dans les études primaires [43]. Selon Kitchenham et al. [29], la qualité d'une étude se rapporte au fait qu'elle minimise le biais et maximise la validité et la possibilité de généralisation.

L'évaluation de la qualité peut être utilisée pour compléter la phase de sélection en excluant des études qui n'atteignent pas un certain seuil de qualité [13]. Elle peut aussi être utilisée pour mesurer l'importance des études et tenir compte de cette importance lors de l'analyse des résultats. Dans ce cas, elle est utilisée pour compléter l'extraction des données. Les études avec un faible score ne seront pas exclues, mais seront évaluées en tenant compte de leur score de qualité [28].

L'évaluation de la qualité se fait à l'aide d'une liste de contrôle (check-list). Celle-ci se présente sous forme d'un questionnaire constitué de questions pour évaluer la qualité de chaque étude primaire selon des facteurs choisit. Les réponses à ces questions ont chacune une valeur qui permettra de mesurer la qualité de chaque étude.

Dans le protocole, il faut définir les questions et réponses constituant la liste de contrôle, ainsi que le choix d'une stratégie d'évaluation. Cette stratégie indique le nombre de participants dans le processus, la validation des résultats ainsi que la résolution de conflits d'évaluation, dans le cas de plusieurs évaluateurs par étude. Il faut également définir, si appliqué, un seuil minimal du score et le sort des études primaires qui ne l'auront pas atteint.

Stratégie d'extraction et synthèse des données L'extraction des données consiste à extraire toute information nécessaire des études primaires, pour répondre aux questions de recherche. L'extraction des données se présente sous forme d'un schéma de classification à compléter pour chaque étude retenue. En plus de collecter les informations spécifiques à l'étude, il faut aussi inclure les méta-informations comme : le titre, auteurs,

les détails de la publication et la date d'extraction.

Lors de la planification, en plus de définir le processus d'extraction et sa validation, il faut concevoir le formulaire d'extraction des données ou schéma de classification qui va permettre d'extraire avec précision toutes les informations nécessaires pour l'étude. Un formulaire d'extraction des données est constitué par des catégories ou types d'informations à extraire dans chaque étude primaire. Pour certaines catégories, on peut définir la liste des valeurs possibles, cela facilite l'extraction des données et l'analyse des résultats après extraction. Pour définir les catégories ce formulaire, Petersen et al. [42] propose un processus systématique utilisant l'extraction des mots-clés dans les abstracts sur un échantillon des études primaires. Les mots-clés extraits sont regroupés pour former un ensemble de catégories représentatif de la population des études dans le domaine ciblé. Dans certain cas, lorsque les abstracts ne fournissent pas les mots-clés significatifs, le processus peut nécessiter la lecture du contenu des articles pour identifier les catégories.

Après l'extraction, une synthèse de ces données permet de rassembler et condenser les résultats afin de faciliter leur analyse. Cette synthèse peut se faire sous formes de tableaux ou de graphiques suivant les catégories identifiées dans le formulaire d'extraction et l'analyse qu'on veut faire de ces résultats. Il faut donc planifier une stratégie qui présente comment les données extraites seront synthétisées. Cette stratégie précise, entre autres, la manière dont les résultats seront regroupés et représentés. Il faut également préciser si ces résultats vont nécessiter une méta-analyse et les techniques qui seront utilisées pour cette méta-analyse.

Stratégie de dissémination Après la réalisation de la revue, un rapport est produit pour présenter les résultats de l'étude. Il est important que ce rapport soit diffusé de façon efficace afin de le faire parvenir aux lecteurs intéressés par le sujet de la RS. Il est recommandé de planifier une stratégie de dissémination des résultats. Cette stratégie

définit ce qui sera publié, comment cela sera présenté et la venue de cette publication.

2.1.2.2 Réalisation de la revue

Après avoir planifié la RS dans le protocole, les chercheurs peuvent passer à la phase de réalisation de la RS. Ils vont donc exécuter toutes les activités comme elles ont été planifiées dans le protocole. Le processus étant itératif, le protocole peut être modifié pendant la réalisation pour apporter des améliorations ou des adaptations. Par contre, il faut bien documenter ces modifications et justifier leur bien fondé.

Identification des études candidates Les chercheurs appliquent la stratégie adoptée dans la planification afin identifier les études candidate pour la revue. Si la recherche automatique a été choisie, les chercheurs appliquent les requêtes de recherche sur les sources sélectionnées et récupèrent les études répondant aux critères dans les requêtes. Avec la recherche manuelle, les chercheurs récupèrent les études candidates avec la stratégie choisie et doivent à chaque fois mentionner la source de l'étude. Tout changement sur la stratégie planifiée doit être bien documentée.

Sélection des études primaires Le processus de sélection se fait généralement en plusieurs phases. La sélection commence par inclure ou exclure chaque étude candidate sur base du titre, de l'abstract et, des fois, la lecture complète pour les publications qui auront passé les phases précédentes.

La sélection est une tâche collaborative afin d'éviter que l'étude soit biaisée par le travail d'un seul chercheur. Dans la mesure du possible, au moins deux chercheurs doivent décider de l'inclusion ou de l'exclusion de chaque étude primaire. A chaque décision d'inclusion ou d'exclusion, il faut associer le critère de sélection appliqué parmi ceux qui ont été définis dans le protocole. Cela aide à bien documenter le processus de sélection Dans le cas de conflits de décision, les chercheurs qui ont participé dans la sélection

peuvent organiser des sessions de concertation pour résoudre ces conflits, ou ajouter une autre personne pour l'arbitrage suivant ce qui a été décidé dans la stratégie de sélection. Il faut aussi mesurer l'accord entre les chercheurs pendant la sélection en utilisant le *Kappa de Cohen* [12] qui quantifie le niveau de consentement. C'est un indicateur sur la clarté et la compréhension de la stratégie de sélection par les participants, ainsi que sur leur maitrise du domaine. Dans le cas d'un seul chercheur (ex. chercheur apprenti ou novice), il pourrait discuter des inclusions et exclusions avec un expert dans le domaine. L'expert choisi peut par la suite valider chaque phase de sélection sur un échantillon et comparer les décisions d'exclusion [29].

Évaluation de la qualité L'évaluation de la qualité commence dans la phase de planification par la définition de la liste de contrôle. Les chercheurs répondent aux questions constituant la liste pour chaque étude incluse dans la phase de sélection. Chaque réponse est associée à une valeur. Après avoir complété la liste de contrôle, un score est calculé pour chaque étude en additionnant les valeurs des réponses choisies. C'est ce score qui représente la qualité de l'étude. Dans cette phase, les chercheurs peuvent aussi décider d'exclure certaines études ayant une évaluation trop faible, ne répondant donc pas aux exigences de la revue. Si chaque étude est évaluée par plusieurs chercheurs, il faut aussi inclure des séances de concertation pour résoudre d'éventuels conflits d'évaluation.

Extraction et synthèse des données Avec le formulaire d'extraction défini durant la planification, les chercheurs parcourent les études primaires sélectionnées pour extraire les informations nécessaires et les classer dans les catégories correspondantes.

Comme pour la sélection des études primaires, si possible, deux ou plusieurs chercheurs peuvent effectuer l'extraction des données en parallèle. Dans le cas de conflits, ils peuvent organiser des sessions de concertation pour les résoudre, ou ajouter une autre personne pour l'arbitrage suivant ce qui a été défini dans le protocole. Pour réduire le

temps requis pour cette activité, Brereton et al. [9] conseille d'avoir un chercheur qui extrait les données et un autre qui valide les données extraites.

Après l'extraction, les données extraites sont synthétisées dans un format qui facilite l'analyse des résultats pour aider à répondre aux questions de recherche.

2.1.2.3 Rapport de la revue

Après la réalisation de la revue, un rapport de l'étude est produit. Ce rapport présente non seulement les résultats, mais documente aussi les différentes étapes de l'étude, avec une analyse et une discussion sur les résultats. Staples et al. [47] recommande fortement de documenter tout ce qui s'est passé durant la réalisation de la revue et surtout les déviations au protocole. Une bonne documentation du processus permet aux lecteurs d'évaluer la rigueur et la complétude de l'étude et ainsi juger sur la crédibilité et l'importance des résultats obtenus. Une structure et le contenu d'un rapport est proposé par Kitchenham et Charters [29].

Si une évaluation du protocole a été réalisé, il est recommandé qu'une évaluation du rapport soit faite par la même équipe, pour déterminer si l'étude respecte le protocole planifié [9].

2.2 Outils pour réaliser les RS

Le processus de réalisation de RS est complexe. Il comporte beaucoup d'étapes, avec des activités qui demandent beaucoup de temps. À cela s'ajoute le souci de traçabilité pour les rendre reproductibles. Il est donc important d'avoir un support logiciel pour automatiser le processus et aider les chercheurs à être plus efficace et à réaliser des RS de qualité. Dans cette partie nous présentons les fonctionnalités nécessaires pour ces outils de RS puis nous allons faire un aperçu sur les outils qui sont actuellement utilisés.

2.2.1 Fonctionnalités

Étant donné les exigences spécifiques aux RS, des études ont identifié les fonctionnalités que devrait offrir un outil logiciel pour aider dans la réalisation des RS [1, 39, 40]. En combinant les résultats de ces études et les recommandations dans les méthodologies pour réaliser les RS, nous avons synthétisé une liste des fonctionnalités pour ces outils.

Les fonctionnalités sont regroupées dans deux catégories :

2.2.1.1 Fonctionnalités liées au processus de réalisation d'une RS

Développement du protocole de la revue L'outil doit faciliter le développement du protocole de la revue de manière collaborative et le suivre durant tout le processus de réalisation de la RS.

Identification des études candidates L'outil doit supporter les différentes stratégies d'identification des études primaires, comme la recherche manuelle ou la recherche automatique, avec la possibilité d'importer les études identifiées dans l'outil. L'outil devrait aussi être capable de détecter les doublons dans les études identifiées.

Sélection des études primaires L'outil doit permettre aux chercheurs d'inclure et exclure, de manière collaborative, des études primaires suivant une série de critères de sélection prédéfinie. Il doit également détecter les conflits de décision et permettre la résolution de ces conflits.

Évaluation de la qualité L'outil doit supporter une évaluation de la qualité des études primaires sélectionnées en permettant la mise en place d'une liste de contrôle. Avec cette liste, les chercheurs évaluent les études en leur affectant un score de qualité.

Extraction des données L'outil doit faciliter l'extraction et la sauvegarde de données grâce à des formulaires interactifs adaptés aux types de données à extraire.

Synthèse des résultats L'outil doit être capable d'effectuer automatiquement une synthèse des données extraites sous forme de graphiques ou tableaux pour une bonne interprétation des résultats et aider dans leur analyse.

Rapport de la revue L'outil doit aider dans la production du rapport de la revue en fournissant les informations nécessaires pour le rapport (synthèse des résultats et processus de l'étude).

2.2.1.2 Fonctionnalités liées à la gestion du processus d'une RS

Collaboration et sécurité L'outil doit permettre à plusieurs personnes de travailler en parallèle sur les mêmes tâches et protéger l'accès à l'outil avec un système d'authentification permettant aux utilisateurs d'accéder aux fonctionnalités suivant leur rôle dans l'étude. L'outil doit également supporter la gestion de conflits, par la détection et l'assistance dans la résolution de conflits de décisions pour les utilisateurs qui travaillent sur les mêmes tâches.

Traçabilité L'outil doit assurer la traçabilité des opérations effectuées et des données extraites. Cette traçabilité est faite en associant les activités réalisées aux personnes qui les ont exécutées et en établissant un lien entre les données extraites et les études primaires.

Supporter plusieurs projets Pour permettre aux chercheurs d'avoir un espace centralisé pour réaliser et présenter les résultats des RS, l'outil doit supporter la réalisation de plusieurs projets ayant des exigences et des configurations différentes.

Maintenance des données L'outil doit assurer une bonne sauvegarde des données des revues réalisées pour permettre une consultation ultérieure ou une réutilisation dans d'autres études.

Pilotage de la revue L'outil doit supporter le pilotage des différentes activités du processus de la RS. Les utilisateurs réalisent une activité du processus (ex. sélection

des études primaires) sur un échantillon des études primaires pour juger si la stratégie adoptée est conforme aux objectifs de l'étude et si elle est bien comprise par les chercheurs participants dans l'étude. Si les résultats ne sont pas concluants, ils peuvent décider d'améliorer la stratégie pour qu'elle soit la plus adaptée à la RS à réaliser.

2.2.2 Outils existants

Plusieurs outils sont actuellement utilisés lors de la réalisation des RS [39]. Dans ces outils on retrouve des logiciels d'usage général tels que les éditeurs de texte (MS Word), les tableurs (MS Excel), les outils de gestion des références (EndNote, RefWorks), les logiciels de statistique (SPSS, MPlus, R), mais aussi des outils développés spécifiquement pour supporter les RS. Nous allons nous intéresser sur cette dernière catégorie car, c'est dans cette classe que s'oriente notre contribution.

Quelques études ont été réalisées pour identifier et évaluer les outils existants. Dans [39], Marshall et al. ont réalisé une SMS afin d'identifier les outils de RS utilisés en génie logiciel, puis dans [40] ils ont conduit une enquête auprès des chercheurs ayant réalisé des RS dans le domaine de la santé et des sciences sociales afin d'identifier les outils qu'ils utilisent et les fonctionnalités qu'ils jugent indispensables. Dans [1], Al-Zubidy et al. ont évalué les forces et faiblesses de ces outils.

Parmi les outils identifiés, *Eppi Reviewer*[11] est une application web qui permet de réaliser une RS de manière collaborative. Il a été conçu pour les RS dans le domaine de la santé et les sciences sociales. Il supporte la gestion des références, la recherche et sélection des études primaires, l'extraction des données et fait une bonne synthèse des données extraites [49]. Des outils similaires et complets existent mais ils sont aussi payants et sont orientés vers le domaine de la santé et les sciences sociales. C'est le cas

11. `https://eppi.ioe.ac.uk`

de *Convidence* [12], *DistillerSR* [13] ou *Review Manager (RevMan)* [14].

Dans le but d'offrir des fonctionnalités adaptées au génie logiciel, d'autres outils ont été développés en s'inspirant des méthodologies conçues pour ce domaine (voir Section 2.1.1). *Parsifal* [15] est un outil en ligne qui permet de réaliser des SLR de manière collaborative. Il dispose, entre autres, d'un module de recherche automatisé d'études primaires qui génère des requêtes de recherche puis interroge de façon automatisée une série de librairies électroniques. Par contre, il est limité au niveau de l'extraction des données car ses formulaires d'extraction ne supportent pas de catégories complexes comme la possibilité d'assigner des valeurs multiples dans la même case, des catégories dont les valeurs dépendent des valeurs d'autres catégories, ou des catégories contenant des sous catégories. *StArt* [20] supporte aussi une grande partie des fonctionnalités énumérées dans la Section 2.2.1. Par contre, il reste aussi limité au niveau de l'extraction des données. *SLR-Tool* [16] lui aussi supporte les différentes activités du processus de réalisation de RS. Son plus grand défaut est qu'il ne permet pas la collaboration alors que c'est l'une des principales exigences d'un outil de RS. *SLRTOOL* [3] supporte aussi toutes les phases du processus, mais ne permet pas la sélection des études primaires en plusieurs phases. D'un point de vue d'utilisabilité, il nécessite l'ajout des études primaires une à une, ce qui le rend non adapté aux RS avec un grand nombre d'études à évaluer. *SLuRp* [7] se distingue au niveau de la visualisation des résultats synthétisés. Il permet de générer automatiquement le rapport de la revue en LaTeX. Par contre, il n'assure pas la traçabilité pour identifier la source des données extraites.

12. `https://www.covidence.org`
13. `https://www.evidencepartners.com`
14. `https://community.cochrane.org/tools/review-production-tools/revman-web`
15. `https://parsif.al`

2.2.3 Synthèse

Avec cet aperçu des outils existants, on note que, malgré les efforts de développement des outils pour réaliser les RS, aucun ne répond correctement à toutes les exigences des RS. En particulier les critères de réalisation de RS que sont la collaboration, le caractère itératif du processus nécessitant la reconfiguration du protocole durant la réalisation et le pilotage. Une évaluation des outils existants est présentée dans la Section 5.1.

2.3 Développement dirigé par les modèles

Le MDD est une approche de développement de logiciel qui utilise les modèles comme élément central du processus de développement [8]. L'objectif visé par le MDD est de réduire l'écart entre le domaine d'application et l'implémentation d'une solution grâce à la Modélisation Spécifique au Domaine (Domain Specific Modeling — DSM) [23]. La DSM élève le niveau d'abstraction au-delà du langage de programmation en spécifiant la solution dans un langage qui utilise les concepts et les règles spécifiques au domaine d'application. Cette abstraction est réalisée à l'aide des modèles. Les modèles représentant la solution sont, par la suite, utilisés pour générer automatiquement ou semi-automatiquement l'implémentation de la solution grâce à la transformation de modèle [8]. Dans le but de générer du code, Kelly et al. [23] proposent une architecture à trois niveaux comprenant : un langage spécifique au domaine, un générateur de code et un framework du domaine.

2.3.1 Langage spécifique au domaine

Le langage spécifique au domaine (Domain Specific Language — DSL) permet de représenter la solution à un niveau d'abstraction du domaine d'application. Cette représentation est faite en utilisant les règles et notations qui sont spécifiques au domaine, ce

qui permet aux experts du domaine de participer directement au développement de la solution.

Une DSL est constituée de trois éléments principaux : la syntaxe abstraite, la syntaxe concrète et la sémantique [17]. La syntaxe abstraite définit les concepts, la structure et les règles du langage. Cette syntaxe est définie par un métamodèle et tous les modèles représentant la solution doivent être conformes à celui-ci. Un métamodèle est souvent présenté sous forme d'un diagramme de classe UML qui définit les types du langage, leurs propriétés et leur composition. La syntaxe concrète définit les notations utilisées pour représenter les éléments du langage. La sémantique définit la signification des différents éléments du langage dans le domaine d'application. Lorsque la syntaxe est textuelle on définit une grammaire où les règles de production spécifient le mapping de chaque élément du métamodèle à des mots-clés.

Par exemple, **Xtext** [5] est un framework open-source utilisé dans le développement des langages textuelles qui fait partie du projet EMF (Eclipse Modeling Framework). Il permet aux développeurs de définir une grammaire d'un langage. Cette grammaire décrit comment un modèle Ecore est dérivé d'une syntaxe textuelle. L'Annexe A montre un exemple de grammaire Xtext. Avec cette grammaire, il génère un parseur et un éditeur intelligent du langage défini. L'éditeur offre une coloration syntaxique, l'auto-complétions et une validation affichant des messages d'erreurs en cas de saisies erronées. Xtext s'intègre facilement avec les outils de modélisations basé sur Eclipse comme les langages de transformation de modèle à modèle (exemple : ATL) ou de modèle à texte (exemple : Xtend) [15].

2.3.2 Génération de code

L'un des avantages du MDD est qu'il permet la manipulation automatique des modèles grâce aux transformations de modèles. Une transformation de modèle peut servir à

la simulation, le refactoring mais aussi la génération de code. Une série d'intentions de transformation de modèles avec leurs propriétés est présentée dans [36].

La génération de code dans le MDD est également appelée Transformation de Modèle à Texte (Model to text transformation — M2T). Elle applique des règles de transformation pour extraire des informations des modèles et les représenter sous forme textuelle dans un code généré. La mise en place de la transformation s'effectue en deux temps. Le processus commence par la définition des règles de transformation qui consistent à un mapping entre les éléments du modèle source avec leurs correspondances dans le résultat à générer. L'étape suivante c'est la génération de code avec un outil qui reçoit en entrée le modèle source avec les règles de transformation et génère le résultat voulu.

Par exemple, **Xtend** [5] est un langage de programmation basé sur Java permettant de générer du texte à partir d'un modèle Ecore. Il utilise la génération de code basée sur les patrons. Les patrons sont définis dans des fonctions avec des parties statiques et des parties dynamiques qui seront remplacées par les informations dans le modèle. Utilisé avec Xtext, il permet de générer instantanément le code à la sauvegarde d'un modèle dans un éditeur Xtext ouvert dans un environnement Eclipse. La Figure 4.3a montre un extrait d'un code Xtend.

2.3.3 Framework du domaine

La plus-value du MDD est de produire différentes applications rapidement à partir d'un même langage grâce à la génération de code. Cette ligne de produit est composée par un code invariable et un code spécifique à la variante de l'application avec les appels au code invariable [45]. Dans ce cas, la génération de code ne produit pas la totalité du code de l'application. C'est uniquement une partie du code qui est générée et qui doit être intégrée dans le code existant. C'est là que le framework du domaine intervient en servant d'interface entre le code généré et code invariable. Le framework du domaine est

généralement défini pour alléger la génération de code. Il s'occupe des tâches permettant d'éviter des répétitions dans le générateur et les modèles, ce qui réduit considérablement la complexité de la génération de code [23].

CHAPITRE 3

CONFIGURATION FLEXIBLE DE RS

Dans ce chapitre, nous présentons une DSL qui offre une grande flexibilité pour configurer des projets de RS. D'abord nous justifions pourquoi le MDD est une solution judicieuse de développement de notre outil.

3.1 MDD pour la configuration de RS

Notre objectif est de permettre aux chercheurs de configurer et installer les projets eux-mêmes, sans avoir nécessairement de connaissance en administration de systèmes informatiques, car ces chercheurs peuvent provenir de divers domaines.

La réalisation d'une RS suit un processus itératif. Il est souvent impossible de planifier correctement le protocole qui sera suivi dès le début du projet. Le protocole a souvent besoin d'être adapté à certains cas spéciaux qui peuvent apparaitre durant la réalisation, comme pendant la sélection des études primaires ou l'extraction des données. Il est également recommandé de faire un pilote de l'étude pour être sûre que tous les chercheurs participants dans l'étude ont la même compréhension du protocole à suivre [47].

Dans les outils existants pour aider dans la réalisation de RS, la configuration est faite via des assistants graphiques organisés en formulaires. Ces assistants sont adéquats pour la paramétrisation, mais n'offrent pas la facilité et la flexibilité requise pour changer la sémantique du projet. Avec les possibilités offertes par le MDD, on peut simplifier le processus de configuration de projet en permettant aux chercheurs de spécifier un modèle de la configuration qui fait abstraction des spécifications communes à tous les projets de RS et des détails liés à l'implémentation et à l'infrastructure utilisée. Pour adapter la configuration, le chercheur n'aura qu'à effectuer un petit changement dans le

modèle, qui pourra avoir un impact significatif sur l'implémentation (base de données, code, fonctionnalités, ...) grâce à la génération automatique de code.

Pour mettre en place cette structure dans notre outil *ReLiS*, nous avons défini une DSL, qui permet de spécifier ou de modifier la configuration des composantes qui sont spécifiques à un projet de RS. Les étapes communes à tous les projets RS sont intégrées dans ReLiS avec quelques niveaux de configurations qui peuvent être définies dans le modèle. Par exemple, la manière dont les conflits sont détectés et résolus pendant la phase de sélection, ou les étapes spécifiques à un projet particulier comme le procédure et formulaire d'extraction des données qui sont définis dans le modèle.

3.2 DSL de configuration

Dans la phase de planification, le projet de RS est configuré à l'aide d'une DSL. Nous définissons ce métamodèle à l'aide du diagramme de classe qui contient les points de personnalisations du processus d'une RS. Ces points de personnalisations correspondent aux éléments du protocole spécifique à un projet de RS particulier (voir Section 2.1.2.1). Dans cette partie nous présentons les composants de ce métamodèle qui est présenté dans la Figure 3.1.

L'élément central de ce métamodèle est l'entité `Project` identifiée par l'identifiant unique `name` qui représente un projet de RS. Un projet est composé de quatre éléments principaux, que nous détaillons dans ce qui suit.

3.2.1 Sélection des études primaires

Cet élément est représenté par l'entité `Screening`. Il permet de configurer comment les études primaires sont assignées aux réviseurs (utilisateurs participants dans la RS) avec les critères d'inclusion et d'exclusion. Il peut aussi définir la procédure à suivre en cas de conflit de décision des réviseurs sur une même étude. Deux types de conflits

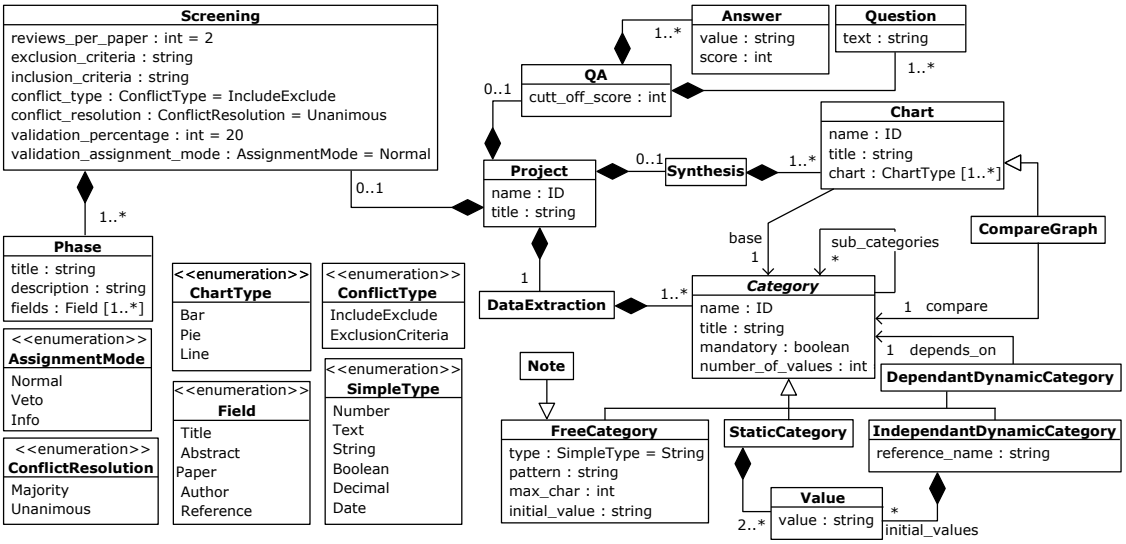


Figure 3.1 – Métamodèle de configuration de projet

sont supportés : si un réviseur exclut une étude alors qu'un autre l'a inclue, ou si tous les réviseurs ont exclus l'étude mais avec des critères d'exclusion différents. Un conflit de décision est résolu soit en ayant une décision unanime entre les réviseurs ou si la majorité converge sur une décision.

Si la validation de la phase de sélection est activée, l'utilisateur peut définir la proportion d'études exclues, choisie de manière aléatoire, qui seront utilisés comme échantillon à la validation. Il peut aussi spécifier comment la décision de la validation va affecter le processus. La validation peut être prise comme les autres décisions de sélection, primer sur les autres décisions (Veto) ou simplement notifier les réviseurs sur les divergences de décisions avec la validation.

Il est également possible de configurer la sélection de manière incrémentale en phases suivant les méta-informations (`Field`) des études qui seront examinées par les réviseurs dans chaque phase.

La phase de sélection est optionnelle dans ReLiS pour permettre de supporter le cas d'utilisation où un chercheur a déjà identifié les études primaires a priori et est intéressé uniquement par l'extraction des données.

3.2.2 Évaluation de la qualité

Le projet peut comporter une évaluation de la qualité, représentée par l'entité `QA` (Quality Assessment). Il faut définir les questions et les réponses possibles (avec le score correspondant) pour construire la liste de contrôle. Un score limite est aussi défini comme score minimal pour les études avec une qualité acceptable.

3.2.3 Extraction des données

L'entité `DataExtraction` définit les catégories des données à collecter dans le formulaire d'extraction pour chaque étude. Les informations dans cette partie vont permettre de générer les formulaires d'extraction des données et la base de données adaptés aux types de valeurs à extraire.

Le métamodèle supporte quatre types de catégories. La `FreeCategory` permet aux utilisateurs d'entrer librement une valeur avec un type spécifique (un entier, du texte, une valeur booléenne, ...). La `StaticCategory` offre la possibilité de choisir une valeur à partir d'une liste prédéfinie de valeurs. Cette liste ne peut pas être modifier pendant la réalisation de la revue. La `IndependantDynamicCategory` est similaire à la `StaticCategory`, mais peut être mis à jour pendant le processus de réalisation de la revue. La `DependantDynamicCategory` est similaire à la catégorie précédente, mais le choix se fait parmi les valeurs entrées dans une autre catégorie.

Chaque catégorie est identifiée par un nom unique. Elle peut avoir une ou plusieurs valeurs et peut être obligatoire pour toutes les études, dont la valeur doit être extraite dans tous les papiers. Une catégorie peut avoir des sous-catégories. Dans ce cas, un formulaire distinct est généré pour les éléments constituants ces sous catégories (ex. Figure 3.2 : ligne 25 – 29).

3.2.4 Synthèse

En dernier lieu, l'entité `Synthesis` spécifie la manière dont les données extraites sont synthétisées et représentées visuellement. Différents types de graphiques sont supportés. Un graphique peut synthétiser les données concernant une catégorie (`Chart`) ou représenter la relation entre les données de différentes catégories (`CompareGraph`).

3.3 Projet de RS dans ReLiS

Le métamodèle que nous venons de présenter permet de définir un modèle de configuration de RS. Pans cette partie, nous illustrons, à l'aide d'un exemple, un modèle de configuration instance du métamodèle présenté dans la Figure 3.1. Le modèle est dans un éditeur intégré dans ReLiS avec sa syntaxe textuelle qui suit la grammaire dans l'Annexe A.

Nous avons choisi d'utiliser une syntaxe textuelle parce que la configuration suit une disposition linéaire (suivant chaque phase) et aussi pour faciliter la manipulation du modèle (copier, coller). L'éditeur dispose d'une coloration syntaxique, et permet l'autocomplétion des termes selon le contexte où est situé le curseur. Il détecte également les erreurs syntaxiques, ce qui facilite la rédaction du modèle de configuration.

Le modèle que nous présentons est un exemple de RS pour *les transformations de modèles*. Il est représenté sur la Figure 3.2. Le modèle débute par la présentation de l'identifiant et du nom du projet. Dans cette RS, la sélection des études primaires (`SCREENING` : ligne 3–10) est réalisée en deux phases (lignes 8– 10). Dans la première phase les études sont évaluées sur base de leur titre, suivi d'une seconde phase où l'évaluation est faite suivant le titre, l'abstract, et le texte complet (disponible via un lien vers l'article). Pour chaque étude, deux réviseurs sont affectés pour la sélection et une validation de 20% des études exclues va être faite.

Cette RS dispose une évaluation de la qualité (`QA` : ligne 12–15) avec deux questions et trois réponses possibles pour chaque question par étude. Si la somme des scores des réponses pour une étude est inférieure à 5, cette étude primaire sera marquée de faible qualité.

La section suivante (`DATA EXTRACTION` : ligne 17–30) configure le formulaire d'extraction des données. La ligne 18 définit à un champ textuel pour entrer le nom de la transformation avec 100 caractères au maximum. Ce champ est obligatoire ('*') et ne peut comprendre qu'une seule valeur ('[1]'). Le champ du langage de transformation à la ligne 19 est une `IndependantDynamicCategory` comprenant une liste de quatre langages prédéfinis parmi lesquels les réviseurs peuvent choisir. La ligne 20 génère une liste de sélection (`StaticCategory`) pour choisir la portée de la transformation qui doit être une des trois choix proposés. Le champ pour des relations d'intentions contient trois sous-catégories (entre les '{...}') (ligne 25–29) et peut avoir plusieurs valeurs ('[0]'). À la ligne 26, une sous-catégorie de type `DependantDynamicCategory` doit prendre une valeur parmi celles qui auront été entrées dans le champ intention (ligne 22). Pour faciliter la création du modèle, ReLiS offre aussi des catégories prédéfinies qui peuvent être réutilisées. C'est le cas de note (ligne 30) qui permet d'ajouter un commentaire libre.

La dernière section concerne la synthèse des données (`SYNTHESIS` : 34–36). Le premier élément va synthétiser les données extraites pour le champ portée dans un diagramme à camembert (pie chart) et un diagramme à barres (bar chart). Le second élément quant à lui va synthétiser dans un graphique (line chart) les données extraites pour le champ portée en ordonnée par année en abscisse.

Figure 3.2 – Un modèle de configuration de projet de ReLiS dans sa syntaxe textuelle

3.4 Processus de RS dans ReLiS

Le modèle de configuration influence les points de variation du processus de RS (Section 2.1.2) dans ReLiS. Dans cette section, nous présentons ce processus personnalisé par le modèle. La Figure 3.3 illustre un processus typique d'utilisation de ReLiS en suivant les méthodologies de réalisation de RS. Dans cette figure nous faisons la distinction entre les actions qui sont exécutées par l'utilisateur et les traitements réalisés automatiquement dans l'outil.

3.4.1 Phase de planification

L'utilisateur commence par planifier la RS et crée un projet dans ReLiS. Pour le faire il définit un modèle de configuration (Section 3.3), qui aboutit à l'installation automatique du projet sur le cloud : une application sur le serveur web avec une base de données propre au projet. Le modèle joue le rôle du protocole de la revue et personnalise les étapes de sa réalisation. Comme une RS est une activité collaborative, il faut ajouter les utilisateurs qui vont participer dans l'étude. Différents rôles peuvent être attribués, tels que des réviseurs, des validateurs ou des gestionnaires de projet, ce qui leur confère des droits d'accès correspondants. Les réviseurs ont les droits leur permettant d'effectuer les activités du processus comme participer dans la sélection d'études primaires,

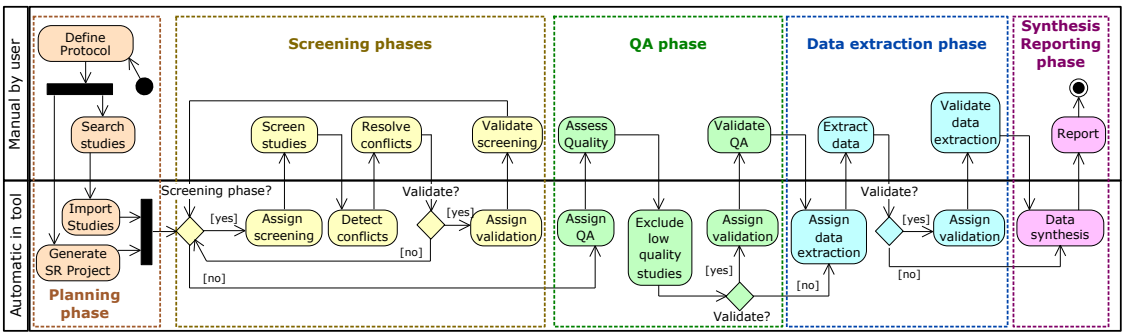


Figure 3.3 – Processus de RS dans ReLiS

l'évaluation de la qualité et l'extraction des données. Les validateurs, en plus des droits des réviseurs, ils ont la possibilité d'effectuer la validation des résultats de chaque phase du processus. Les gestionnaires de projet en plus des droits des deux premières catégories, ils ont les droits de gestion du projet, comme changer la configuration du projet, ajouter des utilisateurs dans le projet, importer les études primaires, assigner les études aux réviseurs et validateurs, ouvrir ou clôturer les phases du processus ainsi que publier le projet après sa réalisation.

Après la mise en place du projet, la recherche d'études primaires commence. Cette tâche est faite en dehors de ReLiS en interrogeant directement les librairies électroniques. Après un filtrage des résultats de recherche suivant les objectifs de la revue, les études primaires identifiées sont importées en format BibTeX, EndNote ou CSV. ReLiS stocke les méta-informations de chaque article (titre, abstract, auteurs, venue de publication, année,...) et un lien vers l'article intégral. Chaque étude importée est étiquetée par sa source et la stratégie de recherche utilisée.

3.4.2 Phase de sélection

Dès que les études sont dans ReLiS, les réviseurs peuvent commencer à trier le corpus et décider quelle étude inclure ou exclure (Figure 3.4). Chaque étude peut être attribuée automatiquement et de manière aléatoire à un nombre fixe de réviseurs et ces derniers peuvent effectuer la sélection. ReLiS détecte automatiquement les conflits de décision entre les réviseurs sur un même document, afin qu'ils puissent les résoudre en suivant la stratégie définie dans le protocole. ReLiS permet un processus avec plusieurs phases de sélection qui sont faites suivant des méta-informations spécifiques (Section3.2.1).

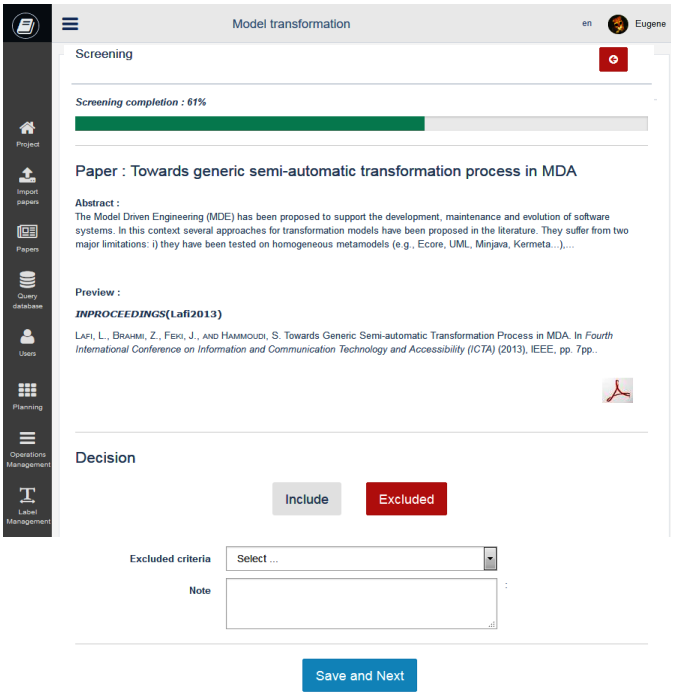


Figure 3.4 – Formulaire de sélection

3.4.3 Phase d'évaluation de la qualité

Suivant les questions et les réponses définies dans le modèle de configuration pour l'évaluation de la qualité, ReLiS génère des formulaires de contrôle de qualité des études incluses. Les études sont attribuées aux réviseurs pour l'évaluation. Ils répondent aux questions sur les formulaires pour chaque étude. ReLiS calcule automatiquement le score de chaque étude et signale les études de faible qualité, candidates pour l'exclusion.

3.4.4 Phase d'extraction et synthèse des données

Dès qu'une étude est incluse, elle peut être aussitôt classifiée. ReLiS génère automatiquement les formulaires d'extraction des données (Figure 3.5) à partir du modèle de configuration. Les réviseurs lisent chaque étude retenue et remplissent le formulaire en ligne pour classer ou extraire les informations pertinentes. Comme pour la sélection, les études peuvent être automatiquement attribuées aux réviseurs pour l'extraction des

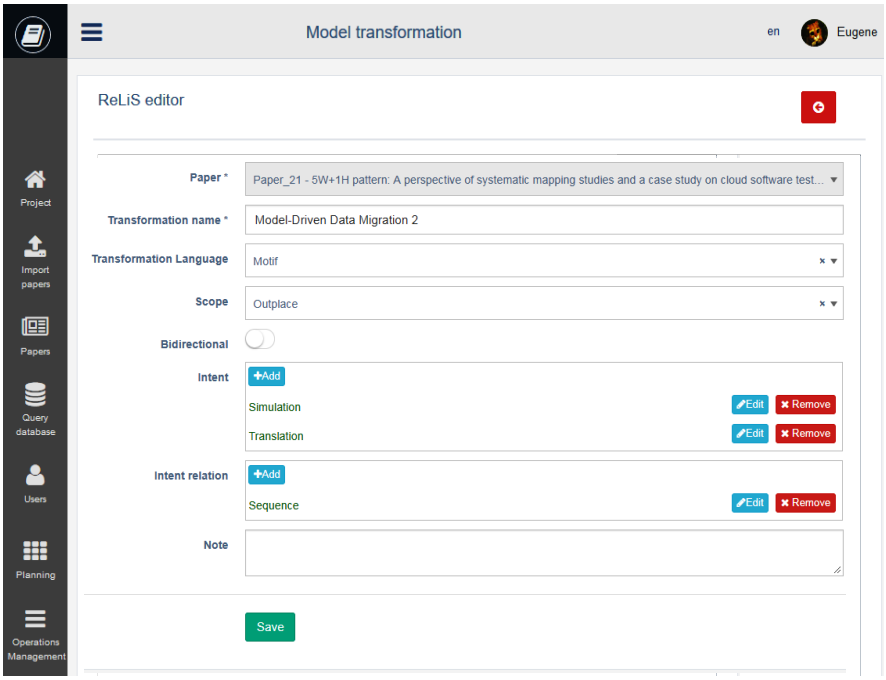


Figure 3.5 – Formulaire d'extraction des données généré à partir de la Figure 3.2

données.

À partir des données extraites, les résultats sont synthétisés automatiquement et peuvent être visualisés sous forme graphique et dans des tableaux (Figure 3.6). Toutes les données extraites peuvent aussi être exportées dans des fichiers CSV prêt à être traités par des outils statistiques plus avancés (SPSS, Mplus, R, Excel). ReLiS peut également générer un rapport partiel du protocole avec toutes les étapes réalisées au cours de la revue.

3.4.5 Synthèse

Si nécessaire, ReLiS permet aux validateurs d'effectuer une validation des résultats de la sélection, de l'évaluation de la qualité ou de l'extraction des données. Il suit la progression et rapporte les statistiques pour chaque phase sous forme de tableaux ou de graphes (Figure 3.7) pouvant être exportés pour une analyse plus approfondie et utilisés

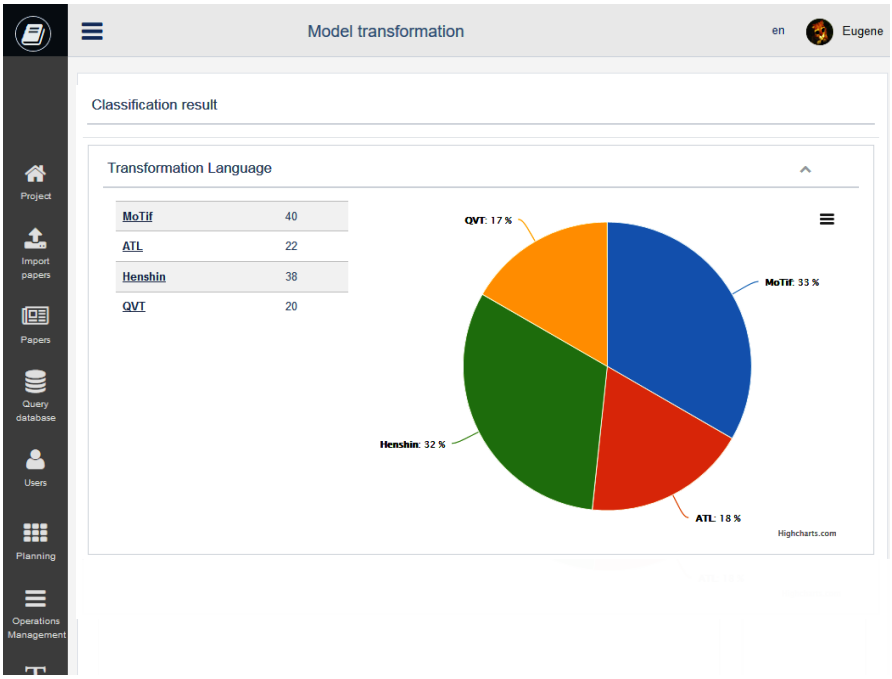


Figure 3.6 – Visualisation de données synthétise

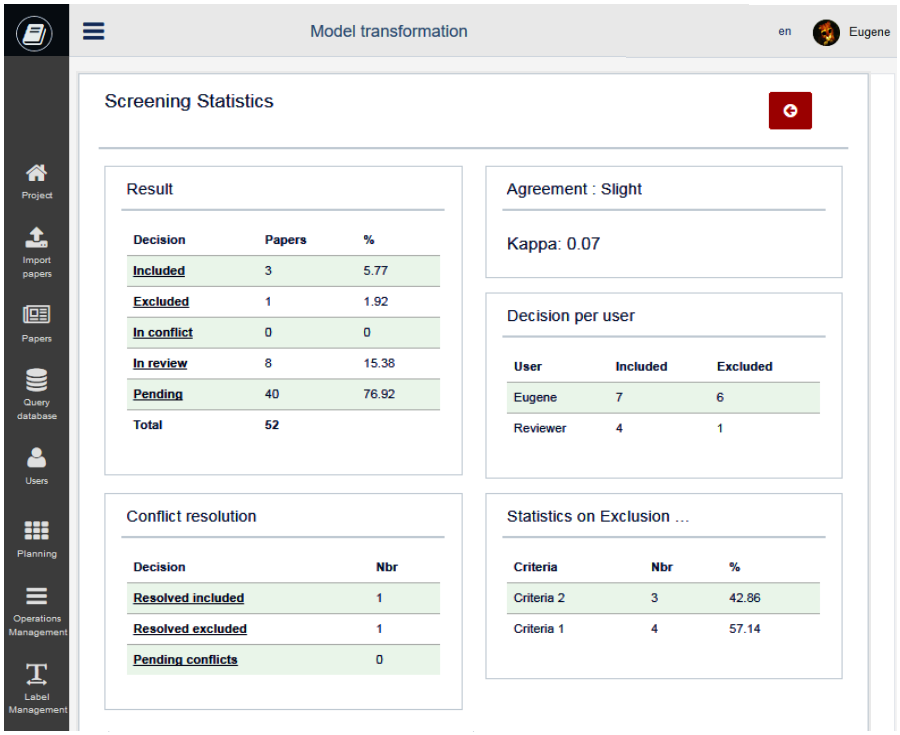


Figure 3.7 – Statistiques sur la sélection d'études

dans la rédaction de rapport de revue.

ReLiS suit un processus itératif, de nouvelles études peuvent être importées à tout moment, les gestionnaires de projet peuvent clôturer ou réactiver une phase terminée et ont accès à la mise à jour la configuration du projet (par exemple, modifier les critères de sélection ou le schéma de classification).

CHAPITRE 4

ARCHITECTURE DYNAMIQUE DE RELIS

La DSL et le processus décrits dans le Chapitre 3 démontre les fonctionnalités de ReLiS quant à la réalisation des RS. Dans l'optique de fournir un outil flexible et collaboratif, nous présentons, dans ce Chapitre, l'architecture de l'infrastructure adoptée. Celle-ci permet de mettre en place une configuration adaptée en tenant compte des exigences du processus de réalisation de RS.

4.1 Exigences

À part supporter les activités liées au processus de réalisation d'une RS, l'outil doit permettre la collaboration, gérer le partage de ressources entre différents projets et supporter le caractère itératif du processus relié aux fonctionnalités de la gestion de projets RS (voir Section 2.1.2). Pour satisfaire ces exigences, l'architecture à adopter doit permettre à l'outil :

- d'être accessible aux utilisateurs pouvant être dans des zones géographiques différentes ;
- d'être configurable et très flexible pour supporter les spécificités de chaque RS ;
- d'être extensible pour permettre d'ajouter des projets ayant des exigences en termes de fonctionnalités ou ressources qui leur sont propres ;
- de gérer le partage de ressources entre différents projets.

La solution qui se présente à nous est l'utilisation des technologies web avec le patron Modèle-Vue-Contrôleur (MVC) généralement utilisé dans les applications web pour séparer l'interface utilisateur du backend [33]. Une application très rependue du patron MVC dans un contexte web permettant l'extension de fonctionnalités se retrouve dans

les systèmes de gestion de contenu (Content Management System — CMS) couramment utilisés pour présenter et gérer du contenu sur le web, comme les outils Joomla [1], WordPress [2] et Drupal [3]. Dans ce cas d'application, l'extension de fonctionnalités du CMS est faite en développant des fichiers d'installations appelés *extensions* qui sont ajoutés dans l'application centrale. Toutefois, la manière dont le MVC est implémenté dans les CMS présente un inconvénient lié au code redondant dans chaque extension, ce qui demande plus d'effort dans le développement manuel de ces extensions et leur intégration dans l'application centrale [44]. Également, à chaque fois que les utilisateurs veulent modifier des fonctionnalités autres que celles faisables à travers une interface utilisateur, ils sont obligés de développer et de réinstaller l'extension.

Afin d'éviter cette redondance et promouvoir la réutilisation, nous avons implémenté le MVC plus *dynamique*, muni d'une architecture partagée et évolutive illustrée dans la Figure 4.1.

4.2 MVC dynamique

Le MVC dynamique reprend l'architecture du MVC en y ajoutant explicitement le concept d'entité. Les entités représentent les instances qui sont gérées par l'application (dans notre cas : études primaires, auteurs, utilisateurs,...). Dans le MVC, l'application est structurée en termes de *Modèles, Vues et Contrôleurs* [33].

Les Modèles traitent l'interaction avec la base de données, ils contiennent les fonctions pour communiquer avec celle-ci, comme l'insertion d'enregistrements ou la sélection d'enregistrements spécifiques de la base de données.

Les Vues s'occupent de l'affichage des données et contrôlent les interactions sur l'interface avec les utilisateurs.

1. `https://www.joomla.org`
2. `https://wordpress.org`
3. `https://www.drupal.org`

Les Contrôleurs contiennent la logique des opérations qui sont effectuées dans l'application. Ils reçoivent les demandes des utilisateurs via des requêtes http et décident des traitements, modèles et vues qui vont intervenir suivant la demande reçue.

Dans notre architecture (Figure 4.1), pour assurer la réutilisabilité et la maintenabilité dans l'application, un type de contrôleur appelé *gestionnaire d'entité* est dédié à la gestion des opérations liées aux entités. Il y a un gestionnaire d'entité pour chaque type d'opération, comme pour : ajouter, modifier, supprimer, lister ou afficher le détail d'une entité. Par exemple, le même contrôleur qui est utilisé pour afficher les méta-données d'une étude primaire est aussi utilisé pour afficher le détail d'un utilisateur ou d'un auteur.

Pour mettre en place cette logique, chaque entité dispose d'un fichier de configuration qui encode ses spécificités : ses attributs et les opérations qui sont réalisées sur l'entité. C'est ce fichier de configuration qui est consulté par un contrôleur gestionnaire d'entité pour avoir la sémantique d'une opération (ex : ajout, modification, détail,...) suivant le type d'entité.

Ce qui fait que cette architecture soit dynamique, c'est que la sémantique des opérations est encodée dans les fichiers de configuration et les contrôleurs sont des classes génériques indépendantes des entités qu'ils vont manipuler. C'est une configuration très importante pour notre cas, car elle nous permet de changer le comportement de l'application simplement en manipulant les fichiers de configuration. C'est une structure qui facilite l'intégration de nouvelles fonctionnalités et la gestion de nouveaux composants sans avoir à modifier le code existant. De plus, l'utilisation d'un langage interprété comme *PHP* nous permet d'ajouter de nouveaux composants sans être obligé de recompiler à chaque ajout.

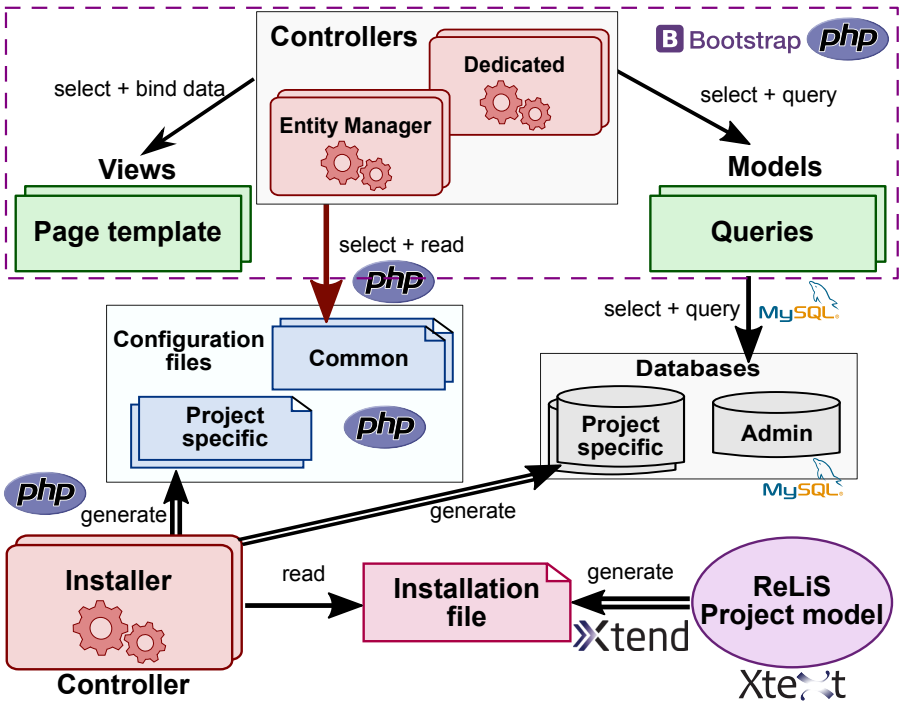


Figure 4.1 – Architecture du MVC dynamique

4.2.1 Structure des entités

Comme la sémantique des opérations réalisées sur les entités est encodée dans les fichiers de configuration d'entité, ces derniers doivent contenir toute l'information que les contrôleurs (gestionnaires d'entité) ont besoin pour exécuter leurs opérations. L'outil étant une application web, la structure de ces fichiers tient compte de l'architecture à trois niveaux utilisée dans la modélisation d'applications web, soit *contenu, hypertexte, présentation* [46]. À ne pas confondre avec le MVC, cette architecture est au niveau de l'information qui est manipulée par l'application, tandis que le MVC concerne l'implémentation et organise les différents composant suivant le rôle qu'ils jouent dans l'application.

4.2.1.1 Organisation de l'information des entités

Le niveau contenu représente la structure de l'information manipulée dans l'application. Cette structure reste inchangée même si l'information peut être présentée différemment à différents endroits de l'application. Il est comparable au schéma de la base de données.

C'est par exemple pour l'entité *auteur* ses caractéristiques, représentant les métadonnées qui seront utilisées pour construire le schéma dans la base de données de la table des *auteurs* (Ex. nom, prénom, affiliation, ...).

Le niveau de l'hypertexte spécifique aux applications web, structure le contenu des entités dans les différentes opérations réalisées. Il spécifie pour chaque opération les éléments du contenu qui sont concernés, ainsi que la logique de navigation entre les différentes opérations.

Par exemple, pour l'opération d'affichage de la liste des auteurs, la spécification des caractéristiques de l'auteur qui sont affichées dans la liste, ainsi que les liens de navigation émanant de cette page. Ces liens peuvent afficher le détail des caractéristiques d'un auteur ou la liste des études réalisées par l'auteur sélectionné.

Le niveau de présentation s'occupe de l'interface graphique de l'application en spécifiant la manière dont les différents éléments sont affichés.

Cette séparation en trois niveaux réduit la complexité du système et permet une réutilisation des composants [46]. Par exemple, nous pouvons spécifier un certain nombre structures hypertextes spécifique aux exigences des différents groupes d'utilisateurs pour un contenu donné. Nous pouvons également spécifier différents types d'affichages (niveau présentation) suivant les préférences d'un utilisateur, tout en gardant un même contenu.

4.2.1.2 Fichier de configuration d'entité

La Figure 4.2 représente la structure d'un fichier de configuration d'entité à l'aide d'un diagramme de classe. Le fichier comprend les caractéristiques de l'entité qu'il représente `(name, title)` ainsi que le nom de la table dans la base de données associé à l'entité.

Le niveau *contenu* de l'entité est représenté par les attributs `(Attribute)`. Chaque attribut a un identifiant, un nom ainsi que le type et la taille maximale que son contenu peut avoir. Cette information aide pour gérer la structure au niveau de la base de données.

Le niveau *hypertexte* est représenté par les opérations `(Operation)` qui sont appliquées à l'entité. Une entité peut avoir plusieurs opérations avec des caractéristiques différentes suivant le type d'opération.

Il y a deux catégories d'opérations. La première est pour les opérations de type `Display` qui dépendent de l'interface graphique, tel que l'énumération de la liste d'entités. Dans ce cas, le fichier de configuration doit mentionner la vue qui est utilisée pour l'interface `(Template)`, ainsi que les liens vers d'autres opérations. L'autre catégorie est pour les opérations qui se passent en arrière-plan, comme la suppression d'un élément `(Remove)`. Pour cette opération, il faut spécifier un lien de redirection après l'exécution de l'opération `(redirect_link)`. Certaines opérations `(Add, Edit)` nécessitent un formulaire. Elles spécifient les champs du formulaire en indiquant lesquels sont obligatoires ou pas, l'état de chaque champ (caché, désactivé ou activé) et l'attribut de l'entité correspondant. Pour les opérations d'affichage `(List, Detail)`, il faut aussi spécifier les attributs affichés `(displayed_fields)`. Pour l'opération (List) il faut également spécifier s'il y a des liens sur les éléments de la liste `(list_links)`. C'est par exemple sur la liste des auteurs, les liens qui mènent soit à la liste de ses publications ou à l'opération pour modifier ses caractéristique (nom, affiliation, etc.).

L'architecture flexible facilitant la configuration de ReLiS est réalisée grâce au MVC

dynamique. Nous avons implémenté cette structure dans les fichiers de configuration d'entités en PHP. Cependant, ils restent indépendants du reste de l'environnement car ils représentent uniquement l'information, sans tenir compte des détails d'implémentation. En terme MDE, les fichiers de configuration sont *indépendants de la plateforme*, donc réutilisables dans d'autres implémentations que celle que nous avons choisie pour ReLiS.

4.3 Architecture partagée et évolutive

Dans cette partie, nous discutons comment un outil RS peut être évolutif quant à l'ajout et la modification de projets sur demande.

4.3.1 Architecture partagée

Pour satisfaire les exigences de la Section 4.1, un outil RS doit supporter plusieurs projets utilisés en parallèle. L'architecture doit donc permettre le partage des ressources entre les différents projets. Ce qui permet une utilisation maximale des ressources et facilite la maintenance de l'application. Chaque projet étant indépendant des autres, il faut assurer leur séparation pour garantir l'intégrité de chaque projet. Cette séparation doit être faite au niveau des données (dans la base de données) et au niveau des fonctionnalités.

Afin d'éviter des conflits entre les données des différents projets et faire en sorte qu'un projet ne puisse pas affecter la performance des autres, chaque projet dispose d'une base de données qui lui est spécifique (voir Figure 4.1). Cette base des données contient des informations qui sont spécifiques à la RS : les études primaires, le processus de sélection, la classification, etc. Il y a aussi une base de données commune (`Admin`), qui contient des informations partagées par tous les projets : les utilisateurs et droits d'accès, les logs et la gestion des projets.

La séparation des fonctionnalités entre les projets est réalisée au niveau des fichiers

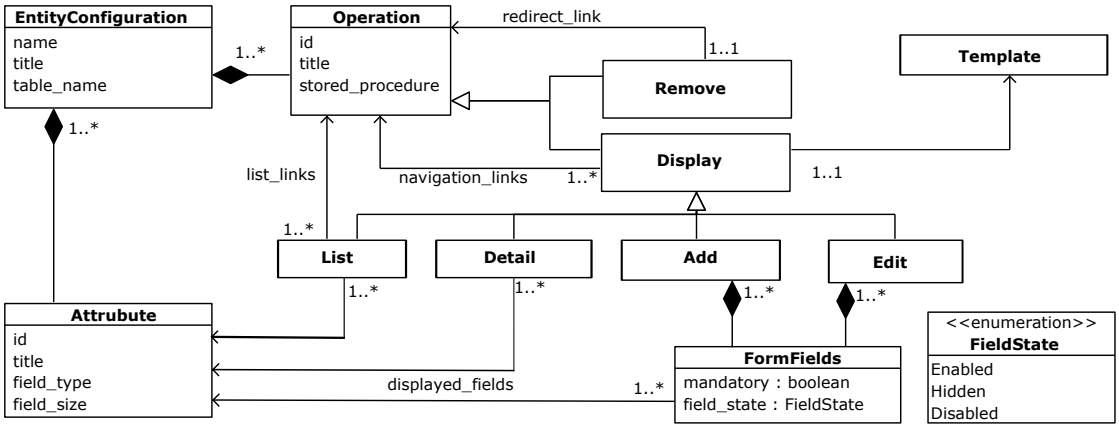


Figure 4.2 – Extrait de la structure de la configuration d'entité

de configuration d'entité. Certaines entités sont gérées de la même manière dans tous les projets. C'est par exemple la manière dont les études primaires et les utilisateurs sont configurés. Ce sont des entités communes à tous les projets et leurs fichiers de configuration sont prédéfinis. Par contre, d'autres entités sont spécifiques à chaque projet. C'est, par exemple, le cas de la classification et de la sélection, qui sont propres au protocole de la RS. Ce sont ces entités spécifiques aux projets et leurs fichiers de configuration sont ajoutés dans ReLiS lors de l'ajout d'un nouveau projet. Les fichiers de configuration et la base de données communs à tous les projets sont quant à eux présents dès l'installation de ReLiS sur le serveur.

4.3.2 Génération de la configuration

Chaque RS a ses spécificités qui sont décrites dans le protocole de la revue (Section 2.1.2.1). Dans ReLiS, les éléments qui sont différents d'un projet à un autre et qui y sont spécifique sont encodés dans un modèle conforme à la DSL présentée dans la Section 3.2. Ce modèle de configuration est défini par un utilisateur dans un éditeur intégré dans ReLiS (voir Section 3.3).

Nous avons implémenté la DSL avec une syntaxe concrète textuelle en utilisant

Xtext. L'éditeur de modèle est intégré dans ReLiS grâce à DSLFORGE [34] qui permet de générer une version web de l'éditeur de texte à partir d'un modèle Ecore et une grammaire Xtext. La grammaire de la DSL de configuration de projets RS dans ReLiS est présentée dans l'Annexe A.

Nous avons couplé à cet éditeur un générateur de code pour produire automatiquement un fichier d'installation à partir de n'importe quel modèle de configuration (voir le bas de la Figure 4.1). Nous avons implémenté la génération du fichier d'installation PHP à l'aide de Xtend. Le choix de Xtend est motivé par le fait que c'est un outil qui peut lire les modèles Ecore et qui offre le meilleur compromis entre son niveau d'expressivité et sa performance [37]. La Figure 4.3a montre un extrait du code permettant de générer la représentation d'un élément du formulaire d'extraction des données de type `FreeCategory`, sous forme d'un tableau (*array*). Sur la Figure 4.3b nous avons le résultat de la transformation dans le fichier d'installation PHP correspondant aux éléments des lignes 18 et 21 du modèle de configuration dans la Figure 3.2.

Après la génération du fichier d'installation, un contrôleur spécial, appelé `Installer`, se sert de ce fichier pour créer les fichiers de configuration d'entité et la base de données spécifique au projet tels que prescrit dans le modèle. Cette création de la base de données s'accompagne par l'ajout de procédures stockées nécessaires pour interagir avec la base de données.

L'intégration de tous les composants dans une même application permet à l'utilisateur de faire ses modifications au niveau du modèle et le reste est traité automatiquement : génération de code et intégration dans les projets existants. Cela réduit sa charge de travail et les connaissances en informatique nécessaire pour mettre en place un projet de RS dans ReLiS.

```
189
190 def generateFreeCategory(FreeCategory cat, String parentField) {
191     var String field_type=cat?.type?.toString ?: 'string' ;
192     var String field_size = cat?.max_char.toString ?: '20' ;
193         '''
194     $config['« parentField »']['fields'][ '« cat.name »']=array(
195     'category_type'=>'FreeCategory',
196     'field_title'=>'« cat?.title ?: cat.name »',
197     « IF (field_type.equalsIgnoreCase('bool') )»
198     'field_type'=>'bool',
199     'field_size'=>1,
200     'field_value'=>'0_1',
201     'input_type'=>'select',
202     'input_select_source'=>'yes_no',
203     «ELSE»
204     'input_type'=>'text',
205     'field_size'=>« field_size »,
206     « IF (field_type.equalsIgnoreCase('string')
207         ||  field_type.equalsIgnoreCase('text')
208         ||  field_type.equalsIgnoreCase('date') )»
209     'field_type'=>'text',
210     « IF (field_type.equalsIgnoreCase('text') )»
211     'input_type'=>'textarea',
212     «ELSEIF (field_type.equalsIgnoreCase('date') )»
213     'input_type'=>'date',
214     «ENDIF»
215     «ELSE»
216     'field_type'=>'«field_type»',
217     «ENDIF»
218     «ENDIF»
219     'number_of_values'=>'«cat.numberOfValues»',
220     « IF (cat.mandatory )»
221     'mandatory'=>' mandatory ',
222     «ENDIF»
223     'pattern'=>'« cat?.pattern ?: "" »',
224     'initial_value'=>'« cat?.initial_value ?: "" »',
225     );
226         '''
227     }
```

(a) Code Xtend pour un élément de type `FreeCategory`

```
104 $config['classification']['fields'][ 'transformation_name']=array(
105         'category_type'=>'FreeCategory',
106         'field_title'=>'Transformation name',
107         'input_type'=>'text',
108         'field_size'=>100,
109         'field_type'=>'text',
110         'number_of_values'=>'1',
111         'mandatory'=>' mandatory ',
112         'pattern'=>'',
113         'initial_value'=>''
114 );


117 $config['classification']['fields'][ 'bidirectional']=array(
118         'category_type'=>'FreeCategory',
119         'field_title'=>'Bidirectional',
120         'field_type'=>'bool',
121         'field_size'=>1,
122         'field_value'=>'0_1',
123         'input_type'=>'select',
124         'input_select_source'=>'yes_no',
125         'number_of_values'=>'1',
126         'pattern'=>'',
127         'initial_value'=>'',
128
129 );
```

(b) Extrait du fichier d'installation généré

Figure 4.3 – Génération de la configuration

4.3.3 Ajout et modification de projets

Comme mentionné dans la Section 2.3.3, lors de la génération de code pour une application, il n'est pas toujours nécessaire de générer la totalité du code de l'application. Le code généré correspond à la variante de l'application et un *framework du domaine* intègre la partie générée dans la partie existante de l'application [23].

4.3.3.1 Installation, mise à jour et archivage de projets

La première étape lors de l'utilisation de ReLiS est l'*installation initiale*. Cette installation ajoute la partie commune de ReLiS comprenant le code source supportant l'architecture MVC avec les fichiers de configuration des entités qui sont communes à tous les projets. Elle déclenche la création de la base de données `Admin`, elle aussi commune pour tous les projets (voir Section 4.3.1).

L'évolution de l'outil fait suite à la gestion des projets de RS : ajout, modification ou clôture de projets. L'ajout d'un projet de RS suit la génération du fichier d'installation à partir du modèle de configuration de la revue à réaliser. Ce fichier est installé dans ReLiS et ajoute les fichiers de configuration d'entité et la base de données spécifiques au projet. La modification d'un projet de RS est aussi faite via le modèle de configuration. Elle conduit aussi à la génération d'un fichier d'installation avec les modifications faites. Ces modifications sont ensuite intégrées et affectent la base de données et les fichiers de configuration des entités spécifiques au projet. Lorsque la RS est complétée, on peut archiver le projet pour permettre une consultation ultérieure. Cette opération désactive les opérations pouvant modifier les données qui sont liées au projet. La base de données spécifique au projet restera donc inchangée pour conserver une image du projet.

4.3.3.2 Intégration des changements

La structure adoptée permet de modifier un projet à partir du modèle de configuration avec une intégration automatique des changements au niveau du code et de la base de données. Le défi est de trouver la manière appropriée d'intégrer ces changements dans le système existant. Après la modification, une solution naïve serait de remplacer l'ancien projet par la nouvelle configuration. Étant donné la nature itérative du processus de RS, les modifications peuvent se faire sur un projet déployé avec une étude déjà en cours. Cette solution serait catastrophique car elle obligerait de recommencer le projet à chaque modification du modèle de configuration. Il est donc primordial de conserver les données déjà existantes dans le système.

Pour bien intégrer ces changements liés à la modification du modèle et éviter la perte ou la corruption des données existantes, il faut identifier les modifications dans le modèle et en tenir compte lors de l'intégration dans le système central. Cela implique la détection de modifications à chaque évolution du modèle de configuration.

4.4 Gestion des modifications de modèle

Dans le développement de logiciels, l'évolution des composants logiciel est omniprésente. Divers composants tels que le code, les données ou la documentation peuvent évoluer. En MDE les modèles sont largement utilisés dans le développement de logiciels. Comme pour les autres composants logiciels, les modèles aussi évoluent ce qui nous donne plusieurs versions d'un modèle durant la vie d'un système basé sur les modèles. Pour comprendre, analyser et gérer ces systèmes, il est crucial d'être capable d'identifier les différences entre les différentes versions de modèle [35]. Après la modification d'un modèle, la comparaison des versions de ce modèle doit donc identifier correctement les différences et trouver une bonne manière de les présenter pour qu'ils soient exploitables.

4.4.1 Modifications dans le modèle de configuration

Pour les modifications du modèle de configuration dans ReLiS, nous cherchons à identifier les changements dans le modèle afin de pouvoir les intégrer correctement dans l'application. Donc, les changements à identifier sont les différences entre les deux dernières versions du modèle de configuration qui ont un impact sur l'application après intégration. Ces différences concernent l'ajout, la suppression ou la modification des composants du modèle à savoir : la partie pour le screening, le QA, l'extraction ou la synthèse des données (voir Figure 3.2). La suppression ne concerne pas l'extraction des données car dans ReLiS chaque projet doit avoir au moins cette phase.

Pour la modification, à part l'extraction des données, les autres composants du modèle de configuration n'affectent pas la structure des données dans l'application. Ils comprennent des valeurs de configuration du processus de la RS. Par contre, les modifications dans la partie d'extraction des données qui peuvent affecter la structure des données dans l'application sont à gérer avec précaution, afin d'éviter la corruption ou la perte de données qui sont déjà dans l'application pendant l'intégration. Les modifications à identifier dans cette partie concernent donc l'ajout d'une nouvelle catégorie, la suppression ou la modification d'une catégorie du modèle initial. La modification d'une catégorie peut avoir plusieurs effets : le changement du nom, de l'un des attributs (label, type, taille, etc.), de l'ordre de la catégorie ou de la relation avec les autres catégories.

Après l'identification des changements, nous avons besoin de choisir une bonne manière de les présenter. Nous avons besoin d'une présentation de ces changements dans un modèle en spécifiant les éléments modifiés et les changements effectués. Cette représentation dans un modèle a l'avantage de permettre leur manipulation automatique grâce aux transformations de modèle. Ces transformations permettent de générer le code qui sera utilisé pour intégrer dans l'application centrale les changements relatifs à aux différences identifiées. Ils peuvent aussi permettre de générer des interfaces pour visualiser

l'évolution du protocole de la revue en suivant les versions du modèle de configuration.

4.4.2 Implémentation préliminaire

Dans l'état actuel de ReLiS, nous ne supportons pas la détection des modifications au niveau des modèles de configuration en comparant les deux dernières versions du modèle. Après la modification du modèle, le fichier généré a la même structure que celui utilisé pour l'ajout d'un projet. C'est au moment de l'intégration que l'installateur va détecter les changements qui ont été effectués par rapport à la configuration antérieure, comme l'ajout, la suppression ou la modification des entités ou des attributs du modèle. Lors de l'intégration, puisque la suppression de certains éléments pourrait corrompre les données existantes, la politique que nous avons adoptée est de permettre la suppression d'un élément seulement s'il ne contient pas encore de données. Sinon, l'élément est désactivé de l'interface utilisateur, mais les données sont préservées dans la base de données. Cela nous permet de pouvoir faire un retour en arrière pour corriger d'éventuelles mauvaises manipulations.

4.4.3 Solution envisagée

La comparaison de modèles comprend l'identification et la présentation des différences entre les modèles comparés. Puisque les modèles sont physiquement sauvegardés dans des fichiers, cette comparaison peut se faire avec des techniques et outils utilisés pour comparer de fichiers textes. Mais, le résultat de cette comparaison ne donne pas d'informations pertinentes sur la différence entre les modèles [35]. Cela est dû au fait que la structure d'un modèle est représentée sous forme d'un graphe de syntaxe abstraite, mais que le modèle est présenté dans sa syntaxe concrète. Les fichiers comparés comprennent la structure du modèle et les informations associées à la syntaxe concrète. Cette comparaison va donc détecter même les changements de la syntaxe concrète alors

qu'ils ne sont pas vraiment liés à la structure des modèles. Également, cette comparaison est faite de façon linéaire, ce qui n'est pas compatible avec la structure du modèle qui suit le graphe de la syntaxe abstraite [41].

Pour identifier les différences, il faut faire la comparaison au bon niveau d'abstraction en prenant en compte la sémantique des modèles. La comparaison est donc faite au niveau des graphes de la syntaxe abstraite des modèles, en traversant en parallèle les graphes des modèles à comparer pour établir la correspondance entre leurs éléments [35]. La comparaison de modèle devient donc similaire au problème d'isomorphisme de graphe qui consiste à évaluer la correspondance entre deux graphes non ordonnés. C'est un problème qui est connu pour être NP-difficile [25].

Pour éviter ce problème, différentes approches sont suivies pour comparer les modèles. Dimitrios et al. [32] analysent un certain nombre d'approches de correspondance de modèles existantes et les évaluent en fonction de leur précision, leur efficacité et leur indépendance à un domaine ou un outil particulier. Ces approches sont :

La correspondance basée sur un identifiant statique affecté à chaque élément du modèle au moment de sa création. C'est avec cette identité que la correspondance est établie entre les éléments des modèles à comparer. L'avantage de cette approche est qu'elle réduit considérablement le temps requis pour établir la correspondance. Par contre, elle fortement liés à l'outil utilisé pour créer les modèles et n'est pas applicable pour comparer des modèles créés séparément.

La correspondance basée sur une signature qui sert d'identifiant pour chaque élément du modèle. La signature est calculée dynamiquement, avec une fonction en utilisant les valeurs des propriétés des éléments du modèle. Par exemple, avec l'outil DSMDiff [35], qui utilise la correspondance par signature, la signature est la concaténation du type, de la catégorie et du nom de l'élément. Contrairement à l'approche précédente, cette approche peut être applicable pour des modèles créés

séparément, car elle ne dépend pas d'un identifiant statique.

La correspondance basée sur les similarités des propriétés des éléments dans les modèles à comparer. La correspondance n'est pas établie seulement si les deux éléments correspondent exactement, mais aussi lorsque les valeurs de leurs propriétés se rapproche. Dans cette approche, un degré d'importance est attribué aux propriétés des éléments des modèles. Ce degré est pris en compte lors du calcul des similarités entre les éléments. Comparé aux deux premières approches, cette approche produit des résultats plus précis, mais nécessite beaucoup d'effort pour configurer les fonctions de calcul des similarités [32]. EMF Compare [10] utilise cette approche dans la comparaison de modèle, mais se sert d'une configuration fixe pour détecter les similarités.

Les algorithmes de correspondance spécifiques aux langages. Cette approche utilise des algorithmes qui sont personnalisés pour des langages spécifiques. L'avantage de cette approche est qu'elle prend en compte la sémantique du langage pour établir la correspondance. Cette connaissance de la sémantique permet d'avoir des résultats très précis et une réduction de l'espace de recherche pour établir la correspondance des modèles. Par exemple, UMLDiff [50] utilise cette approche pour comparer des modèles UML.

Après l'identification des différences, il faut trouver une bonne manière de les représenter. Il y a une représentation dite *opérationnelle*, qui représente les différences dans une série d'opérations à exécuté sur le modèle initial pour obtenir le modèle final. Il y a aussi la représentation *structurelle* qui représente les différences dans un modèle utilisé par les outils de modélisation pour manipuler ou visualiser ces différences [10] . On choisit donc le type de représentation suivant l'objectif de la comparaison des modèles.

Pour ReLiS, une détection des changements grâce aux algorithmes spécifiques au langage avec une représentation structurelle est la plus appropriée. Elle permet d'avoir

des résultats qui sont adaptés à notre domaine et facilite l'intégration des changements dans l'application principale avec une transformation de modèle qui, à partir du modèle des différences génère un fichier d'installation pour l'intégration.

CHAPITRE 5

ÉVALUATION

Dans cette partie nous évaluons ReLiS sur sa capacité à assister les chercheurs durant la réalisation de RS. Avec cette évaluation nous cherchons à répondre aux questions suivantes :

"Comment est-ce que ReLiS améliore le support des activités durant la réalisation d'une RS ?". Pour répondre à cette question nous allons faire une vérification de la couverture des fonctionnalités du processus des RS et une comparaison avec les outils existants.

"Les fonctionnalités dans ReLiS sont-elles complètes et correctement implémentées ?". Pour cette question nous allons faire une validation des fonctionnalités implémentées dans ReLiS en reproduisant des RS systématiques déjà réalisées et publiées. Cette validation est complétée par une évaluation qualitative informelle de l'implémentation et l'architecture adoptées.

5.1 Couverture des fonctionnalités

Comme indiqué dans la Section 2.2, quelques outils existent actuellement pour la réalisation de RS. Certaines études ont aussi identifié et priorisé les fonctionnalités que les outils devraient offrir pour aider dans la réalisation de RS, particulièrement en génie logiciel [1, 39, 40]. Grâce aux données qu'ils ont rendues disponible, nous avons été capable de reproduire une analyse quantitative de ReLiS sur la couverture des fonctionnalités et le comparer aux outils existants.

5.1.1 Méthodologie

Suite à un atelier réunissant des chercheurs expérimentés dans la réalisation de RS, Hassler et al. ont dressé une liste de fonctionnalités prioritaires pour les outils de RS [18]. C'est une liste plus exhaustive que dans [40] : elle considère des fonctionnalités allant au-delà des exigences des chercheurs en génie logiciel et évalue un éventail plus large d'outils qui sont aussi plus récents. Al-Zubidy et al. [1] rapportent l'ensemble final de fonctionnalités avec des scores basés sur leur importance et évaluent six outils supportant la totalité du processus RS : *StArt, Parsifal, SESRA, SLuRp, SLRTOOL et SLR-Tool* (voir Section 2.2.2).

La méthodologie utilisée est une analyse de fonctionnalités basée sur la méthodologie DESMET [30]. Elle évalue l'importance relative des fonctionnalités et la manière dont elles sont implémentées dans chaque outil. Pour chaque outil, une fonctionnalité reçoit la valeur 1 si elle est implémentée complètement, la valeur 0.5 si elle est partiellement implémentée et 0 si elle n'est pas implémentée. Chaque fonctionnalité a un niveau d'importance utilisé pour calculer le score global d'un outil comme une somme pondérée. La liste détaillée des 112 fonctionnalités et les valeurs pour chaque outil évalué sont disponibles en ligne [1]. Les fonctionnalités sont regroupées suivant les exigences de RS qu'elles répondent. Ces groupes sont ordonnés suivant l'importance des fonctionnalités qui les composent. Nous avons effectué une évaluation quantitative de la couverture des fonctionnalités de ReLiS en utilisant les mêmes métriques.

5.1.2 Résultats

Comme indiqué dans [1], le Tableau 5.I montre le score de chaque outil pour les 10 groupes de fonctionnalités plus prioritaires (avec le 11ème car il a le même score que

1. `http://carver.cs.ua.edu/Studies/SLR-Requirements` (Consulté : le 10 Septembre 2017)

le 10ème). Le tableau comprend les résultats de l'évaluation des outils faite dans [1] combinés avec ceux de notre évaluation sur ReLiS. Avec cette évaluation, ReLiS arrive en quatrième position avec un score total de 55%. ReLiS offre un meilleur support pour la collaboration (*Collaboration support*), grâce à sa capacité à assigner des tâches, à permettre la validation, ainsi qu'à détecter et résoudre les conflits. Il a également une bonne couverture des exigences quant à l'évaluation de la qualité (*Quality Assessment*) des études primaires sélectionnées. Bien qu'avec une couverture partielle, il obtient le meilleur score pour automatisation de l'analyse (*Automated Analysis*) concernant la génération de métadonnées, de données statistiques sur les critères de sélection et de graphiques. De même, pour l'encodage des méthodes et des données (*Coding of Methods and Data*), ReLiS prend en charge le codage de la méthodologie (voir Section 3.3), la saisie de données quantitatives et qualitatives et la réalisation d'études pilotes.

La traçabilité, le support d'inclusion / exclusion et la visualisation sont également bien pris en charge dans ReLiS. On explique la faible couverture de la maintenance des données (*Data Maintenance*) par le fait que les fichiers PDF des études primaires ne sont pas stockés dans ReLiS. Par contre, l'outil peut importer les métadonnées des études exportées à partir de n'importe quelle source (y compris manuellement) et les références peuvent être complètement gérées par un service d'analyse de BibTeX intégré (BiBler [2]).

Selon [1], l'intégration de la recherche des études primaires (*Integrated Search*) et la fouiller de texte (*Text Mining*) sont des fonctionnalités très importantes. Cependant, nous avons décidé de ne pas les prendre en charge dans ReLiS comme indiqué la Section 3.4. Il existe plusieurs librairies électroniques avec des API différentes, leurs propres moyens pour récupérer les études qu'ils indexent et différents types de filtres pour exclure automatiquement certaines études. Comme présenté dans [38, 40], il existe de nombreux outils de gestion de référence puissants qui sont déjà largement utilisés pour stocker

2. BiBler - `https://sourceforge.net/projects/bibler`

et gérer des références d'études primaires pour les RS. Dans ReLiS, nous avons choisi de réutiliser ces outils en permettant l'importation de références dans divers formats. ReLiS ne supporte pas la fouille de texte, étant donné que les documents ne sont pas sauvegardés, en raison des droits d'auteur. Dans cette logique, si on enlève ces deux fonctionnalités du Tableau 5.I, ReLiS obtiendrait le score de 76%. Dans ce cas, seul *StArt* le devancerait avec 85%, suivie de *Parsifal* et *SESRA* à 69% .

Une menace à la validité de cette évaluation est le score subjectif qui a été attribué à chaque fonctionnalité rapportée dans [1], ce qui affecte le classement des différentes fonctionnalités. Une alternative consiste à évaluer la couverture de toutes les fonctionnalités sans tenir compte des scores. Dans ce cas, ReLiS se classerait en première position avec un score de 58 sur les 112 fonctionnalités, suivi de *Parsifal* et *StArt* à 45. L'évaluation de la couverture des fonctionnalités pour ReLiS est dans l'Annexe B. Parmi les fonctionnalités non présentées dans le Tableau 5.I, ReLiS est le seul outil qui supporte un processus itératif avec pilotage.

5.2 Validation de la complétude

Pour validation la complétude de ReLiS dans le support de RS, nous avons reproduit des RS déjà réalisées et publiées. Cette validation constitue un bon indicateur, non seulement de l'utilité de l'outil mais aussi de son adaptation aux exigences concrets de RS. Afin prendre en compte les deux types de RS (voir Section 2.1.1), nous reproduisons en premier lieu une SMS intitulée *Systematic Mapping Study of Template-based Code Generation* [48]. En deuxième lieu nous reproduisons une SLR intitulée *Systematic literature reviews in software engineering – A tertiary study* [31]. Nous commençons par décrire la structure de ces études, puis montrer comment se déroulerait leur réalisation en utilisant ReLiS.

Weight	Top ranked high-level features	StArt	Parsifal	SESRA	ReLiS	SLuRP	SLRTOOL	SLR-Tool
10	**Collaboration Support**	100%	100%	100%	**100%**	100%	66%	0%
9	Integrated Search	26%	62%	53%	**9%**	17%	17%	17%
8	Traceability	100%	75%	75%	**75%**	25%	50%	50%
7	Support Text Mining	23%	30%	7%	**0%**	14%	14%	14%
6	Support Inclusion and Exclusion	60%	30%	30%	**40%**	20%	30%	30%
5	**Support Quality Assessment**	100%	50%	50%	**100%**	50%	50%	50%
4	Data Maintenance	100%	100%	100%	**50%**	100%	100%	100%
3	**Automated Analysis**	50%	50%	50%	**67%**	66%	33%	33%
2	Visualization	60%	60%	60%	**67%**	90%	60%	60%
1	**Coding of Methods and Data**	14%	0%	0%	**86%**	14%	14%	0%
1	**Storage of Studies**	66%	66%	66%	**67%**	66%	66%	66%
	Total score	68%	63%	59%	**55%**	48%	44%	32%

Tableau 5.I – Comparaison de la couverture des fonctionnalités

5.2.1 SMS sur la génération de code basée sur les patrons

Dans cette étude, Syriani et al. [48] ont réalisé un SMS sur les publications traitant de la génération de code basée sur des patrons (*Template-based Code Generation*). L'objectif de l'étude est d'identifier les tendances des études traitants de telles générations de code, leurs caractéristiques, les outils utilisés et leur popularité, ainsi que l'importance du MDE dans l'approche. L'étude est réalisée sur une période allant de 2000 à 2016 et suit la méthodologie proposée par Petersen et al. [42].

La stratégie utilisée pour identifier les études candidates est la recherche automatique dans les librairies électroniques : *Engineering Village, Scopus et SpringerLink*.

La sélection des études est réalisée par quatre réviseurs avec un minimum de deux par étude candidate. En cas de conflit de décision, les réviseurs font des séances de concertation pour résoudre les conflits. S'ils n'arrivent pas à un consensus, un autre réviseur sénior est ajouté pour trancher. Après la sélection, au moins 20% des études exclues sont réanalysées par un réviseur sénior pour valider le processus. La sélection est faite dans deux phases. Dans la première phase, les études sont évaluées sur base du titre et de l'abstract. Dans la deuxième phase, la sélection est faite pendant l'extraction des données après une lecture du contenu des études primaires. Les critères d'exclusion

sont : *No code generation*, *Not template-based code generation* et *Not a paper* .

Nous avons reproduit cette étude dans ReLiS. La Figure 5.1 présente le modèle de la configuration. Dans la partie `SCREENING` on remarque que la DSL n'a qu'une seule phase de sélection, malgré que l'article [48] en mentionne deux. Cela est dû au fait que dans ReLiS la phase d'extraction des données inclus automatiquement la possibilité d'exclure d'autres études qui ne sont pas conformes.

L'extraction des données correspond à une classification car c'est un SMS. Les études primaires incluses sont classées suivant un schéma de classification présenté dans [48]. Dans ReLiS, cela figure sous la partie `CLASSIFICATION` du modèle dans la Figure 5.1. Celle-ci permet de générer le formulaire d'extraction des données adapté à l'étude. Le formulaire est représenté dans la Figure 5.2 . Après la classification des études primaires, les données sont synthétisées automatiquement dans des tableaux et graphiques, permettant ainsi de voir les tendances et les regroupements de publications suivant les catégories. La Figure 5.3 montre quelques graphiques synthétisés automatiquement par ReLiS à partir des données cette étude. Nous obtenons les mêmes résultats que ceux trouvés dans [48]. Le projet est disponible en ligne [3]

5.2.2 SLR des SLRs en génie logiciel

Dans cette étude [31], Kitchenham et al. ont réalisé une SLR sur les SLRs qui ont été réalisées en génie logiciel, ce qui en fait une étude tertiaire (voir Section 2.1) . Cette revue fait suite à une autre RS [27] réalisée pour identifier les SLRs en génie logiciel entre Janvier 2004 et Juin 2007. En plus d'étendre la période couverte, cette revue fait une analyse approfondie des études pour identifier, non seulement les tendances, mais aussi les caractéristiques des SLRs. L'étude est réalisée en suivant la méthodologie proposée par Kitchenham et Charters [29].

3. ReLiS – `http://relis.iro.umontreal.ca/projects.php?id=7`

```
SCREENING
Reviews 2
Conflict on Decision resolved_by Unanimity
Criteria=["No code generation","Not template-based code generation","Not a paper"]
Validation 20 % Normal
Phases
    "Phase 1" "Screen per title and abstract" Fields(Title,Abstract)

CLASSIFICATION
List template_style "Template style" * [1] =["Predefined","Output-based","Rule-based"]

Simple comment_template_style "Comment for Template style"   :text(500)

List design_time "Design-time input type" * [1]  =["General purpose","Domain specific"
    ,"Schema","Programming Language"]

Simple comment_design_time "Comment for Design-time input type"   :text(500)

DynamicList tool "Tool" * [1] "Tool" =["Acceleo","Xpand","EGL","JET","MOFScript","Other"
    ,"Programmed","Simulink TLC","StringTemplate","T4","Unspecified","Velocity","Rational"
    ,"XSLT","Fujaba","F   reeMarker","Rhapsody","Xtend"]

Simple comment_tool "Comment for tool"   :text(500)

List run_time "Run-time input type" * [1] =["General purpose","Domain specific","Structured
    data","Source code"]

Simple comment_run_time "Comment for Run-time input type"   :text(500)

DynamicList output_type "output_type" * [1] "Output type" =["Source code","Structured data"
    ,"Natural language"]

Simple comment_output_type "Comment for output_type"   :text(500)


Simple mde "MDE" [1] :bool

DynamicList scale "Application scale" * [1] "Application scale" =["Small scale","Large scale"
    ,"No application"]

DynamicList domain "Application domain" * [1] "Application domain" =["Software engineering"
    ,"Embedded systems","Web technology","Networking","Aspect-oriented","Mobile systems"
    ,"Prog. Lang","Testing","Other","Compiler","Bio-med","Dist. Sys","Simulation ","DB"
    ,"Security","AI","Refactoring","Robotics","Graphic"]

DynamicList orientation "Orientation" * [1] "Orientation" =["Academic","Industry"]

```

Figure 5.1 – Modèle de configuration pour le SMS

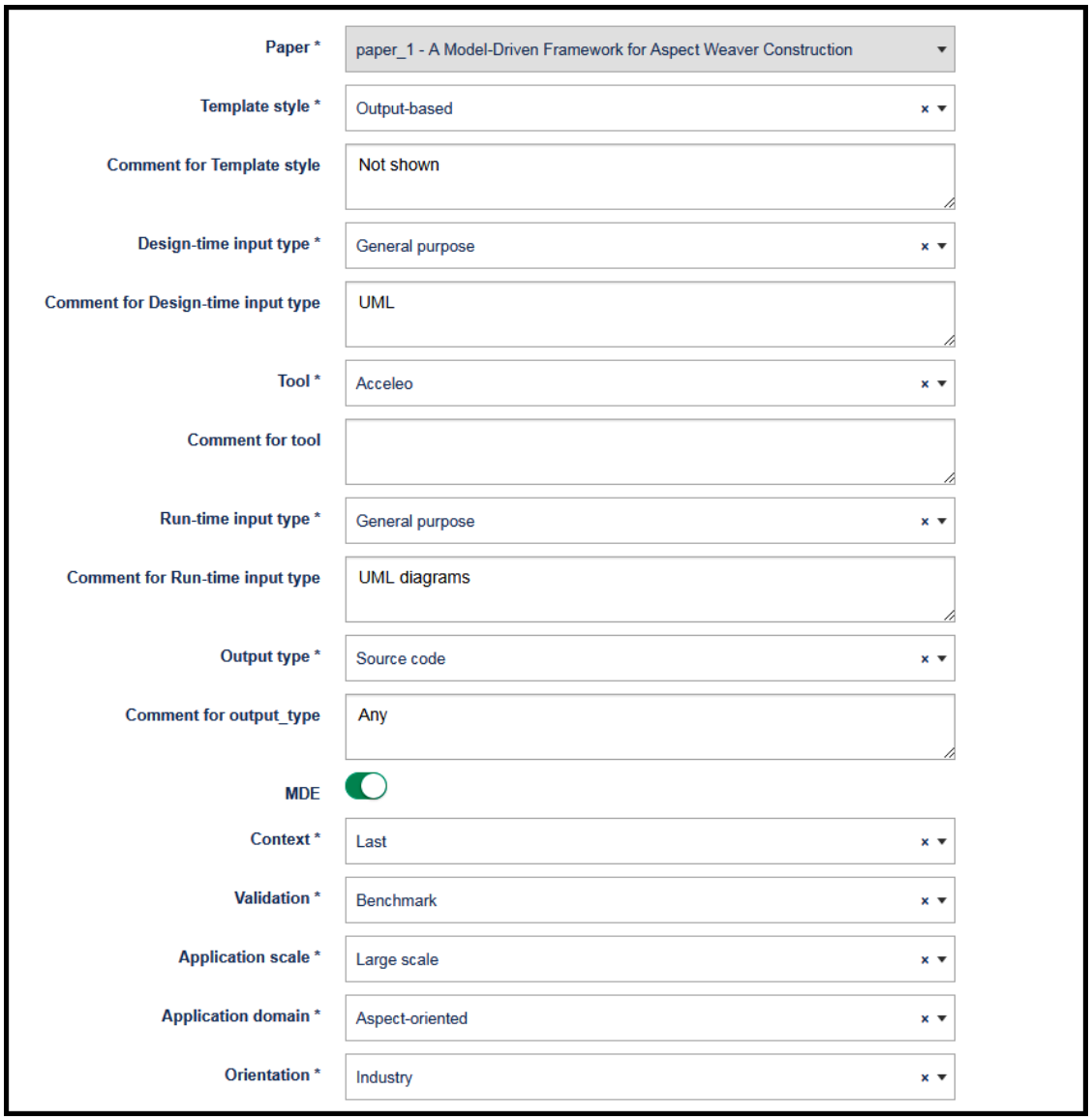
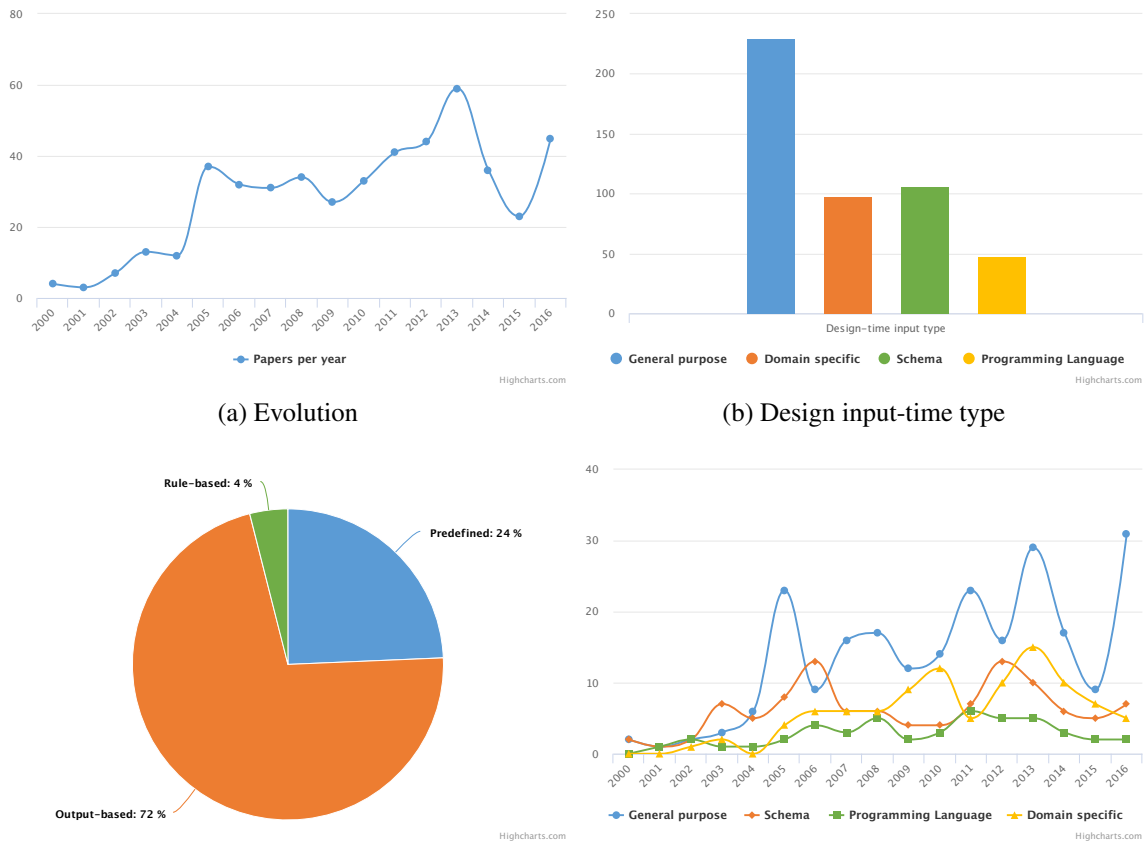


Figure 5.2 – Formulaire généré dans ReLiS pour l'extraction des données du SMS



(a) Evolution

(b) Design input-time type

(c) Template style

(d) Evolution of design input-time type

Figure 5.3 – Graphiques générés par ReLiS montrant les mêmes résultats que le SMS reproduit

Pour l'identification des études candidates ils ont utilisé la recherche automatique dans les librairies électroniques : *IEEE Computer Society Digital Library, ACM, Citeseer, SpringerLink, Web of Science* et *Scopus*.

La sélection des études primaires est réalisée dans trois phases. La première est faite par un seul réviseur sur base du titre, abstract et des mots-clés. Une deuxième phase de sélection est faite par trois réviseurs avec deux pour chaque étude candidate. En cas de conflit, les réviseurs doivent se concerter pour trouver un consensus. La troisième phase requiert aussi deux réviseurs par étude candidate et se base sur le contenu de l'étude à évaluer.

Cette RS contient une étape d'évaluation de la qualité (QA). La liste de contrôle utilisé dans cette phase est composée de quatre questions. À chaque question les réviseurs répondent par *Oui, Partiellement* ou *Non*, et l'étude reçoit respectivement un score de 1, 0.5 ou 0. L'objectif du QA n'étant pas d'exclure des études à faible qualité, aucune étude n'a été exclue dans cette phase. Deux réviseurs évaluent chacune des études primaires et, comme pour la sélection, en cas de conflits, les réviseurs se concertent pour trouver un consensus. Un troisième réviseur est ensuite affecté pour une validation de la totalité des évaluations faites. En cas de conflit avec l'évaluation des deux premiers réviseurs, ils discutent pour trouver un consensus.

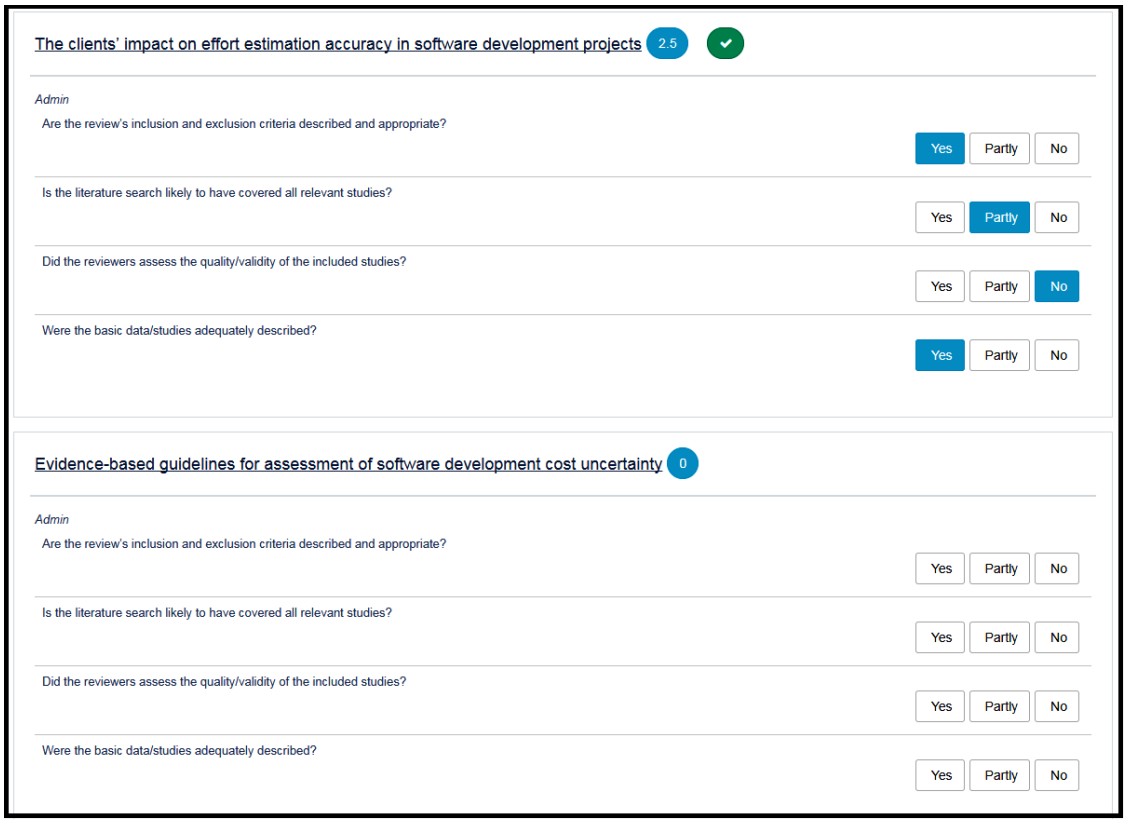
Nous avons reproduit tout ce processus dans ReliS. La Figure 5.4a représente les questions de QA dans le modèle de configuration. Nous avons ensuite affecté des réviseurs pour effectuer l'évaluation en répondant aux questions de la liste de contrôle sur le formulaire généré automatiquement dans ReLiS (Figure 5.4b).

Après l'évaluation, nous avons affecté un autre réviseur pour une phase de validation. Le réviseur évalue l'étude et accepte l'évaluation s'il trouve qu'elle a été bien faite ou la rejette et mentionne en commentaire les raisons du rejet. Dans ce cas, la personne qui a évalué une étude rejetée revoit son évaluation en tenant compte des commentaires de la validation.

L'extraction des données est effectuée suivant le schéma de classification présenté dans l'étude [31]. Elle est faite par deux réviseurs en parallèles avec des séances de concertation pour résoudre d'éventuelles conflits. La reproduction de l'extraction et la synthèse de données dans ReLiS est faite de la même manière que celle présenté dans la Section 5.2.1. Seules les catégories du formulaire d'extraction et les valeurs à affecter changent pour s'adapter au schéma de l'étude.

```
QA
Questions  =[
             "Are the review's inclusion and exclusion criteria described and appropriate?",
             "Is the literature search likely to have covered all relevant studies?",
             "Did the reviewers assess the quality/validity of the included studies?",
             "Were the basic data/studies adequately described?"]
Response   =[
             "Yes":1,
             "Partly":0.5,
             "No":0 ]
```

(a) Modèle de configuration



(b) Formulaire généré

Figure 5.4 – Évaluation de la qualité

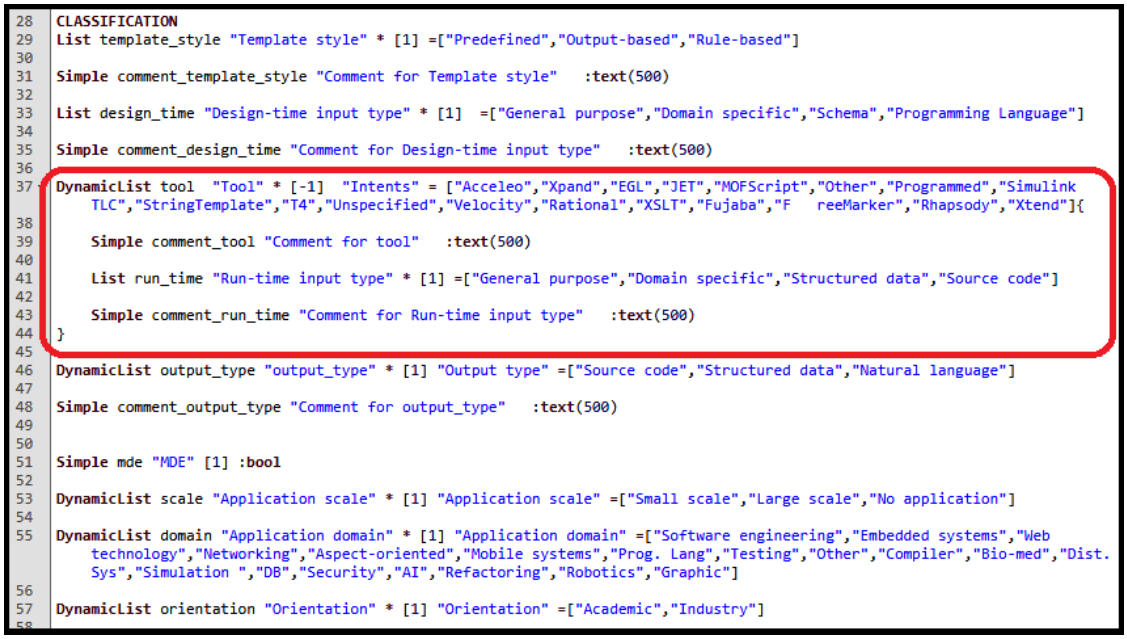


Figure 5.5 – Amélioration du modèle de configuration pour l'extraction des données

5.2.3 Discussion

Dans cette validation, nous avons démontré que ReLiS peut supporter les stratégies qui a été suivie dans les deux RS, ayant obtenu au final les mêmes résultats après la synthèse de données extraites. Cela montre que ReLiS permet de réaliser les différents types de RS et produire les résultats attendus.

Cette activité nous a aussi permis de mettre en évidence les avantages que comporte l'utilisation de ReLiS. Après l'encodage du protocole de la revue dans le modèle de configuration, ReLiS s'adapte à la planification établie (phases de sélections, formulaire d'évaluations de la qualité et d'extractions de données, etc.). Les chercheurs utilisent donc un outil adapté à leur RS, ce qui leur facilite la tâche et les empêchent de s'écarter du protocole planifié durant la réalisation.

Également, durant la réalisation de la revue, certaines activités sont automatisées. C'est le cas de l'assignation aléatoire d'études (pour la sélection, QA ou extraction des données), la détection automatique de conflits et la synthèse des résultats. L'outil sup-

porte aussi la collaboration, ce qui permet à plusieurs réviseurs de travailler sur une même plateforme avec les mêmes données. On peut suivre l'évolution du processus, réassigner les études au besoin et intégrer automatiquement le travail des différents réviseurs. Dans les deux RS originales, les opérations comme l'extraction des données étaient faites dans des classeurs Excel séparés qui, en plus de ne pas être bien adaptés au RS car très générales, demandaient un travail supplémentaire d'intégration des données des différents réviseurs après l'extraction. C'est une tâche qui ajoute la charge de travail et qui peut entrainer l'introduction d'erreurs si l'activité n'est pas faite minutieusement.

L'autre avantage est que, étant donné que les résultats de RS sont généralement publiés (journaux, conférences ou rapports techniques) avec des exigences sur le format et le nombre de pages que doit comporter la publication, les auteurs n'ont pas suffisamment d'espace pour mettre tous les résultats de l'étude. ReLiS constitue donc un espace complémentaire pour compléter le rapport officiel, en offrant la totalité des informations récoltées durant la réalisation de la revue, que ce soit au niveau du processus ou des données extraites.

Dans une discussion avec les auteurs de [48], ils nous ont confirmés que les possibilités offertes par ReLiS dans son formulaire d'extraction leur permettrait d'améliorer le schéma de classification (Figure 5.1) en exploitant surtout la possibilité des champs multi-valeurs et les champs avec des sous-catégories. C'est le cas des catégories *Tool* et *Runtime input type* qui sont normalement liées et pour une même étude peuvent avoir plusieurs valeurs. Nous avons effectué ces améliorations tel que reflété dans la Figure 5.5. Cette observation montre que les possibilités et l'expressivité offertes par la DSL permettent d'améliorer l'extraction et la représentation des données, facilitant ainsi l'analyse.

Cette validation nous a également permis d'identifier certaines activités qui sont réalisées dans les RS, mais qui ne sont pas implémentées de la même manière dans ReLiS.

Par exemple dans [31] l'extraction des données et le QA sont faits par deux réviseurs en parallèle. Puis, ils mettent ensemble les résultats pour comparer et résoudre d'éventuels conflits. Pour ces tâches, ReLiS supporte un seul réviseur pour extraire les données et les autres peuvent être affectés pour la validation avec la possibilité de modifier les données. Ce choix suit Brereton et al. [9] qui recommandent un utilisateur pour l'extraction et un autre pour faire la validation, afin de réduire la charge de travail et le temps requis pour cette activité.

5.3 Conformité et utilité

Nous avons effectué une évaluation qualitative informelle sur l'exactitude de l'implémentation et l'utilité de l'architecture choisie pour ReLiS. Dans le cadre d'un cours de cycle supérieur (IFT6252) sur les méthodes empiriques en génie logiciel à l'Université de Montréal, les étudiants réalisent un SMS sur différents sujets de recherche dans le cadre de leur projet d'étude. Dans la session d'automne 2016, 10 étudiants sans expérience préalable dans la réalisation de RS ont été invités à utiliser ReLiS pour leurs projets de SMS. Cette activité a été réalisée avec la version antérieure de ReLiS qui ne supportait pas encore la totalité du processus de RS. Elle supportait l'importation d'études primaires, l'extraction et la synthèse de données ainsi que l'export des résultats. Cela nous a permis d'avoir des sujets de test pour vérifier si les fonctionnalités de l'outil sont implémentées correctement. Nous avons également pu évaluer si les types de catégories, pour générer les formulaires d'extraction des données, sont suffisants pour couvrir tous les SMS des étudiants. Dans leurs projets, les étudiants ont utilisé tous les types de catégories, mais aucun ne nécessitait l'utilisation de contraintes avancées sur les catégories.

Nous étions aussi intéressées de savoir comment l'installation et la configuration des projets ont été utilisées. Pour cela, nous avons étendu ReLiS pour enregistrer l'histo-

rique des installations de chaque projet, tout en suivant les modifications effectuées dans le modèle de configuration de projet. Nous avons trouvé que toutes les réinstallations visaient à améliorer le schéma de classification. Les étudiants ont constaté que la réinstallation automatique de leurs projets était très utile. Ceci était attendu car, au cours du processus d'apprentissage, ils eu recours à plusieurs itérations avant d'arriver au schéma de classification souhaité. Les étudiants ont apprécié le fait qu'ils étaient en mesure de réinstaller un projet en mettant à jour le modèle de configuration sans perdre l'information sur les documents qu'ils avaient déjà classés. Cependant, nous avons constaté une plus grande quantité de catégories désactivées que prévu. Nous avons découvert que c'était principalement à cause de l'incompréhension de la syntaxe de la DSL.

CHAPITRE 6

CONCLUSION

Nous concluons en résumant les contributions de ce mémoire et une présentation des travaux futurs. Ce mémoire contribue au domaine de génie logiciel basé sur l'évidence en appuyant les revues systématiques.

6.1 Résumé

Le processus de réalisation de RS est composé de plusieurs activités nécessitant beaucoup de travail et de temps. Ce processus est exécuté itérativement ce qui n'est pas bien supporté par les outils de RS existants actuellement. Dans ce mémoire nous avons présenté un nouvel outil, ReLiS, pour aider dans la réalisation de RS. C'est un outil flexible, configurable, qui permet la collaboration et supporte le caractère itératif et collaboratif du processus de réalisation des RS. Il s'appuie sur le MDD pour générer une application personnalisée à partir d'un modèle de la RS.

Pour configurer les projets de RS dans ReLiS, nous avons implémenté une DSL qui permet d'encoder le protocole de la revue dans un modèle de configuration. Les composants de ce modèle illustrent les points de variation du processus de réalisation d'une RS tels que supportés dans ReLiS. Il comprend la configuration de la sélection d'études primaires, le QA, l'extraction et la synthèse des données. Ce modèle est par la suite utilisé pour générer un fichier d'installation d'un projet de RS personnalisé suivant le processus et les exigences de la revue à réaliser.

En vue de satisfaire les exigences des RS, nous avons conçu une architecture de l'infrastructure qui repose sur un MVC dynamique pour permettre à l'application d'être très configurable et adaptable aux variantes des RS. C'est une architecture partagée et évolu-

tive permettant la gestion et l'ajout de nouveaux projets aux caractéristiques et fonctionnalités qui leurs sont spécifiques. L'architecture supporte le partage de ressources entre les différents projets de RS. Elle comprend une partie qui est commune à tous les projets et une partie qui est spécifique à chaque projet de RS.

Après l'implémentation de ReLiS, nous avons également évalué sa couverture des fonctionnalités du processus de réalisation de RS (identifié dans [1]) et une comparaison avec les outils existants. L'évaluation a montré que ReLiS est un des outils qui couvre le plus grand nombre de fonctionnalités prioritaires. Nous avons démontré la complétude de ReLiS en reproduisant plusieurs RS déjà publiées. ReLis est capable de supporter les différentes stratégies suivies lors de la réalisation de RS. ReLiS a été testée par plusieurs utilisateurs et est maintenant disponible au publique.

6.2 Portée future

Notre objectif est de rendre cet outil de RS accessible aux chercheurs n'ayant pas nécessairement de connaissance dans l'administration de systèmes informatiques. Nous prévoyons donc fournir une syntaxe plus intuitive pour la DSL de configuration de ReLiS. Par exemple, certaines parties de la configuration sont plus adaptée à un formulaire (comme la configuration de la sélection des études primaires) et d'autre à une syntaxe textuelle ou un tableau (comme la définition du formulaire d'extraction). Nous explorons les éditeurs projectionnels [4] pour permettre une syntaxe plus adaptée. Cela va s'accompagner d'une évaluation pour attester de la qualité de la syntaxe adoptée. Nous prévoyons aussi améliorer la couverture des fonctionnalités en ajoutant les fonctionnalités qui ne sont pas actuellement couvertes ou qui sont partiellement implémentées dans ReLiS. Pour suivre les changements du protocole de la revue, nous prévoyons implémenter une détection des modifications dans les modèles de configurations pour les appliquer automatiquement à l'application en utilisant des techniques basées sur le MDD.

BIBLIOGRAPHIE

[1] AL-ZUBIDY, A., CARVER, J. C., HALE, D. P. et HASSLER, E. E. (2017). Vision for SLR tooling infrastructure : Prioritizing value-added requirements. *Information and Software Technology*, 91:72–81.

[2] ALDERSON, P., GREEN, S., HIGGINS, J. *et al.* (2004). Cochrane Reviewers' Handbook 4.2. 2. *The Cochrane Library*, 1.

[3] BARN, B., RAIMONDI, F., ATHIAPPAN, L. et CLARK, T. (2014). Slrtool : A Tool To Support Collaborative Systematic Literature Reviews.

[4] BERGER, T., VÖLTER, M., JENSEN, H. P., DANGPRASERT, T. et SIEGMUND, J. (2016). Efficiency of Projectional Editing :A Controlled Experiment. *In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 763–774. ACM.

[5] BETTINI, L. (2013). *Implementing Domain-Specific Languages with Xtext and Xtend*. Numéro 2. Packt Publishing.

[6] BIGENDAKO, B. et SYRIANI, E. (2018). Modeling a tool for conducting systematic reviews iteratively. *In Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development*, pages 552–559.

[7] BOWES, D., HALL, T. et BEECHAM, S. (2012). SLuRp – A Tool to Help Large Complex SystematicLiterature Reviews Deliver Valid and Rigorous Results. *In Proceedings of the 2nd international workshop on Evidential assessment of software technologies*, pages 33–36. ACM.

[8] BRAMBILLA, M., CABOT, J. et WIMMER, M. (2017). Model-Driven Software Engineering in Practice. *Synthesis Lectures on Software Engineering*, 3(1):1–207.

[9] BRERETON, P., KITCHENHAM, B. A., BUDGEN, D., TURNER, M. et KHALIL, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of systems and software*, 80(4):571–583.

[10] BRUN, C. et PIERANTONIO, A. (2008). Model Differences in the Eclipse Modelling Framework. *UPGRADE, The European Journal for the Informatics Professional*, 9(2):29–34.

[11] CARVER, J. C., HASSLER, E., HERNANDES, E. et KRAFT, N. A. (2013). Identifying Barriers to the Systematic Literature Review Process. *In Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on*, pages 203–212. IEEE.

[12] COHEN, J. (1968). Weighted kappa : Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

[13] DYBÅ, T. et DINGSØYR, T. (2008). Empirical studies of agile software development : A systematic review. *Information and software technology*, 50(9):833–859.

[14] DYBA, T., KITCHENHAM, B. A. et JORGENSEN, M. (2005). Evidence-Based Software Engineering for Practitioners. *IEEE Software*, 22(1):58–65.

[15] EYSHOLDT, M. et BEHRENS, H. (2010). Xtext - Implement your Language Faster than the Quick and Dirty way. *In Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, pages 307–309. ACM.

[16] FERNÁNDEZ-SÁEZ, A. M., BOCCO, M. G. et ROMERO, F. P. (2010). SLR-Tool : A Tool for Performing Systematic Literature Reviews. *In ICSOFT (2)*, pages 157–166.

[17] HAREL, D. et RUMPE, B. (2000). Modeling Languages : Syntax Semantics and All That Stuff. Rapport technique, Technical report.

[18] HASSLER, E., CARVER, J. C., HALE, D. et AL-ZUBIDY, A. (2016). Identification of SLR tool needs–results of a community workshop. *Information and Software Technology*, 70:122–129.

[19] HEALTH, N. et (AUSTRALIA), M. R. C. (2000). *How to review the evidence : systematic identification and review of the scientific literature : handbook series on preparing clinical practice guidelines*. National Health and Medical Research Council Canberra.

[20] HERNANDES, E., ZAMBONI, A., FABBRI, S. et THOMMAZO, A. D. (2012). Using GQM and TAM to evaluate StArt - a tool that supports Systematic Review. *CLEI Electronic Journal*, 15(1):3–3.

[21] IMTIAZ, S., BANO, M., IKRAM, N. et NIAZI, M. (2013). A Tertiary Study : Experiences of Conducting Systematic Literature Reviews in Software Engineering. *In Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, pages 177–182. ACM.

[22] JALALI, S. et WOHLIN, C. (2012). Systematic Literature Studies : Database Searches vs. Backward Snowballing. *In Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '12, pages 29–38, New York, NY, USA. ACM.

[23] KELLY, S. et TOLVANEN, J.-P. (2008). *Domain-specific modeling : enabling full code generation*. John Wiley & Sons.

[24] KHAN, K. S., TER RIET, G., GLANVILLE, J., SOWDEN, A. J., KLEIJNEN, J. *et al.* (2001). *Undertaking systematic reviews of research on effectiveness : CRD's gui-*

dance for carrying out or commissioning reviews. Numéro 4 (2n. NHS Centre for Reviews and Dissemination.

[25] KHULLER, S. et RAGHAVACHARI, B. (1996). Graph and Network Algorithms. *ACM Computing Surveys (CSUR)*, 28(1):43–45.

[26] KITCHENHAM, B. (2004). Procedures for Performing Systematic Reviews. Tech. report TR/SE-0401, Keele University.

[27] KITCHENHAM, B., BRERETON, O. P., BUDGEN, D., TURNER, M., BAILEY, J. et LINKMAN, S. (2009). Systematic literature reviews in software engineering–a systematic literature review. *Information and software technology*, 51(1):7–15.

[28] KITCHENHAM, B. et BRERETON, P. (2013). A systematic review of systematic review process research in software engineering. *Information and software technology*, 55(12):2049–2075.

[29] KITCHENHAM, B. et CHARTERS, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. Rapport technique EBSE 2007-001, Keele University and Durham University Joint Report.

[30] KITCHENHAM, B., LINKMAN, S. et LAW, D. (1997). DESMET : a methodology for evaluating software engineering methods and tools. *Computing & Control Engineering Journal*, 8(3):120–126.

[31] KITCHENHAM, B., PRETORIUS, R., BUDGEN, D., BRERETON, O. P., TURNER, M., NIAZI, M. et LINKMAN, S. (2010). Systematic literature reviews in software engineering–a tertiary study. *Information and Software Technology*, 52(8):792–805.

[32] KOLOVOS, D. S., DI RUSCIO, D., PIERANTONIO, A. et PAIGE, R. F. (2009). Different Models for Model Matching : An analysis of approaches to support model differencing. *In Comparison and Versioning of Software Models*, pages 1–6. IEEE.

[33] KRASNER, G. E., POPE, S. T. *et al.* (1988). A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. *Journal of object oriented programming*, 1(3):26–49.

[34] LAJMI, A., MARTINEZ, J. et ZIADI, T. (2014). DSLFORGE : Textual Modeling on the Web. *In Demonstrations at MODELS*, volume 1255. CEUR-WS.org.

[35] LIN, Y., GRAY, J. et JOUAULT, F. (2007). DSMDiff : a differentiation tool for domain-specific models. *European Journal of Information Systems*, 16(4):349–361.

[36] LÚCIO, L., AMRANI, M., DINGEL, J., LAMBERS, L., SALAY, R., SELIM, G. M., SYRIANI, E. et WIMMER, M. (2016). Model transformation intents and their properties. *Software & systems modeling*, 15(3):647–684.

[37] LUHUNU, L. et SYRIANI, E. (2017). Comparison of the Expressiveness and Performance of Template-Based Code Generation Tools. *In Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering*, pages 206–216. ACM.

[38] MARSHALL, C. et BRERETON, P. (2013). Tools to Support Systematic Literature Reviews in Software Engineering : A Mapping Study. *In Empirical Software Engineering and Measurement*, pages 296–299. IEEE.

[39] MARSHALL, C., BRERETON, P. et KITCHENHAM, B. (2014). Tools to Support Systematic Reviews in Software Engineering : A Feature Analysis. *In Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, page 13. ACM.

[40] MARSHALL, C., BRERETON, P. et KITCHENHAM, B. (2015). Tools to Support Systematic Reviews in Software Engineering : A Cross-Domain Survey using Semi-structured Interviews. *In Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*, page 26. ACM.

[41] MEYERS, B. et VANGHELUWE, H. (2011). A framework for evolution of modelling languages. *Science of Computer Programming*, 76(12):1223–1246.

[42] PETERSEN, K., FELDT, R., MUJTABA, S. et MATTSSON, M. (2008). Systematic Mapping Studies in Software Engineering. *In Evaluation and Assessment in Software Engineering*, pages 68–77. British Computer Society.

[43] PETERSEN, K., VAKKALANKA, S. et KUZNIARZ, L. (2015). Guidelines for conducting systematic mapping studies in software engineering : An update. *Information and Software Technology*, 64:1–18.

[44] PRIEFER, D., KNEISEL, P. et TAENTZER, G. (2016). JooMDD : A Model-driven Development Environment for Web Content Management System Extensions. *In International Conference on Software Engineering Companion*, pages 633–636. ACM.

[45] ROYER, J.-C. et ARBOLEDA, H. (2013). *Model-Driven and Software Product Line Engineering*. John Wiley & Sons.

[46] SCHWINGER, W. et KOCH, N. (2006). Modeling web applications. *Web Engineering*, pages 39–64.

[47] STAPLES, M. et NIAZI, M. (2007). Experiences using systematic review guidelines. *Journal of Systems and Software*, 80(9):1425–1437.

[48] SYRIANI, E., LUHUNU, L. et SAHRAOUI, H. (2018). Systematic Mapping Study of Template-based Code Generation. *Computer Languages, Systems and Structures*, 52:43–62.

[49] THOMAS, J., BRUNTON, J. et GRAZIOSI, S. (2010). EPPI-Reviewer 4.0 : software for research synthesis. Tech report, University of London, EPPI-Centre, Social Science Research Unit, Institute of Education.

[50] XING, Z. et STROULIA, E. (2005). UMLDiff : An Algorithm for Object-Oriented Design Differencing. *In Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, pages 54–65. ACM.

Annexe A

Grammaire Xtext du modèle de configuration de RS

```
grammar org.xtext.language.relis.Relis with org.eclipse.xtext.common.Terminals
generate relis "http://www.xtext.org/language/relis/Relis"

Relis :
    (elements+=Project);
Project:
    'PROJECT' short_name=ID name=STRING
    ('SCREENING'
    screening=Screening)?
    ('QA'
    quality_assess=QA)?
    'DATA EXTRACTION'
    (category+=Category)+
    ('SYNTHESIS'
    (reporting+=Report)*)? ;

Screening:
    'Reviews' review_per_paper = INT
    'Conflict' ('on' conflict_type = ConflicType ) ('resolved_by' conflict_resolution =
        ConflictResolution)
    'Criteria' '=' '['exclusion_criteria+=Values (','exclusion_criteria+=Values)*']'
    ('Sources' '=' '['source_papers+=Values (','source_papers+=Values)*']')?
    ('Strategies' '=' '['search_startegy+=Values (','search_startegy+=Values)*']')?
    ('Validation' validation_percentage = INT '%' ( validation_assigment_mode =
        AssignmentMode )? )?
    ('Phases' (phases += Phase (',' phases += Phase)*))?;

QA:
    'Questions' '=' '['question+=STRING (','question+=STRING)*']'
    'Answers' '=' '['response+=Response (','response+=Response)*']'
    'Min_score' min_score = DOUBLE;

Response:
    title=STRING ':' score=DOUBLE;
```

```
33
34 Report:
35    Simple_graph|Compare_graph;
36
37 Simple_graph:
38    'Simple' name = ID (title = STRING)? 'on' value=[Category|ID] ('charts(' chart+=
          GraphType (','chart+=GraphType)*')');
39
40 Compare_graph:
41    'Compare' name = ID (title = STRING)? 'on' value=[Category|ID] 'with' reference =[
          Category|ID] ('charts(' chart+=GraphType (','chart+=GraphType)*')');
42
43 Phase:
44    (title = STRING) (description = STRING)? ('Fields''('fields+= Fields (','fields+=
          Fields)*')')?;
45 Category:
46    FreeCategory | StaticCategory | IndependentDynamicCategory |
          DependentDynamicCategory ;
47
48 FreeCategory:
49    'Simple' name = ID
50    (title = STRING)? (mandatory?='*')? ('['numberOfValues = nValues']')? ':' type=
          simpleType ('('max_char=INT')')? ('style(' pattern=STRING')' )? ('=' '['
          initial_value=STRING']')?
51    (
52       '{'
53          (sub_categories+=Category)+
54       '}'
55    )?;
56 Note:
57    'note' name = 'note' title = 'Note' type='string';
58 StaticCategory:
59    'List' name=ID (title = STRING)? (mandatory?='*')? ('['numberOfValues = nValues']')?
           '=' '['values+=Values (','values+=Values)+']'
60    (
61       '{'
62          (sub_categories+=Category)+
63       '}'
64    )?;
65
```

```
IndependentDynamicCategory:
   'DynamicList' name=ID (title = STRING)? (mandatory?='*')? ('['numberOfValues =
       nValues']')? (reference_name = STRING)? ('=' '['initial_values+=STRING (','
       initial_values+=STRING)*']')?
   (
      '{'
         (sub_categories+=Category)+
      '}'
   )?;
DependentDynamicCategory:
   'DynamicList' name=ID (title = STRING)? (mandatory?='*')? ('['numberOfValues =
       nValues']')? 'depends_on' depends_on=[Category|ID]
   (
      '{'
         (sub_categories+=Category)+
      '}'
   )?;
Values:
   name=STRING;
enum simpleType:
   int='int'| text='text'| string='string'| bool='bool'| real='real' | date='date';
enum AssignmentMode:
   Normal='Normal'| Veto='Veto'| Info='Info';
enum GraphType:
   bar='bar'| pie='pie' | line='line';
enum ConflictResolution:
   Majority='Majority'| Unanimity='Unanimity' ;
enum ConflicType:
   IncludeExclude='Decision'| ExclusionCriteria='Criteria';
enum Fields:
   Title='Title'| Abstract='Abstract'| Link='Link'| Preview='Preview'| Bibtex='Bibtex';
DOUBLE:
   INT ('.' INT)?;
nValues:
   '-1'|INT;
```

Annexe B

Couverture des fonctionnalités pour ReLiS

F: Fully supported

P: Partially supported

Composite	**Requirement**	**ReLiS**
Protocol Development	Ability to update the protocol at any stage	F
Protocol Development	Provide templates for the protocol	
Protocol Development	Support protocol piloting	P
Protocol Development	Support Protocol Development and Validation	
Decision Tracking	Tracability Tracking decisions	F
Differences of SLR and Mapping Studies	Support differences in the process between SLR vs. Mapping Studies	P
Connecting with Industry	Portal to find industry problems that can be turned to SLRs	
Data Maintenance	Reference management	
Data Maintenance	Document management	
Data Maintenance	Data management	F
Data Maintenance	Support backup and maintenance of data for longterm evolution of research	F
Traceability	The ability to take snapshots and roll back features	P
Traceability	Consistent documentation of results across the process	F
Traceability	Support for tracing extracted data back to source study	F
Traceability	Map relationships between papers	P
Data Sharing	Citation and Document Sharing	
Data Sharing	Share and reuse SLR data and meterial	
Collaboration Support	Support for distributed review teams	F
Collaboration Support	Support for collaborative SLR	F
Collaboration Support	Support for conflict resolution	F
Motivation and RQ Templates	Support for motivation and RQ (picco) templates	

Table B.I – Tool Coverage - Protocol

Composite	**Requirement**	**ReLiS**
Seach Execution	Support the ability pause/resume the search process	
Seach Execution	Search string standrization	
Seach Execution	Factoring search strings into multiple searches	
Seach Execution	Archive search queries and search process	P
Seach Execution	Export search results	
Search Results	Duplicates removal	
Search Results	Need mass export of results in different formats(bibtex, XLS, etc)	F
Search Results	Support search results filtering	
Storage of Studies	Automated storage of PDFs	
Storage of Studies	Management of stored studies	F
Storage of Studies	Ability to import external studies not from digital libraries	F
Integrated Search	Search and storage of data other than PDFs	
Integrated Search	Portal to search stored literature	
Integrated Search	Automated search in digital libraries	
Integrated Search	Support for collaborative search	
Integrated Search	Automated search and storage of studies in digital libraries	
Integrated Search	Integration with databases services	
Integrated Search	Unified protal to search literature in common digital libraries	
Integrated Search	Cross search of digital libraries	
Integrated Search	Importing search results from different databases	F
Integrated Search	Support piloting search string	
Integrated Search	Support for search iterations	

Table B.II – Tool Coverage - Search

Composite	Requirement	ReLiS
Collaboration Support	"Generate ""random set"" for validation (IRR)"	F
Collaboration Support	Support collaborative Selection process	F
Collaboration Support	Conflict detection and resolution	F
Selection Process	Citations and documents management	F
Support Text Mining	Natural language processor for included studies	
Support Text Mining	Automated search of similar studies based on defined keywords	
Support Text Mining	Textual analysis support	
Support Text Mining	Keywords definition support	
Support Text Mining	Automated keyword/phrases extraction, topic analysis in primary studies	
Support Text Mining	Support search automation for evidence in study	
Support Inclusion and Exclusion	Semi-Automated inclusion/exclusion of studies	
Support Inclusion and Exclusion	Automated support to map snowballing of references	
Support Inclusion and Exclusion	Support pilot of inclusion/exclusion	F
Support Inclusion and Exclusion	Semi-Automated to conduct snowballing	
Support Inclusion and Exclusion	Support for inclusion/exclusion criteria management	F

Table B.III – Tool Coverage - Selection

Composite	Requirement	ReLiS
Collaboration Support	Support for collaborative assessment of articles	F
Collaboration Support	Conflict resultions during assessment	F
QA Process	Provide QA checklist	F
QA Process	Support assessing study relevence	F
QA Process	Support the ability to assess the quality of study while extracting data	P
Assessing Articles Quality	Measuring study relevence using the content of the study	
Studies Diversity	Provide QA criteria for different types of studies	F
QA Measurement	Validating selected papers/studies	F
Process Standardization	Ability to create QA criteria list	F
Process Standardization	Provide quality assesment criteria template	
Process Standardization	Ability to reuse existing QA criteria	
Support Quality Assessment	Support quality assessment of studies, protocol, and process activites.	F

Table B.IV – Tool Coverage - Quality Assessment

Composite	**Requirement**	**ReLiS**
Data Extraction Form	Ability to revise data extraction form at any stage	F
Data Extraction Form	Ability to extract multiple studies from one article	
Data Management	Export extracted data	F
Process Efficacy Assurance	Data Extraction Validity	F
Tool Support	Integration with Excel	P
Tool Support	Extraction of bibliographical items from web pages	
Tool Support	Support traceability of extracted data	F
Tool Support	Data management	F
Tool Support	Document management	
Tool Support	Support automated reading of articles	
Collaboration Support	Comments for reasons made when applying criteria	F
Collaboration Support	Collaborative coding	F
Collaboration Support	Support collaborative data extraction process	F
Collaboration Support	Conflict resolution during extraction	
Coding of Methods and Data	Support encode study methodology	F
Coding of Methods and Data	Support categories management	F
Coding of Methods and Data	Quantitative data capture	F
Coding of Methods and Data	Support codes management and results	F
Coding of Methods and Data	Qualitative data capture	F
Coding of Methods and Data	Support pilot of data extraction	F
Coding of Methods and Data	Support PDF annotation	

Table B.V – Tool Coverage - Data Extraction

Composite	Requirement	ReLiS
Synthesizing Primary Studies	Tracing a definition back to the original article	F
Synthesizing Primary Studies	Population analysis of studies	
Synthesizing Primary Studies	Measure how close articles are	P
Synthesizing Primary Studies	Mix evidences from multiple studies	F
Synthesizing Primary Studies	Mix evidences across study design	
Synthesizing Primary Studies	Support export of synthesied data	F
Automated Analysis	Coding Support	
Automated Analysis	Automated analysis and summary of coding	
Automated Analysis	Automated classification based on coding	P
Automated Analysis	Automated reports generation of stuides metadata	F
Automated Analysis	Automated reports generation of statistical data, list of included/excluded stuides, and criteria	F
Automated Analysis	Support graph generation	F
Handling Data	Summarizing qualitative and quantitative data	F
Collaboration Support	Support collaborative process during the synthesis	F
Statistical Analysis	Support comparision of qualitative evidence	F

Table B.VI – Tool Coverage - Synthesis

Composite	Requirement	ReLiS
Report Preparation	Ability to aggreagate the extracted data in one place	F
Report Preparation	Generate a template for the documentation	
Validation	Report Validation	
Visualization	Visualization support of cluster of papers	F
Visualization	Generate presentations using Latex	
Visualization	Visualization support of qualitative findings	F

Table B.VII – Tool Coverage - Documentation