MEMOIRE

En vue de l'obtention du

MASTER PROFESSIONNEL

Ingénierie du Logiciel

Délivré par:

Faculté des Sciences de l'Université Libanaise

Présenté et soutenu par :
Nisrine Abdl Raouf Osman

le lundi 28 septembre 2015

Titre

Amélioration des fonctionnalites d'un outil pour une architecture logicielle multivues

---

Encadrant
Dr Hala NAJA

Lecteurs
Dr Salah EL FALOU
Dr Mohammad SAADE

Table des matières

Remerciements

Je tiens à remercier sincèrement Dr. Hala NAJA, pour la chance qu'elle m'a donnée de faire mon stage au sein du campus universitaire, pour sa poursuite de la démarche de mon travail, ses orientations et ses conversations enrichissantes.

Egalement, Je tiens à remercier Dr. Ahmad KHEIR, pour ses conseils et ses orientations concernant *Moval* et Moval-*Tool*.

J'adresse mes remerciements aux lecteurs Dr. Mohamad SAADE et Dr. Salah FALOU.

Je remercie ma famille, spécialement ma mère pour son support et son encouragement. Je n'oublie pas à remercier mes collègues pour leurs encouragements.

Glossaire

Terme	Définition
MOVAL (Model, View, and Abstraction Level based software architecture)	Une approche d'architecture logicielle hiérarchique multi-vue qui vise à réduire la complexité des systèmes informatiques.
MOVAL-TOOL	Un outil qui permet la création des architectures logicielles tout en appliquant l'approche *Moval.*
MOVAL – ADP	Processus de définition de l'architecture spécifique à *Moval*
Processus Unifié (*Unified process UP*)	Méthode de développement d'un logiciel itérative et incrémentale
ISO/IEC/IEEE 42010	Standard de définition de l'architecture logicielle multipoints de vue
MDA (Model Driven Architecture)	Démarche de réalisation de logiciels, proposée et soutenue par l'OMG basée sur les modèles.
MOF (*Meta-Object Facility*)	Standard de *l'Object Management Group* (OMG), s'intéresse à la représentation des méta-modèles et leur manipulation.

Stakeholder	Ensemble des acteurs intervenants dans le logiciel
Formalisme	Langage de modélisation et de notations graphiques ayant des sémantiques bien définies et non ambiguës.

Introduction

Moval est un méta-modèle de définition d'architectures multipoints de vues, qui a été initié par Dr. Ahmad KHEIR dans le cadre de sa thèse. Ce projet a été réalisé au sein de L'Université Libanaise à la faculté des sciences. Il est en collaboration entre L'Université Libanaise (Dr Hala NAJA) et L'Université de Nantes (Prof Mourad OUSSALLAH). L'approche *Moval* est dotée d'un outil spécifique appelé *Moval-Tool* qui est un prototype d'un outil complet servant les architectes logiciels à effectuer chaque opération dont ils ont besoin dans le processus de définition de leur architecture telle définie dans le processus de définition d'architecture de Moval, appelé *Moval-ADP*.

Objectifs du projet

Le but de ce projet est de mettre en œuvre toutes les fonctionnalités du prototype *Moval-Tool*. Il vise à définir quelques nouvelles fonctionnalités qui facilitent l'application de l'approche *Moval* dans un cas pratique.

Structure du rapport

La structure du rapport est comme suit :

- Dans le chapitre 1, nous présentons les concepts de base de *Moval* : les vues, les niveaux de réalisation et les niveaux de description. Ainsi nous exposons brièvement le processus de définition d'architecture *Moval-ADP*. Dans la seconde partie, nous définissons les principales caractéristiques du logiciel *Moval-Tool.*
- Dans le chapitre 2, nous présentons le cahier des charges, qui contient les exigences fonctionnelles à ajouter à l'outil *Moval-Tool,* que nous proposons à concevoir et implémenter.
- Dans le chapitre 3, nous présentons les scénarios du système où nous décrivons les principaux services améliorés dans *Moval-Tool.*
- Le chapitre 4 constitue la vue logique du système où nous décrivons comment les parties internes du système interagissent.

- Dans le chapitre 5, nous présentons l'aspect persistance des données et nous décrivons la structure de la base de données.
- Dans le chapitre 6, nous présentons la vue physique du système. Ainsi, nous dégageons nos perspectives, les avantages de ce stage et les obstacles que j'ai rencontrés.

Chapitre 1 : Présentation de *Moval*

1.1 Introduction

Ce chapitre vise à présenter *Moval* [2] (**Model, View, and Abstraction Level based software architecture)** pour clarifier aux lecteurs l'idée du projet. Tout d'abord, nous définissons les concepts de base de *Moval*, vues, niveaux d'abstraction et niveaux de description. Ensuite, nous exposons brièvement le processus de définition *Moval-ADP.* Enfin, nous présentons *Moval-Tool* et sa structure interne.

1.2 Qu'est-ce que *Moval* ?

D'après [1], *Moval* (**Model, View, and Abstraction Level based software architecture)** est une approche de construction d'architecture logicielle hiérarchique multi-vues [3], qui vise à l'organisation des modèles des systèmes informatiques dans des -hiérarchies à base de vues et de niveaux d'abstraction, afin de réduire la complexité de conception et de développement de ces systèmes, et afin de permettre une meilleure communication entre les intervenants concernés. En plus, cette approche est conforme aux standards *ISO/IEC/IEEE 42010*, *MOF*, *MDA* dans le domaine des architectures logicielles et de développement informatique.

1.3 Les Vues et points de vue dans *Moval*

Partant du fait qu'un système peut être analysé selon différentes vues (sous différents angles), d'après [1], nous pourrons dire qu'une architecture d'un système est constituée d'un ensemble de vues distinctes reflétant ses différents aspects. Ainsi, le nombre des vues évolue et dépend directement du type de problèmes étudiés ainsi que du domaine d'application. Les architectures logicielles vues sous cet angle, peuvent être appelées des architectures logicielles multipoints de vue.

Dans *Moval*, une architecture est constituée d'un ensemble de vues distinctes. Une vue représente une partie du système, qui est en relation avec un *stakeholder* spécifique. Elle est une représentation d'un ou plusieurs aspects structuraux de l'architecture qui illustre comment l'architecture traite un ou plusieurs problèmes soulevés par les *stakeholders*.

D'autre part, un point de vue [3-4] est définit dans *Moval* comme étant un regroupement des conventions spécifiant les opérations nécessaires pour construire une vue d'une architecture logicielle. Dans *Moval*, une vue est conforme à un ou plusieurs points de vue dans le sens qu'elle utilise les conventions dans les opérations de construction de ses modèles. Pour plus d'informations consulter.

1.4 Les Niveaux d'abstraction dans *Moval*

Les vues sont composées de plusieurs niveaux d'abstraction [3]. L'abstraction est la relégation de certains détails qui paraissent inutiles, et le fait de se focaliser sur les aspects les plus importants, à un instant donné, du système.

Dans *Moval*, deux types d'abstraction ont été définies, qui sont :

- l'abstraction de réalisation, dénotée niveau de réalisation.
- l'abstraction de description, dénotée niveau de description.

1.4.1 Les Niveaux de réalisation dans *Moval*

D'après [1], dans *Moval* un niveau de réalisation est conforme à un point de vue particulier. De plus, le passage d'un niveau de réalisation à un autre niveau plus concret, implique l'existence d'un lien de type is_{+A}. La figure 1.1 représente deux niveaux de réalisation n1 et n2 d'une architecture logicielle *Moval*. Ainsi, n2 étant plus concret que n1, alors n2 est lié par un lien vertical is_{+A} à n1. La transition du niveau de réalisation n1 au niveau n2 indique une transition d'une phase dans le -processus de définition de l'architecture logicielle, à une autre phase plus avancée.

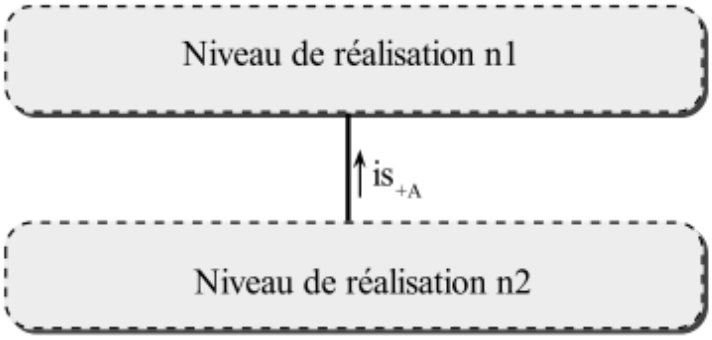


FIGURE 1.1 - Le lien is_{+A} entre deux niveaux de réalisation, extrait de [1].

1.4.2 Les Niveaux de description

D'après [1], un autre type d'abstraction défini dans *Moval* est l'abstraction de description. Ce type d'abstraction permet à l'architecte logiciel d'effectuer des Zoom In/Zoom Out sur les modèles associés à une vue de l'architecture logicielle.

Au sein d'une vue, les niveaux de description sont liés. Dans la figure 1.2 le niveau de description nd1ajoute plus de détails au niveau de description nd2. Ceci implique la présence du lien horizontal is_{+D} entre ces deux niveaux.

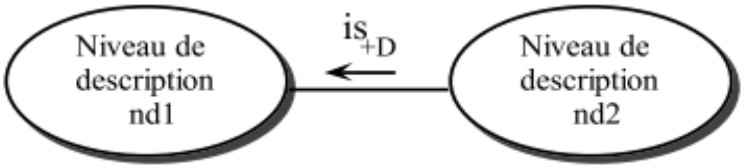


FIGURE 1.2 - Deux niveaux de description liés entre eux, extrait de [1].

Un niveau de description est constitué de modèles. Ces modèles sont au plus bas niveau de l'architecture. Ils peuvent être considérés comme étant les composants ou les éléments les plus importants dans une description d'architecture logicielle car ils représentent le moyen de communication le plus simple et efficace entre l'architecte logiciel d'une part, et les intervenants au système d'une autre part, pour que ces derniers puissent comprendre et analyser les situations que le premier essaie de modéliser. Ils peuvent être par exemple des diagrammes UML ou bien des schémas E/A expliquant le point de vue physique d'une base de donnée.

1.5 La Configuration dans *Moval*

D'après [1], Une configuration est définie dans *Moval* dans le but de minimiser et de simplifier la représentation de l'architecture logicielle. Une configuration représente un sous-ensemble de l'ensemble des modèles de l'architecture, choisis pour représenter les aspects voulus ou les préoccupations d'une catégorie des intervenants adressés. Ainsi, elle est définie par l'ensemble des intervenants pour lesquels elle est adressée, et par l'ensemble des répertoires contenant les modèles choisis.

1.6 Le Processus de définition d'une architecture *Moval*

Afin de profiter de l'approche *Moval* d'une manière efficace, une méthodologie spécifique est définie pour guider l'architecte pendant le développement de son architecture. Ainsi un processus de définition d'architecture spécifique à *Moval* est défini. Il s'appelle *Moval-ADP* et est conforme au processus unifié (UP).

Moval-ADP est un processus de développement itératif et incrémental, il est décomposé en différentes activités regroupées en quatre phases distinctes, illustrées dans la figure 1.3, qui sont : phase de l'Inception, phase d'Elaboration, phase de la Construction et phase de Transition. Ainsi, chaque phase peut être divisée en plusieurs itérations.

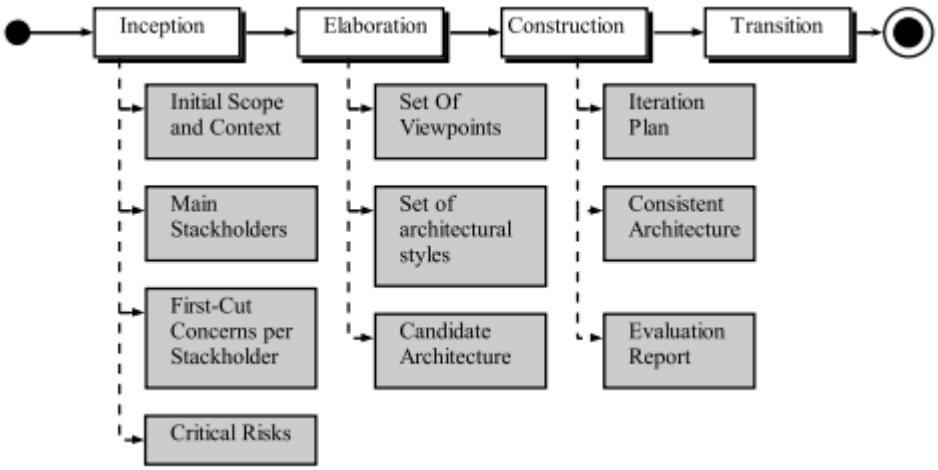


FIGURE 1.3 – Les phases de *Moval-ADP,* extrait de [1].

1.7 *Moval-Tool*

Moval-Tool est un outil qui permet à l'architecte d'appliquer les concepts de base de *Moval* et de créer une architecture logicielle multipoints de vue tout en passant par les phases du processus de la définition d'une architecture *Moval-ADP*.

Moval-Tool est implémenté sous la plateforme de développement de Microsoft, la plateforme .NET 4.0 [5], en utilisant le langage de programmation C#. En plus, l'interface graphique de cette application est construite en utilisant une librairie de contrôle graphique dénotée Dev Express v2010. Les informations utilisées dans l'application sont sauvegardées sous la forme de fichiers XML.

1.7.1 Les fonctionnalités du *Moval-Tool*

Moval-Tool implémente les fonctionnalités représentées dans le diagramme des cas d'utilisation de la figure 1.4. Par exemple, Il assure la création d'une architecture logicielle et la construction de cette architecture. Il permet aussi de la visualiser d'une façon hiérarchique ou matricielle.

Certaines fonctionnalités ne répondent pas à l'objectif final. Ainsi, cet outil peut encore présenter quelques *bugs* récalcitrants.

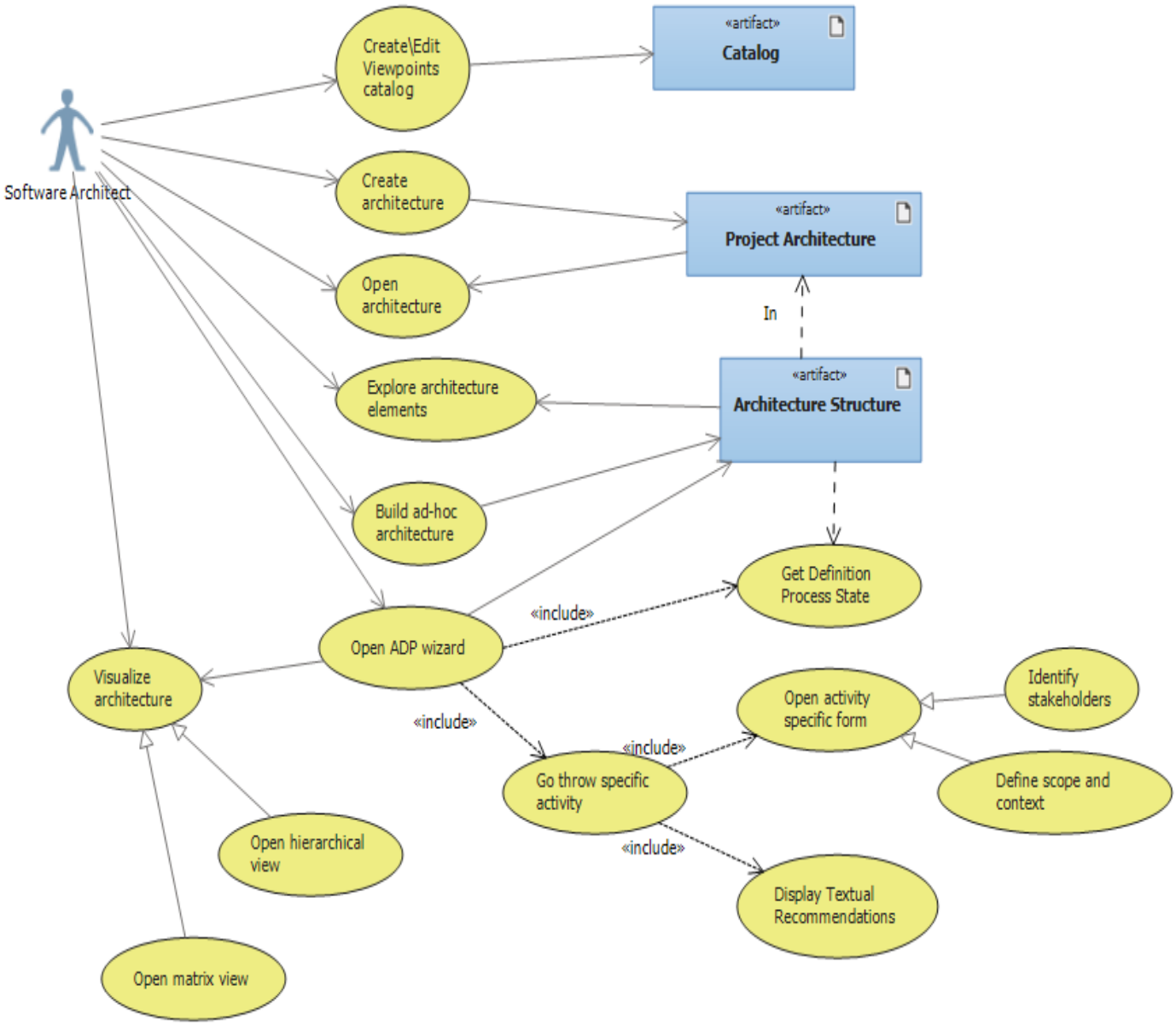


FIGURE 1.4 - Le diagramme UML des cas d'utilisation de *Moval-Tool*.

1.7.2 Vue Globale de la Structure Interne de *Moval-Tool*

Moval-Tool est subdivisé comme le montre la figure 1.5 en trois couches principales : La couche métier, la couche interface graphique et une couche de librairies extérieures.

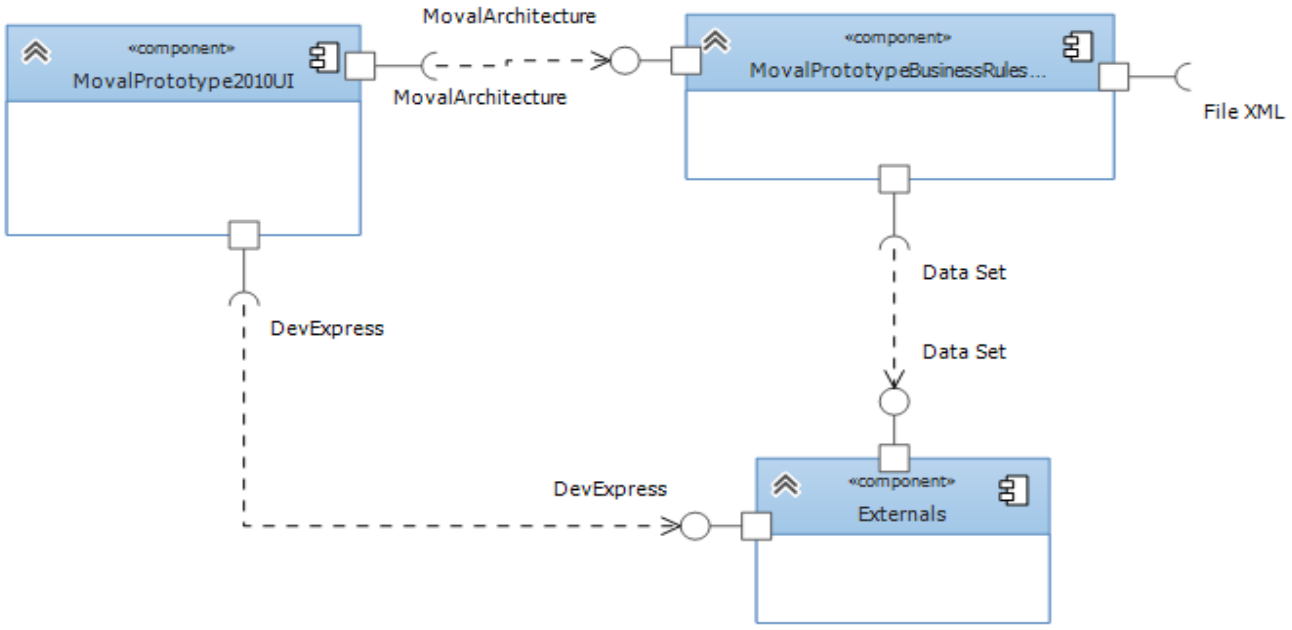


FIGURE 1.5 Le diagramme UML de composants de *Moval-Tool.*

1.7.3 GUI de *Moval-Tool*

L'interface graphique a été implémenté en utilisant une librairie de contrôle graphique dénotée Dev Express v2010.

Pour construire son architecture, l'utilisateur clique sur le menu « Architecture elements » puis sur « Architecture Designer » pour entrer à la page de l'éditeur d'une architecture illustrée dans la figure 1.6. Dans cet éditeur on trouve une fenêtre à gauche, dénotée "Tools" contenant les différents types d'éléments d'une architecture logicielle qui peuvent être ajoutés explicitement par l'architecte, qui sont : (1) une vue, (2) un niveau de réalisation, (3) un niveau de description, (4) un modèle, (5) un lien is_R et finalement un lien de correspondance $is_=$. Egalement, on trouve une autre fenêtre à droite, dénotée "Properties" représentant les propriétés de l'élément architectural sélectionné.

Ainsi, l'utilisateur clique sur « Visualization » pour choisir une visualisation « Hiérarchique », voir figure 1.7, ou sur « Matrix » pour choisir une visualisation matricielle de l'architecture, voir figure 1.8. Dans la figure 1.9, la page de l'*ADP-Wizard* est montrée. Il peut cliquer sur « Architecture tree view » pour une visualisation arborescente de l'architecture comme illustré dans la figure 1.11. En plus, La figure 1.10 présente la page de la création d'une configuration.

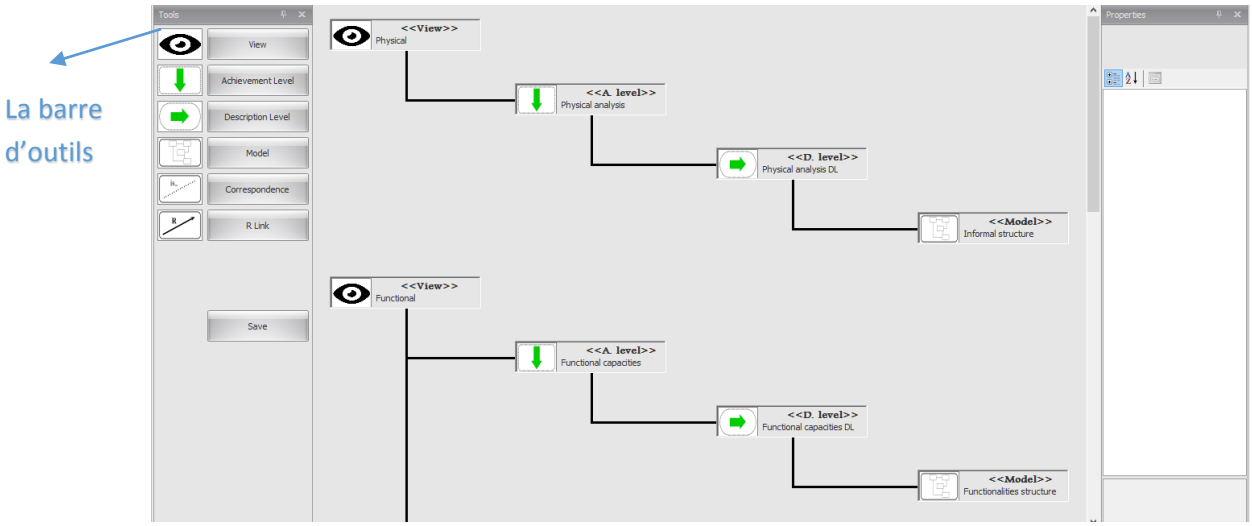


FIGURE 1.6 L'éditeur de l'architecture dans *Moval-Tool*.

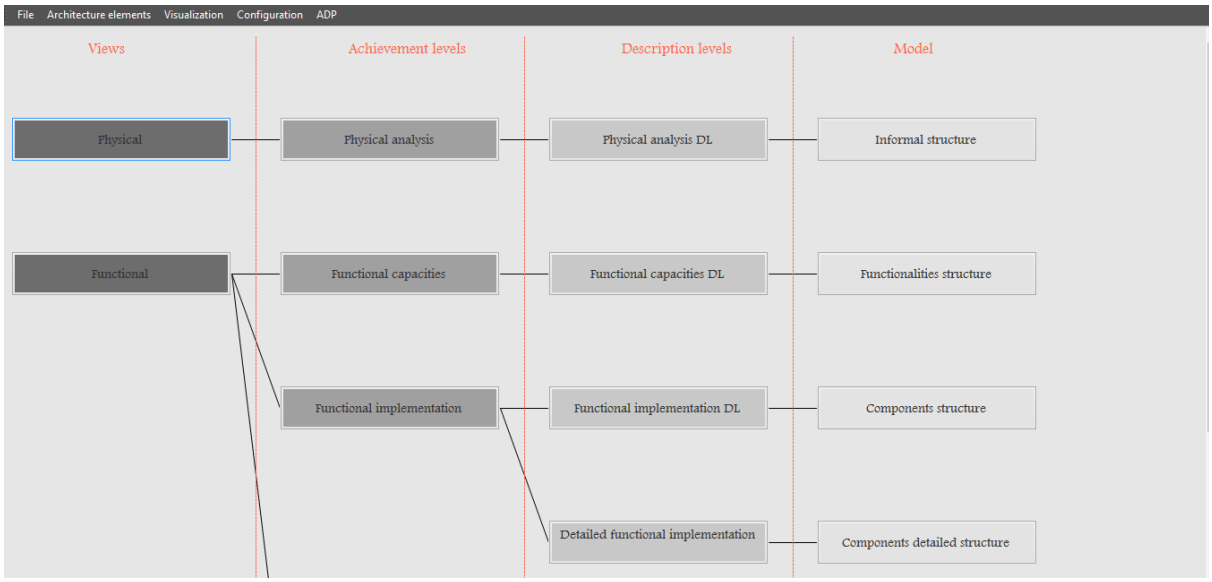


FIGURE 1.7 Représentation hiérarchique d'une architecture dans *Moval-Tool*.

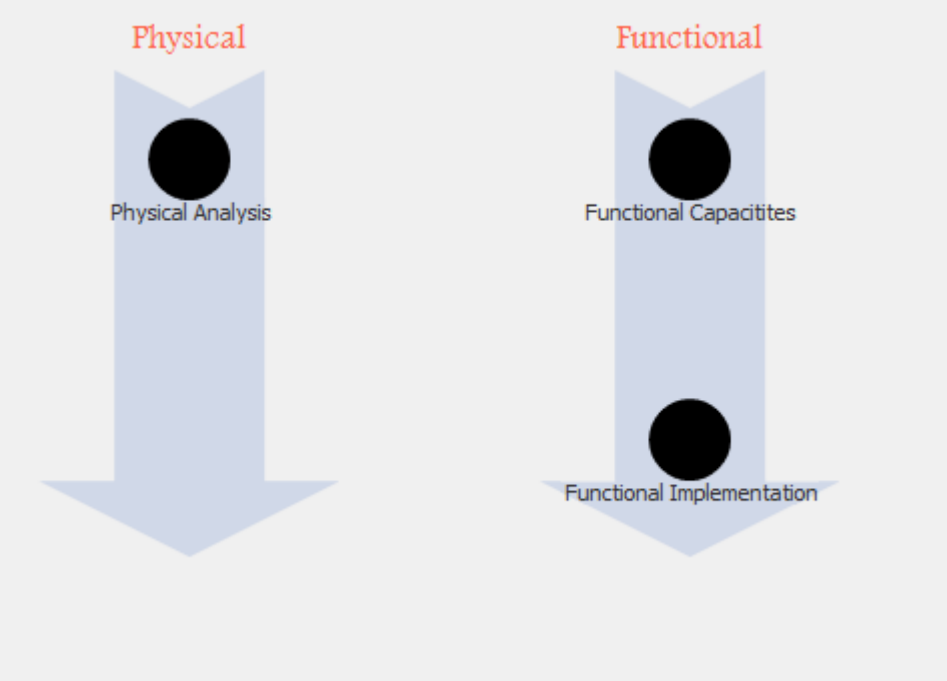


FIGURE 1.8 Représentation matricielle de l'architecture.

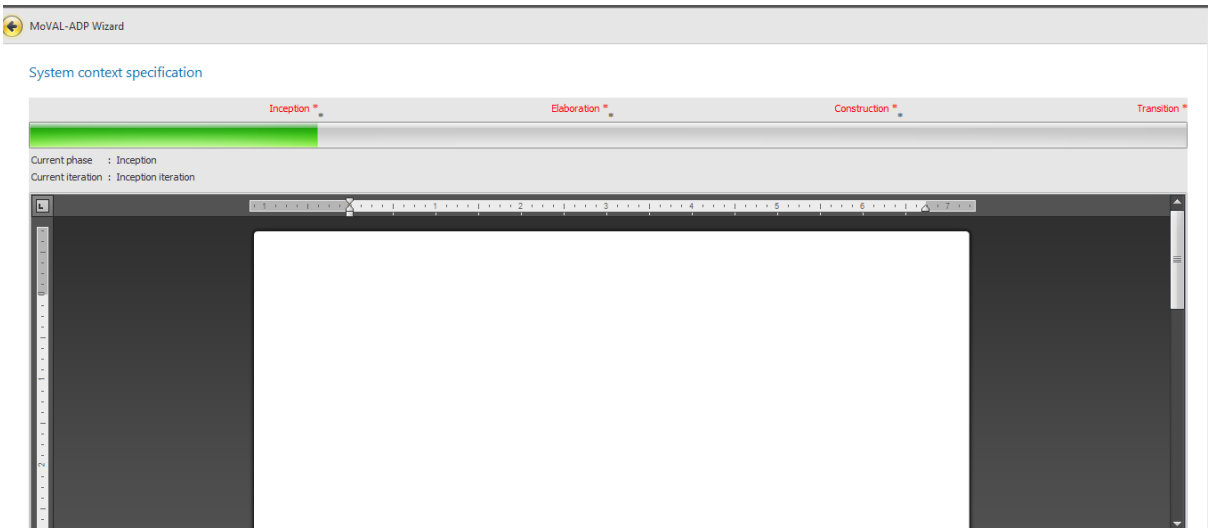


FIGURE 1.9 L'activité de spécification du contexte d'un système de *l'ADP-Wizard*.

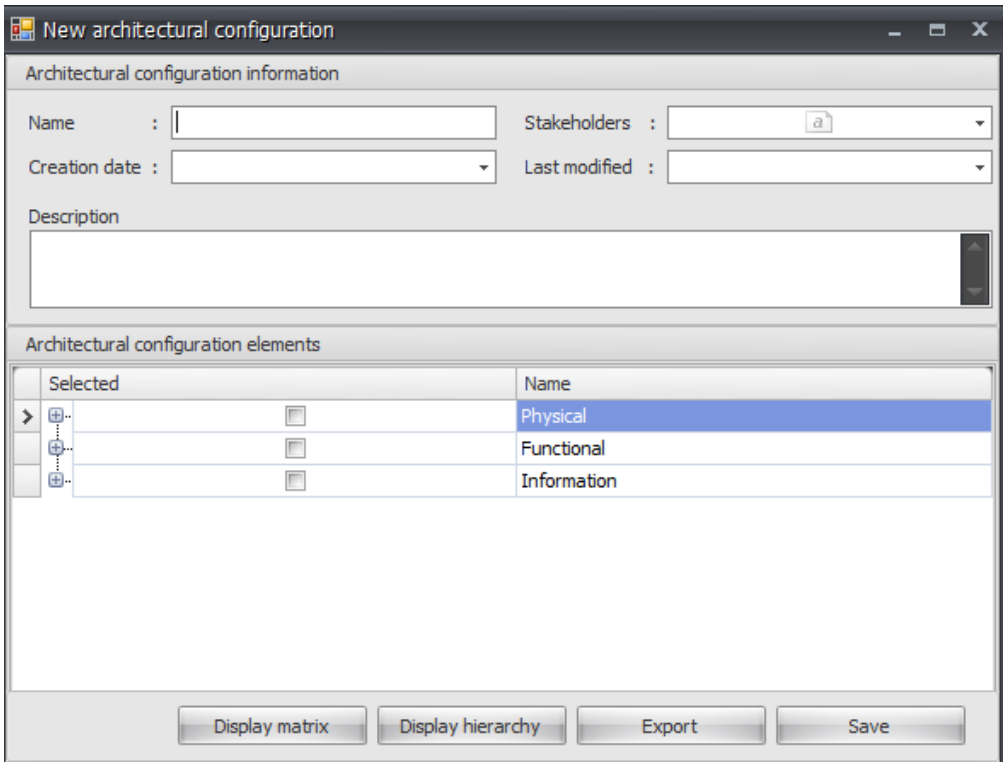


FIGURE 1.10 Création d'une configuration dans *Moval-Tool*.

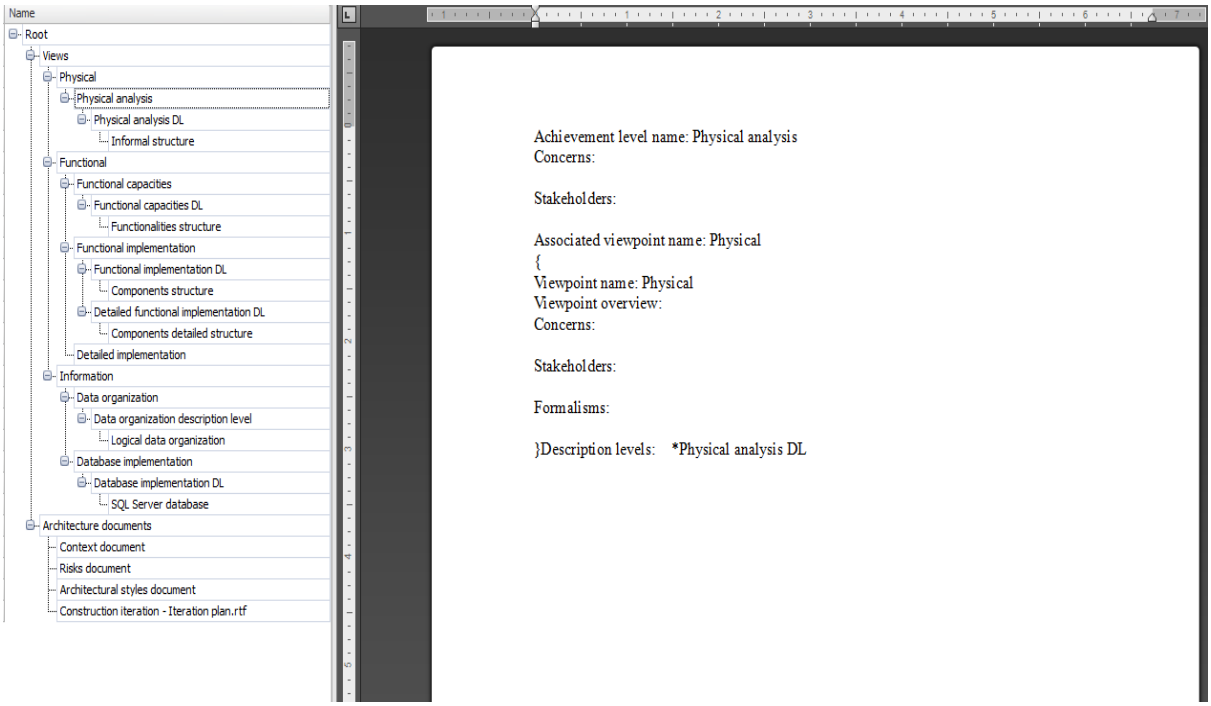


FIGURE 1.11 Visualisation arborescente de l'architecture

1.8 Conclusion

Dans ce chapitre, nous avons présenté les caractéristiques de *Moval*. Ensuite, nous avons présenté l'outil *Moval-Tool*. Dans le chapitre suivant, nous dégageons les limitations de cet outil que nous cherchons à dépasser/résoudre.

Chapitre 2 : Cahier des Charges

2.1 Introduction

Dans ce chapitre nous allons dégager les fonctionnalités que nous devons ajouter ou modifier dans l'outil *Moval-Tool*. Nous commençons à dégager les limitations. Ces limitations sont exprimées sous forme d'erreurs et d'améliorations que nous proposons. Avant de les exposer (voir paragraphes 2.4 - 2.5), nous dégageons les règles de validation d'une architecture *Moval* que nous avons définies et implémentées dans le *Moval-Tool*. Ensuite nous dégageons clairement les besoins.

2.2 Les règles de validation d'une architecture *Moval*

Nous définissons dans *Moval* quelques règles de validation de l'architecture. Celles-ci vérifient si une architecture construite par l'architecte dans l'éditeur de la figure 1.7 est valide ou non. Voici les différentes règles :

- RVAL1 : La validation d'une vue

  Chaque vue dans l'architecture est valide si elle possède un nom, au moins un intervenant (*stakeholder*), et des préoccupations. Aussi, une vue qui n'a aucun niveau de réalisation, est considérée non valide.
- RVAL2 : La validation d'un niveau de réalisation

  Un niveau de réalisation est considéré valide lorsqu'il possède un nom, au moins un stakeholder, des préoccupations et un point de vue déterminé. Aussi, un niveau de réalisation qui n'a aucun niveau de description est considéré non valide.
- RVAL3 : La validation d'un niveau de description

  Un niveau de description doit posséder un nom et au moins un modèle pour qu'il soit valide. Ainsi, un niveau de description qui n'ajoute pas plus de détails à un autre du même niveau de réalisation n'est pas considéré valide.
- RVAL4 : La validation d'un modèle

  Un modèle est valide s'il possède un nom, un formalisme et une image pour une représentation graphique.
- RVAL5 : Règle de suppression d'un élément architectural

La suppression d'un élément de l'architecture est soumise à des règles sémantiques. En effet, la suppression d'une vue n'est pas autorisée lorsqu'elle contient des niveaux de réalisation, de même la suppression de ces derniers est interdite lorsqu'ils contiennent des niveaux de description. Un niveau de description ne peut pas être supprimé s'il contient des modèles. Tandis que la suppression d'un modèle ou d'un lien n'est soumise à aucune règle d'interdiction. Néanmoins, si l'architecte souhaite forcer la suppression d'un élément contenant des sous éléments, nous procédons à une suppression par cascade.

2.3 Les besoins en termes d'erreurs et d'améliorations

D'après les tests que nous avons menés sur l'outil, nous avons dégagé deux types de besoins : les bugs et les améliorations. Dans les paragraphes suivants nous détaillons chacun des deux types.

2.4 Les bugs ou erreurs

Par **bug**, nous désignons tout comportement du système qui génère une erreur ou un mal-fonctionnement de ce dernier. Dans ce qui suit, nous détaillons les bugs obtenus lors du test du logiciel. Chaque bug étant référencé par un code suivi de sa description.

2.4.1 *B1-MPL*: Missing Project Location

Problème existant : Lors de la création d'une nouvelle architecture, un nouvel projet sera créé dans la location déterminée par l'utilisateur. Par défaut, c'est « A : \Workspace » comme le montre la figure 2.1. Or, ce chemin n'est pas le même sur tous les ordinateurs. Un utilisateur ne change pas la location par défaut un bug sera généré.

Notre solution : Pour assurer la portabilité du *Moval-Tool*, il est nécessaire de déterminer par défaut une location du projet qui est adaptable à l'environnement de l'utilisateur. Nous l'avons déterminé. Ainsi, nous avons ajouté un sous-menu **settings** dans lequel un utilisateur peut changer l'espace par défaut (***workspace***) de ses projets.

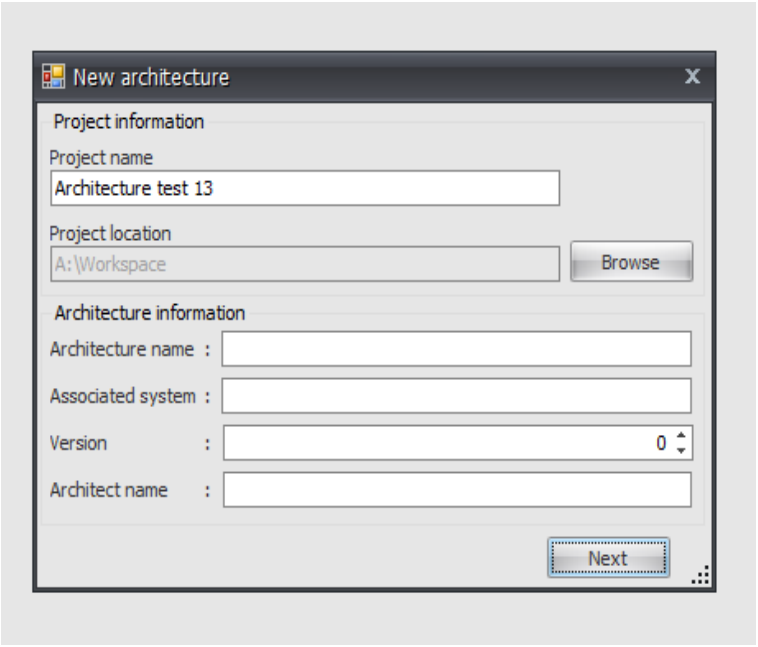


FIGURE 2.1 : Location du Projet.

2.4.2 *B2-MSAI*: Missing Architecture Information Storage

Problème existant : Lors de la création d'une architecture, l'utilisateur remplit le nom de l'architecte, la version et le nom du système comme le montre la figure 2.1. **Or ces informations ne sont pas stockées**.

Notre solution : Notre solution consiste à **stocker** ces informations dans le **fichier XML** correspondant à cette architecture.

2.4.3 *B3-CNA*: Error when Creating New Architecture

Problème existant : Quand l'utilisateur **est déjà ouvert** une architecture, puis a décidé de **créer une autre nouvelle architecture**, un bug se déclenchera comme le montre la figure 2.2.

Notre solution : Notre solution consiste à permettre la création d'une architecture même si une autre était déjà ouverte sans aucun problème.

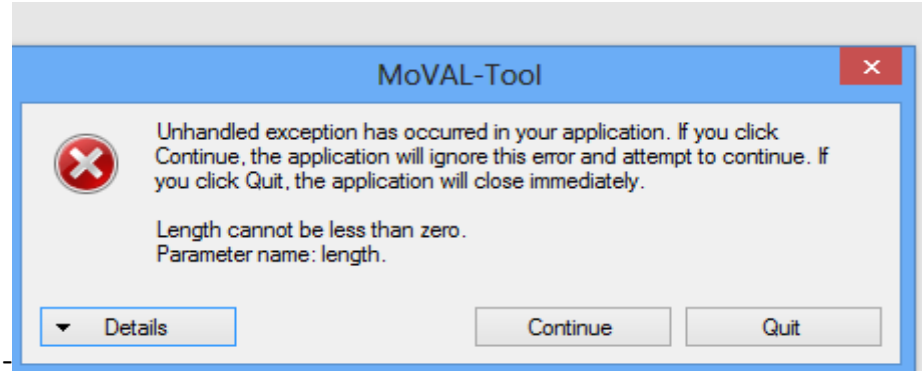


FIGURE 2.2 Erreur lors de la création d'une nouvelle architecture.

2.4.4 *B4-OEA*: Error when Opening an Existent Architecture

Problème existant: Un bug est déclenché par conséquence du **mal fonctionnement** de l'outil lors de l'ouverture de plusieurs architectures **successivement**. Voir Figure 2.3. Ainsi, les éléments d'une architecture peuvent être **totalement effacés**.

Notre solution : Notre solution consiste à permettre à l'architecte d'ouvrir plusieurs architectures alternativement sans aucun problème.

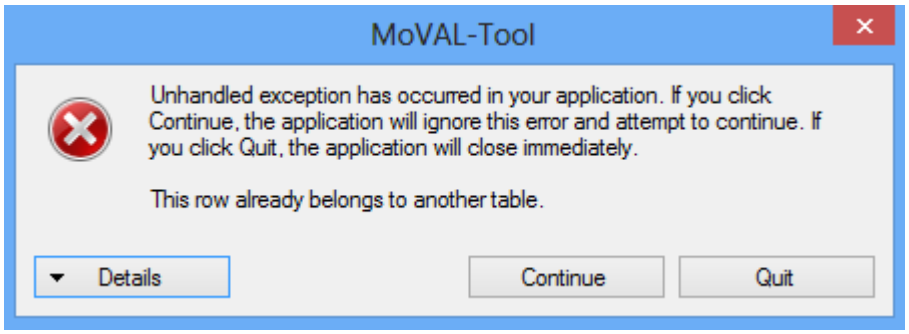


FIGURE 2.3 Erreur lors de l'ouverture d'une architecture.

2.4.5 *B5-LBD:* Link Buttons are disabled

Problème existant : Lors de la construction de l'architecture dans l'éditeur de la figure 1.7, **les liens entre les modèles, les niveaux de description et les niveaux de réalisation ne sont pas accessibles**. Un utilisateur ne peut pas ajouter des liens à son architecture. **Les boutons : *Rlink* et *Zoom in*** illustrés dans la Figure 2.4 de l'éditeur d'une architecture **ne fonctionnent pas**.

Notre solution : Notre solution consiste à implémenter les évènements déclenchés par ces boutons et donc permettre à l'architecte d'ajouter des liens de types is_{+D} et is_{R} entre les éléments d'une architecture. Ainsi nous ajoutons un onglet à cette fenêtre qui permet l'accès à une vue matricielle et qui permet d'ajouter des liens is_{+A} entre les niveaux de réalisation de l'architecture.

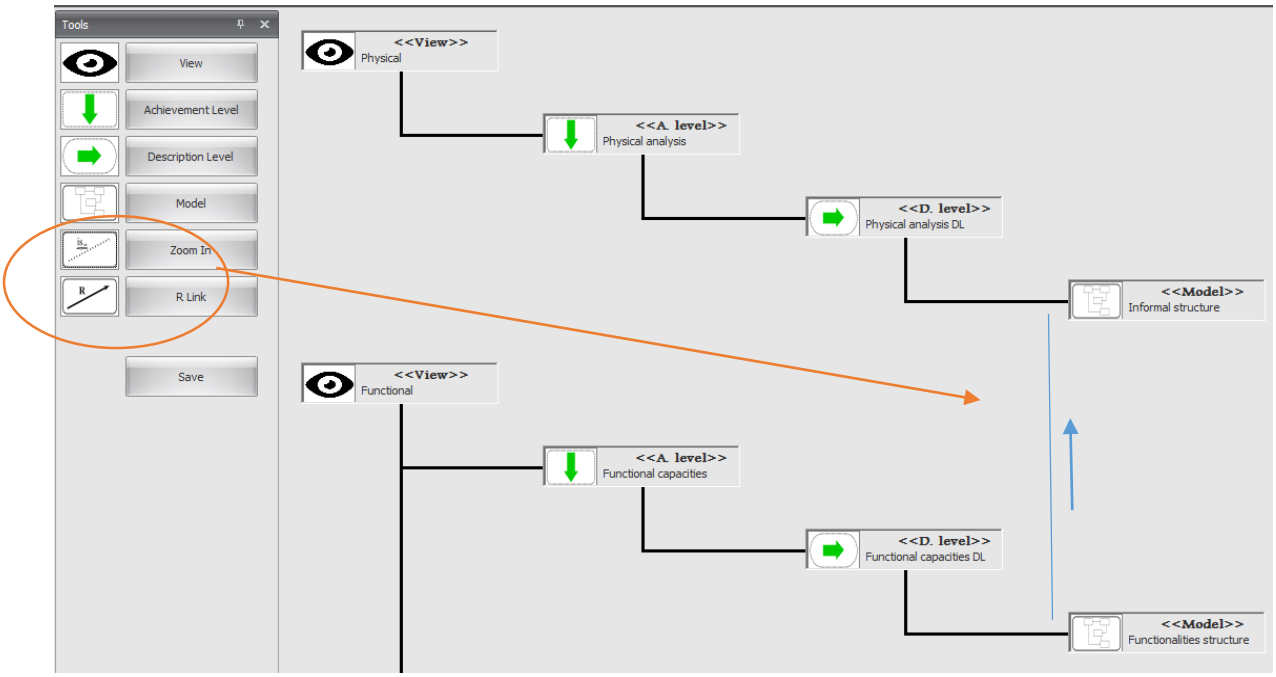


FIGURE 2.4 Les liens entre les éléments de l'architecture

2.4.6 *B6-MDEF:* Missing Deleting Architectural Element Functionality

Problème existant : Une erreur est déclenchée lors de la suppression d'un modèle, d'un niveau de description, d'un niveau de réalisation ou d'une vue. Cette erreur informe que **la méthode *delete* n'est pas implémentée**.

Notre solution : Notre solution consiste à l'implémentation de la méthode de suppression d'un élément architectural tout en appliquant la règle de suppression des éléments (voir la section 2.2). Si la suppression est valide selon cette règle, elle aura lieu sans aucun problème. Si l'élément qu'on désire supprimer contient des éléments, le système doit informer l'utilisateur que l'élément n'est pas vide (i.e. il contient des éléments), auquel cas l'utilisateur peut soit annuler la suppression soit forcer une suppression par cascade.

2.4.7 *B7-PS*: Problem when Saving

Problème existant : Lorsque l'utilisateur ajoute un nouvel élément architectural et clique sur le bouton *Save* plusieurs fois, l'élément ajouté sera dupliqué et donc sauvegardé plusieurs fois dans l'architecture comme le montre la figure 2.5.

Notre solution : Notre solution consiste à permettre à l'architecte de sauvegarder une seule fois un élément même si plusieurs cliques sur le bouton *Save* ont eu lieu concernant le même élément.

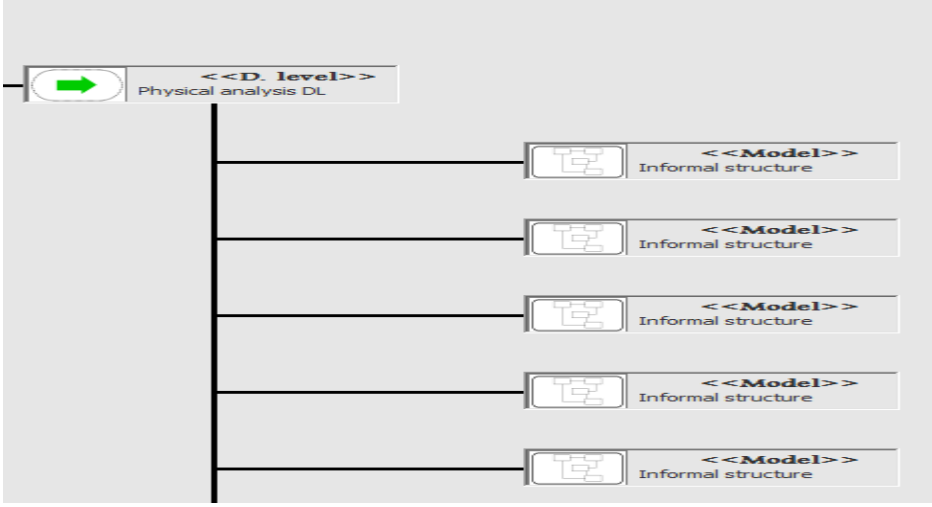


FIGURE 2.5 Duplication du modèle *Informal structure*

2.4.8 *B8- EAV*: Error in Architecture Visualization

Problème existant : Une Erreur peut être déclenchée lors de la visualisation hiérarchique ou matricielle d'une architecture. Voir figure 2.6.

Notre solution : Permettre la visualisation d'une architecture sans problème.

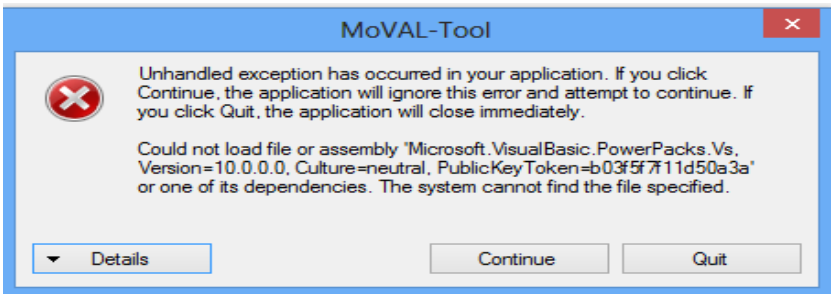


FIGURE 2.6 : Erreur lors de la visualisation de l'architecture

2.4.9 *B9-DMM*: Display of Multiple Models

Problème existant : Comme le montre la figure 2.7, même si un niveau de description contient plusieurs modèles, uniquement un seul est affiché dans la représentation hiérarchique de l'architecture.

Notre solution : Affichage de tous les modèles dans une représentation hiérarchique.

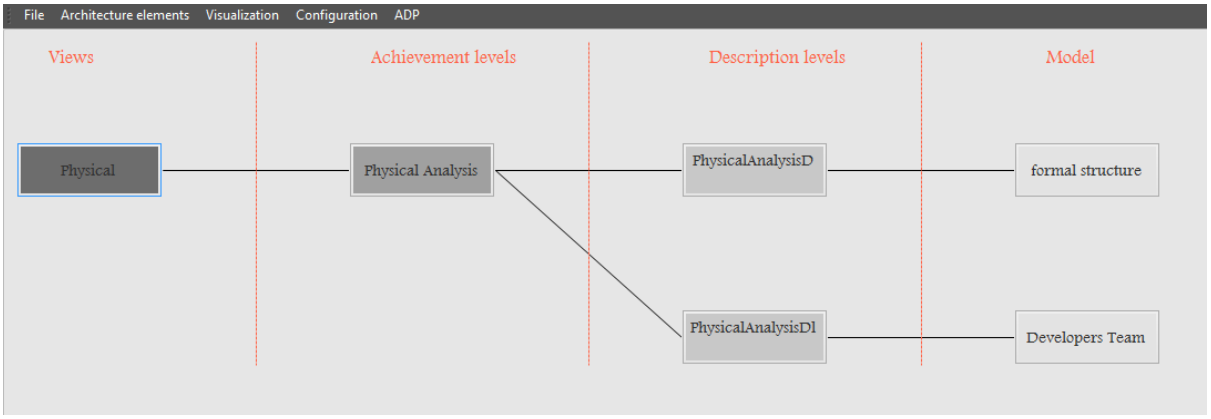


FIGURE 2.7 Visualisation hiérarchique d'une architecture

2.4.10 *B10-EAW*: Error in *ADP-Wizard*

Problème existant : Après l'ouverture de la fenêtre de l'*ADP-Wizard* illustré dans la figure 1.9, une erreur est déclenchée lorsque l'utilisateur laisse vide les noms de *stakeholders* comme le montre la figure 2.8.

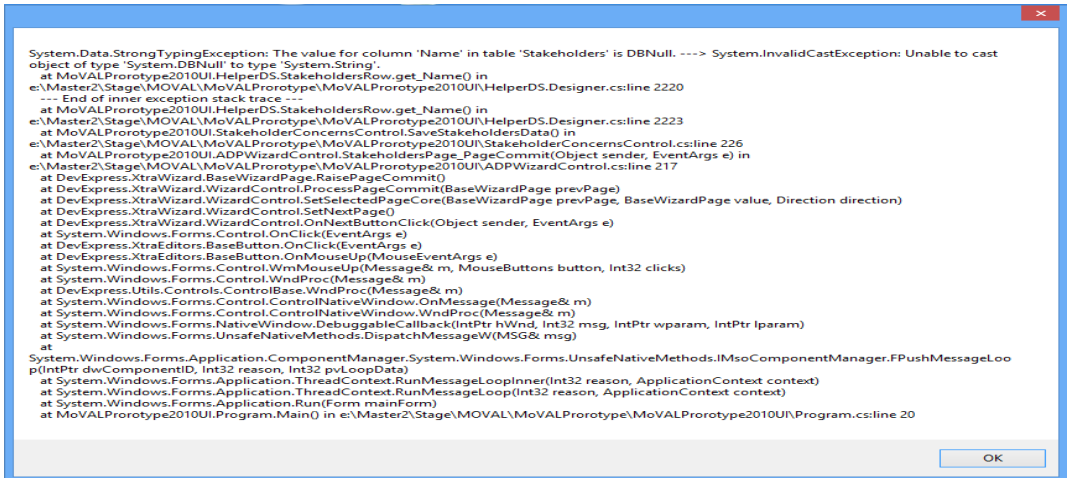


FIGURE 2.8 : Erreur lors de la navigation dans l'*ADP-Wizard*.

Notre solution : Notre solution consiste à ne pas déclencher un bug lorsque les noms de *stakeholders* sont nuls.

2.4.11 B11-*PBD*: Problem in Progress Bar Displaying

Problème existant : La barre de progression sera dupliquée lorsque l'utilisateur navigue sur l'*ADP-Wizard* et surtout lorsqu'il presse sur le bouton back comme il est montré dans la figure 2.9.

Notre solution : La fenêtre de l'*ADP-Wizard* représente une seul barre de progression qui ne sera pas dupliqué lors de la navigation sur l'*ADP-Wizard*.

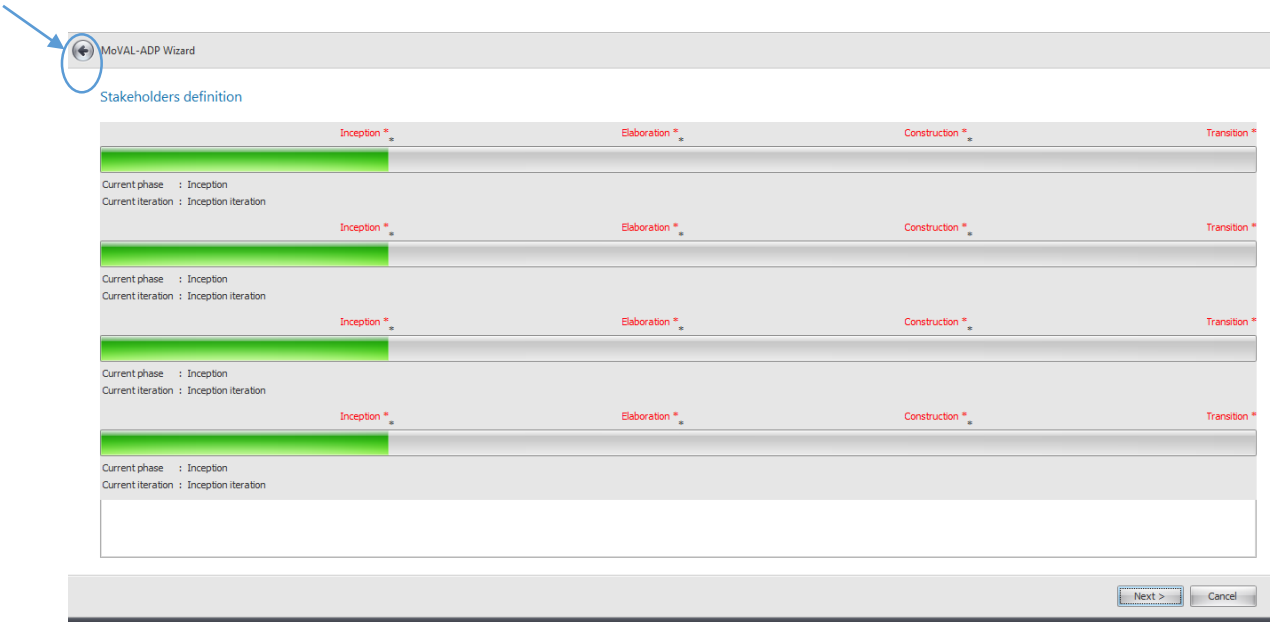


FIGURE 2.9 : Les itérations dans l'*ADP-Wizard*.

2.4.12 *B12-PVC*: Error when Creating Point of Views Catalog

Problème existant : Lors de la création d'un catalogue de points de vue de l'architecture, une erreur est déclenchée comme montré dans la figure 2.13 si l'utilisateur a cliqué sur le bouton « *import* » et n'a pas choisi un fichier pour importer.

Notre solution : Pouvoir créer un nouvel catalogue sans déclencher des bugs et en prenant en considération les fautes des utilisateurs.

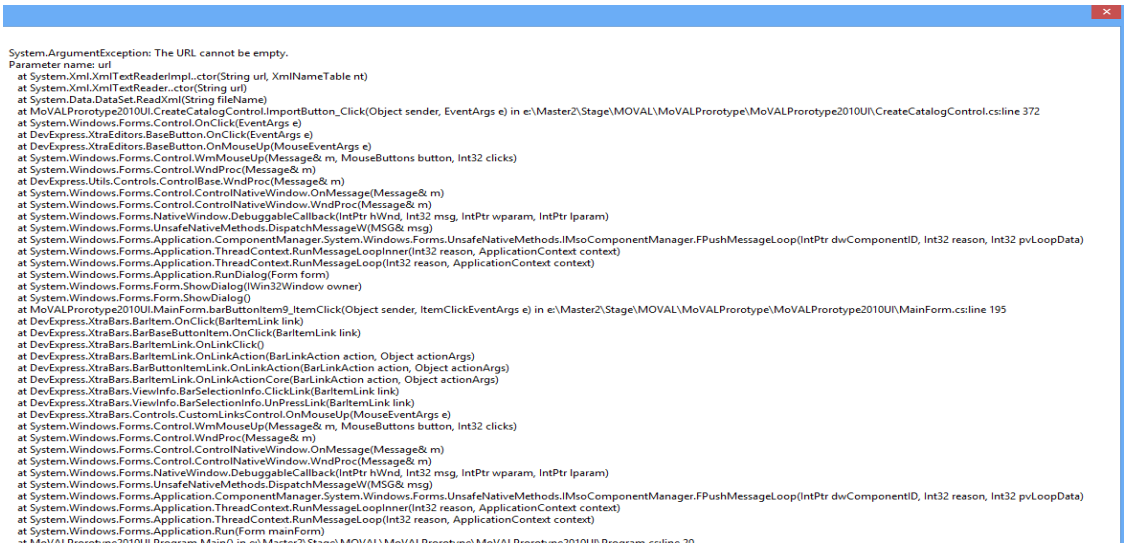


FIGURE 2.10 : Erreur lors de la création du catalogue

2.4.13 *B13-SMI*: Error when Saving Model Image

Problème existant : Lors de la sauvegarde des éléments de l'architecture, l'image ajoutée à un modèle dans l'éditeur de la figure 1.6 ne sera pas stockée. Alors, elle ne peut pas être visualisée dans la visualisation arborescente de l'architecture (figure 1.11) lorsque l'architecte clique sur ce modèle.

Notre solution : Notre solution est de permettre à l'architecte de sauvegarder l'image qu'il ajoute à son modèle sans aucun problème. Ensuite, les images seront affichées lorsqu'il clique sur les modèles dans la visualisation arborescente de l'architecture.

2.4.14 Tableau Récapitulatif de tous les bugs dans *Moval-Tool*

Table 2.1 - Tableau Récapitulatif de l'analyse des bugs dans *Moval-Tool*

Bug ID	Bug Title	Our Solution
B1-MPL	Missing Project Location	Add Settings – change Workspace option to determine the default location of *Moval* projects
B2-MSAI	Missing Architecture Information Storage	*Moval-Tool* has to save architect name, version and system information

B3-CAN	Error when Creating New Architecture	Creating new architectures without problems
B4-OAE	Error when Opening an Existent Architecture	Opening existent architecture without problems
B5-LBD	Link Buttons are disabled	Enable Links between architecture element
B6-MDEF	Missing Deleting Architectural Element Functionality	Implement delete methods
B7-PS	Problem when Saving	Prevent elements duplication while saving architecture
B8-EAV	Error in Architecture Visualization	Visualize architecture without any problem
B9-DMM	Display of Multiple Models	Display all architecture models in hierarchical view
B10-EAW	Error in *ADP-Wizard*	Let the stakeholders names be null
B11-PBD	Problem in Progress Bar Displaying	Do not duplicate progress bar while pressing on back button
B12-PVC	Error when Creating Point of Views Catalog	Create new catalog without problems
B13-SMI	Error when Saving Model Image	Save model image and display it in tree view

2.5 Les améliorations ou enhancements

Par enhancement, nous désignons toutes les fonctionnalités ajoutées au logiciel dans le but de son amélioration. Dans ce qui suit, nous détaillons les enhancements que nous ajouterons au système. Chaque enhancement étant référencé par un code suivi de sa description.

2.5.1 *E1-ADV*: Architecture Designer Validation

Avant amélioration : Le logiciel ne présente aucune fonctionnalité qui vérifie la validation de l'architecture créée par l'architecte.

Notre proposition : Nous proposons d'ajouter des règles de validation RVAL1, RVAL2, RVAL3, RVAL4 (voir la section2.2) pour vérifier la validation de l'architecture après la construction. Ainsi nous avertissons l'utilisateur dans le cas de l'invalidation ; Par exemple, dans le cas de la création d'une vue dépourvue d'aucun niveau de réalisation.

2.5.2 *E2-ICAV*: Enable Inactive Check Boxes when Architecture Visualization

Avant amélioration : L'interface graphique de la visualisation matricielle de l'architecture contient à gauche des cases à cocher (***check boxes***) qui sont **inactives.** La visualisation matricielle de l'architecture est **sous une forme linéaire**, représentant chaque vue de l'architecture dans une ligne ayant les niveaux de réalisation comme points sur cette ligne comme le montre la figure 2.11.

Notre proposition : Lorsque les quatre cases à cocher sont inactives, Nous proposons de les mettre en fonction et préciser à chacun son rôle et sa fonction :

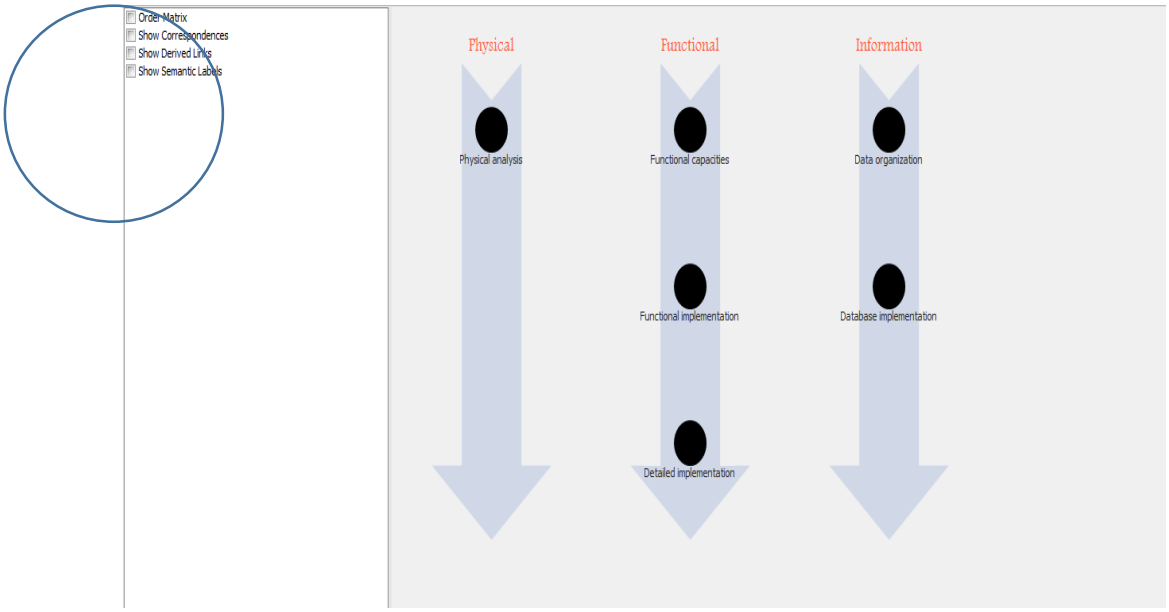


FIGURE 2.11 : visualisation matricielle d'une architecture.

- Order Matrix : Les niveaux de réalisation de chaque vue seront replacés en respectant les correspondances entre les vues.
- Show Correspondances : Les liens de correspondance is= seront visibles entre les différentes vues de l'architecture.
- Show Derived Links : Les liens entre les vues et les niveaux de réalisation seront visibles sur la matrice.
- Show Semantic Label : Les labels sémantiques des liens sont visibles.

2.5.3 *E3-ECF*: Enable Create Configuration Functionality

Avant amélioration : La création de la configuration d'une architecture n'est pas encore assurée par le logiciel. Les boutons illustrés dans figure 2.12 **ne fonctionnent pas**.

Notre proposition : Nous proposons de changer l'interface graphique de cette configuration et d'ajouter un bouton **« *next* »** qui amène l'architecte à une visualisation hiérarchique dont l'architecte peut choisir les modèles qu'il veut ajouter à sa configuration. Puis, on ajoute un bouton « ***Visualize it now*** »qui permet la visualisation hiérarchique de cette configuration.

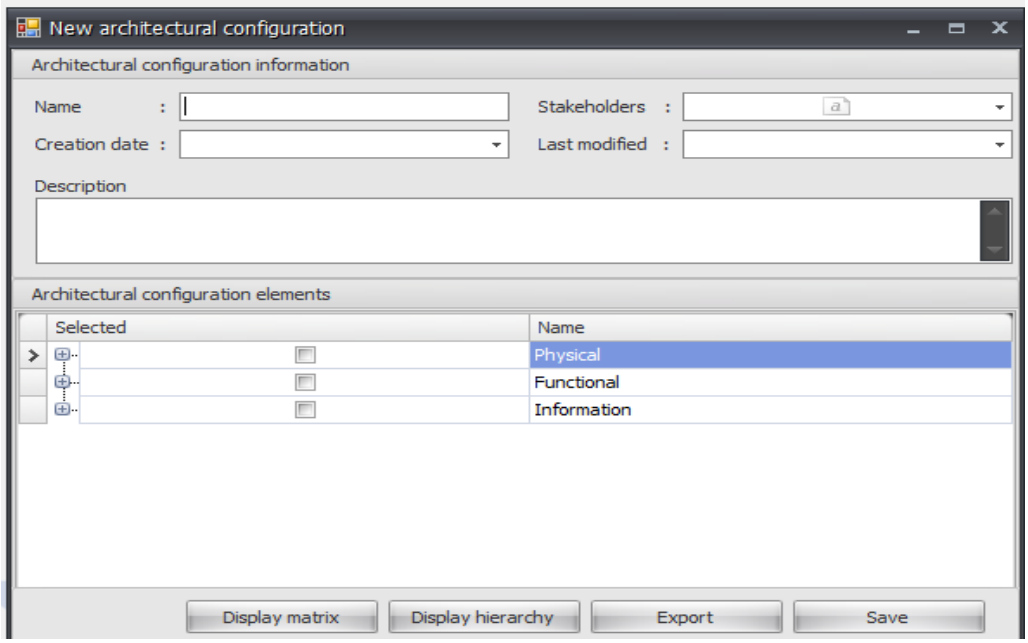


FIGURE 2.12 : Création de la configuration d'une architecture.

2.5.4 *E4-OC*: Open Configuration

Avant amélioration : Les configurations créées ne seront pas affichées par l'architecte ensuite il ne peut pas les éditer.

Notre proposition : Afin de permettre à l'architecte d'éditer sa configuration, nous lui permettons d'ouvrir une configuration, soit en mode visualisation soit en mode édit.

2.5.5 *E5-AORO*: ADP Wizard is an Option Rather than Obligation

Avant amélioration : seul le mode *Adp-assisted* de la création d'une architecture existait auparavant.

Notre proposition : Afin d'assister l'architecte lors du processus de définition d'une architecture, nous lui proposons deux choix de modes :

- Mode ***Ad-hoc*** : consiste à construire l'architecture par l'architecte lui-même sans passer par l'*ADP-Wizard*.
- Mode ***Adp-assisted*** : consiste à passer par les étapes du processus de définition d'une architecture lors de sa création.

2.5.6 *E6-IVAW*: Iterations View in *ADP-Wizard.*

Avant amélioration : Vu que Le processus de définition d'une architecture est un processus itératif et incrémental, chaque phase du processus est composée de plusieurs itérations (voir la section 1.6). Les itérations de chaque phase (it1, it2, etc…) sont visualisées chacune dans une nouvelle barre de progression comme il est montré dans la figure 2.13.

Notre proposition : Nous proposons de présenter les itérations d'une façon linéaire sur le même *Progress Bar*.

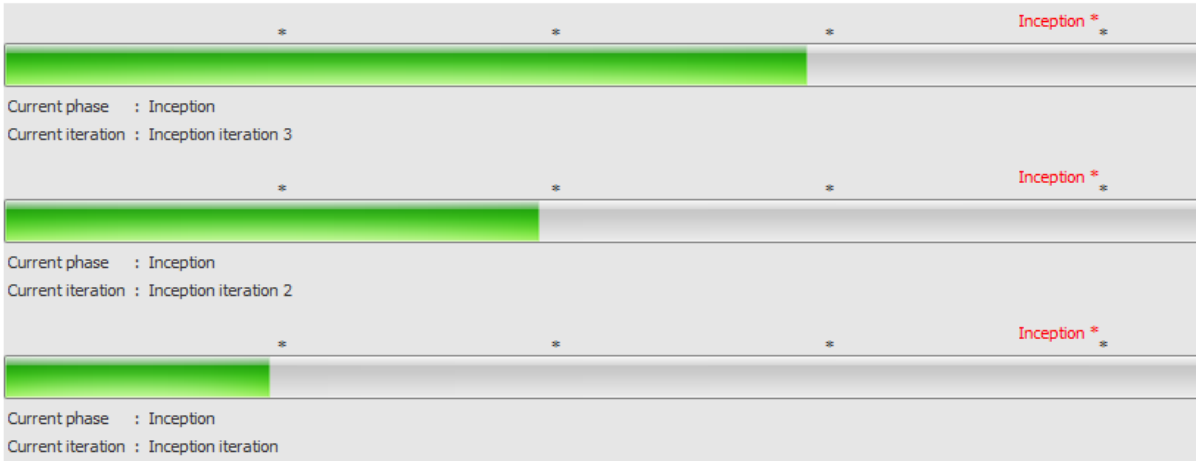


FIGURE 2.13 : Le ADP Wizard de l'architecture *Moval*

2.5.7 *E7-SA:* Save As

Avant amélioration : Le système n'a aucun bouton qui fait « Save as» de l'architecture ça veut dire qu'on ne peut pas créer une copie de notre architecture dans une autre location.

Notre proposition : Nous proposons d'ajouter une fonctionnalité « Save as » au système.

2.5.8 *E8-AC:* Architecture Creation

Avant amélioration : La création d'une nouvelle architecture dans un projet déjà existant n'aboutit pas au remplacement du projet par une nouvelle architecture.

Notre proposition : Nous proposons de créer une seule architecture par projet. Lors de la création d'une nouvelle architecture dans un projet déjà existant, l'utilisateur sera demandé par un message box s'il voulait remplacer le projet existant par la nouvelle architecture.

2.5.9 *E9-DA:* Delete Architecture

Avant amélioration: Dans le système, la fonction de l'effacement de l'architecture n'existait pas.

Notre proposition : Nous proposons d'ajouter une fonctionnalité qui supprime totalement une architecture.

2.5.10 *E10-DC:* Delete Configuration

Avant amélioration : De même pour une configuration, le système n'assure pas son effacement.

Notre proposition : Nous proposons d'ajouter une fonctionnalité qui supprime totalement une configuration.

2.5.11 *E11-CC*: Create Configuration (under file)

Avant amélioration : On ne peut pas créer une configuration avant d'ouvrir une architecture au préalable.

Notre solution : Nous proposons d'ajouter un *shortcut* qui permet de créer une configuration sans avoir à ouvrir une architecture au préalable.

2.5.12 Tableau récapitulatif des améliorations dans *Moval-Tool*

Table 2.2 - Tableau Récapitulatif de l'analyse des améliorations *Enhancements* dans *Moval-Tool*.

Enhancement ID	Enhancement Title	Enhancement Description
E1-ADV	Architecture Designer Validation	Architecture should be validated after construction
E2-ICAV	Enable Inactive Check Boxes when Architecture Visualization	Enable Check Box functionalities in matrix view
E3-ECF	Enable Create Configuration Functionality	Implement Create configuration function
E4-OC	Open Configuration	Architect should open a configurations either in view mode or in edit mode
E5-AORO	ADP Wizard is an Option Rather than Obligation	Architect should choose between *Ad-hoc* mode or *Adp-assisted* mode

E6-IVAW	Iterations View in *ADP-Wizard*	Iterations of a phase should be shown in the progress bar below every activity.
E7-SA	Save As	
E8-AC	Architecture Creation	Replace existent project by the new architecture
E9-DA	Delete Architecture	Architect could Delete architecture totally
E10-DC	Delete Configuration	Architect could Delete configuration totally
E11-CC	Create Configuration (under file)	Architect doesn't have to open an architecture in advance to create a configuration

2.6 Conclusion

Dans ce chapitre, nous avons présenté le cahier des charges de notre projet. Il s'agit de résoudre des erreurs (bugs) soulevées lors de nos différents tests sur le *Moval-Tool*. Aussi, il s'agit d'apporter des améliorations sur cet outil. Les deux tableaux précédents résument le travail que nous devons mener. Dans le chapitre suivant, nous dégageons les spécifications de cet outil. Nous représentons le diagramme de cas d'utilisation de *Moval-Tool*.

Chapitre 3: Le scénario du système

3.1 Introduction

Dans ce chapitre nous allons dégager les différents scénarios du système qui sont construites après l'établissement du cahier des charges. Nous présentons aussi quelques cas d'utilisation avec des détails textuels tout en utilisant le modèle de ***Cockburn***.

3.2 Le diagramme de Cas d'utilisation

Le diagramme de cas d'utilisation [7] du système est présenté dans la figure 2.1 en fonction de l'acteur du système : **l'architecte**.

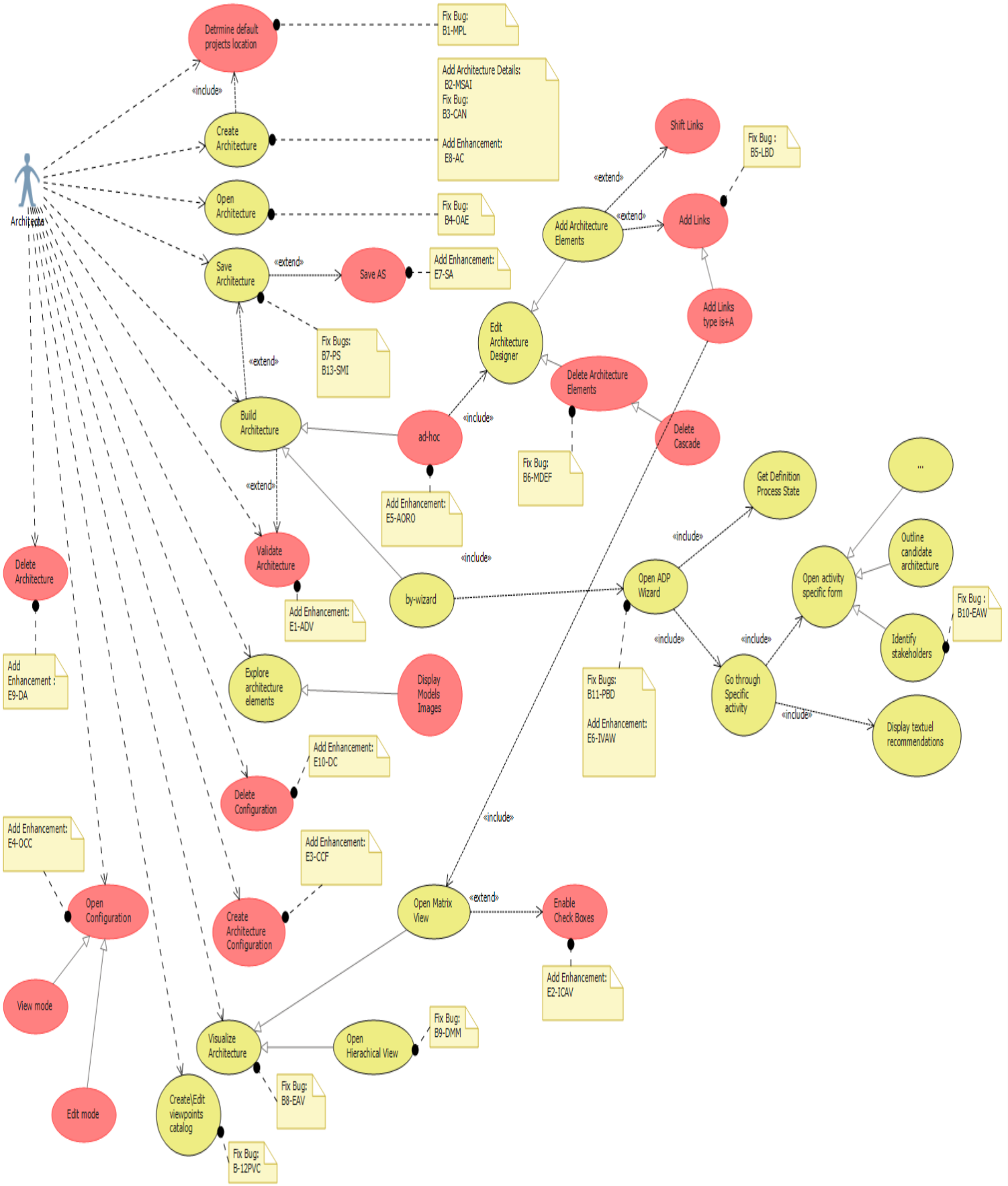


FIGURE 3.1 – Le diagramme UML des cas d'utilisation de *MoVAL-Tool*.

Cas d'utilisation déjà implémenté

Cas d'utilisation que nous avons ajouté

3.3 Les cas d'utilisation revisités ou ajoutés

Dans ce qui suit, nous détaillons les cas d'utilisation sur lesquels nous avons travaillé soit pour rectifier certaines erreurs, soit pour les implémenter complétement. Nous utilisons le modèle de ***Cockburn*** inventé par `Alistair Cockburn` qui aide à la rédaction efficace des cas d'utilisation.

3.3.1 Create Architecture

Les fixations des bugs ayant les codes *B2-MSAI* et *B3-CAN*, et l'amélioration dont le code *E8-AC* ajoutés au cas d'utilisation ***create architecture*** sont illustrés dans le tableau 3.1.

Numéro	**01**	
Nom	Create architecture	
Précondition	Accès au système	
Post conditions	Le nom de l'architecture, la location de l'architecture, le nom du système, le nom de l'architecte et le mode de la création de l'architecture (*Ad-hoc*, *Adp-assisted*) sont connus	
Acteurs primaires	Architecte	
Scénario Principal	**Step**	**Action**
	1-	L'acteur saisit le nom du projet, le nom l'architecture, sa location, la version, le nom du système et le nom de l'architecte ainsi que le mode de création de son architecture
	2-	Si le projet déjà existe dans la location déterminée le système informe l'acteur
	3-a	L'acteur insiste, le projet sera remplacé et l'architecture sera crée
	3-b	L'acteur annule la création

	4-a	Si le mode choisi est *Adp-assisted* une page *ADP-Wizard* sera ouverte
	5-a	Si le mode choisi est *Ad-hoc* une page de l'éditeur de l'architecture sera ouverte

Table 3.1 Description du cas d'utilisation ***create architecture***

3.3.2 Create architecture configuration

La création d'une configuration est une amélioration ajoutée au système ayant le code *E3-ECF*. La description du cas d'utilisation ***create architecture configuration*** est illustrée dans le tableau 3.2.

Numéro	**02**	
Nom	Create architecture configuration	
Précondition	Accès au système	
Post conditions	La configuration est créée et sauvegardée dans fichier XML. Le nom, la date de création, la description et les stakeholders de cette configuration sont connus, ainsi que les modèles qui 'y appartiennent	
Acteurs primaires	Architecte	
Scénario Principal	**Step**	**Action**
	1-	L'acteur clique « *create configuration* »
	2-	L'acteur clique « *Browse »* pour déterminer l'architecture de cette configuration
	3-	L'acteur saisit le nom de sa configuration, sa description etc...
	4-	L'acteur clique « *Next* »
	5-	La page de la visualisation hiérarchique est ouverte
	6-	L'acteur choisit les modèles à ajouter à son architecture

	7-	L'acteur clique « *Visualize it now* » et visualise les modèles choisis dans sa configuration
	8-	L'acteur clique « *Finish* »

Table 3.2 Description du cas d'utilisation ***create architecture configuration***

2.3.3 Build Architecture ad-hoc

La construction de l'architecture dans le mode *ad-hoc* est une amélioration ajoutée au fonctionnement du système ayant le code *E5-AORO*. Le tableau 3.3 décrit le cas d'utilisation ***build architecture ad-hoc***

Numéro	0**3**	
Nom	Build architecture ad-hoc	
Précondition	Une architecture créée dans le mode ad-hoc est ouverte	
Post conditions	L'acteur est capable d'ajouter des **liens architecturaux** à son architecture. L'acteur peut **supprimer** des éléments. L'acteur peut **valider** son architecture. L'acteur peut sauvegarder **sans aucun problème**	
Acteurs primaires	Architecte (ad-hoc)	
Scénario Principal	Step	Action
	1-	L'acteur clique «*architecture elements*»
	2-	L'acteur clique sur « *designer* »
	3-	L'acteur ajoute les éléments nécessaires à son architecture
	4-	L'acteur supprime les éléments erronés de son architecture
	5-	L'acteur clique sur « *Validate* » pour valider son architecture

	6-	L'acteur décide s'il veut continuer à éditer son architecture ou s'il voudrait la sauvegarder auquel cas il clique « *Save* »

Table 3.3 Description du cas d'utilisation ***build architectue ad-hoc***

3.3.4 Add links

Le cas d'utilisation ***add links*** que nous avons ajouté dans le but de fixer un bug ayant le code *B5-LBD,* est illustré dans le tableau 3.4.

Numéro	**04**	
Nom	Add links	
Précondition	La page de l'éditeur d'une architecture illustré dans la figure 1.6 est ouverte	
Post conditions	Des liens seront ajoutés aux différents éléments de l'architecture	
Acteurs primaires	Architecte	
Scénario Principal	Step	Action
	1-a	L'acteur clique sur l'onglet « *Designer* » s'il veut ajouter des liens de type *(Rlink, How To)*
	1-b	L'acteur clique sur l'onglet Matrix s'il veut ajouter des liens de type *(Correspondance, More Achieved)*
	2-	L'acteur clique sur le lien qu'il voudrait ajouter à son architecture contenue dans la barre d'outils de la figure 1.6
	3-	L'acteur clique sur les éléments de l'architecture (*AL, DL ou Model*) qu'il voudrait mettre en relation
	4-a	Si l'acteur a bien choisi les éléments à relier, le lien est ajouté à l'éditeur sans aucun problème
	4-b	Sinon, l'acteur sera notifié et le lien ne sera pas ajouté

	5-a	L'architecte ajoute des propriétés (une description et un label sémantique) au lien ajouté
	6-a	L'acteur clique « *Save* »
	7-a	le lien sera ajouté à l'architecture s'il est vérifié par le système

Table 3.4 Description de cas d'utilisation ***add links***

3.5. Delete architecture elements

La description du cas d'utilisation ***delete architecture elements*** que nous avons ajouté dans le but de fixer le bug référencé par le code *B6-MDEF*, est illustrée dans le tableau 3.5.

Numéro	**01**	
Nom	Delete architecture elements	
Précondition	La page de l'éditeur d'une architecture illustrée dans la figure 1.6 est ouverte	
Post conditions	Les éléments de l'architecture seront supprimés	
Acteurs primaires	Architecte	
Scenario Principale	Step	Action
	1-	L'acteur clique avec le bouton droit sur l'élément qu'il désire supprimer
	2-	Si la suppression n'est pas vérifiée par la règle RVAL5 (voir section 2.2), le système informe l'acteur que l'élément n'est pas vide (i.e. il contient des éléments)
	3-a	L'acteur force la suppression
	4-a	Une suppression par cascade aura lieu
	3-b	L'acteur annule la suppression

Table 3.5 Description de cas d'utilisation ***delete architecture elements***

3.4 Conclusion

Dans ce chapitre, nous avons présenté le scénario du système, les diagrammes de cas d'utilisation et les cas d'utilisation revisités ou ajoutés suivant le modèle *Cockburn*. Dans le chapitre suivant nous dégageons la vue logique du système en utilisant les digrammes UML de classe et de séquence.

Chapitre 4 : La vue Logique du système

4.1 Introduction

La vue logique permet d'identifier les différents éléments et mécanismes du système à réaliser. Elle exprime comment les différentes couches du logiciel interagissent. Dans ce chapitre, nous la présentons en utilisant les différents diagrammes UML [6] tel que les diagrammes de classe, de séquence et du composant.

4.2 Le diagramme de classe simplifié de la couche métier

Notre projet est structuré en trois couches. (1) La couche métier appelée *MovalProtoypeBusinessRules*, (2) la couche interface graphique dénotée *MovalPrototypeUI* et (3) une couche de librairies extérieures dénotée *Externals*. Dans la figure 4.1, nous présentons le diagramme de classe [7] simplifié de la couche *MovalProtoypeBusinessRules*.

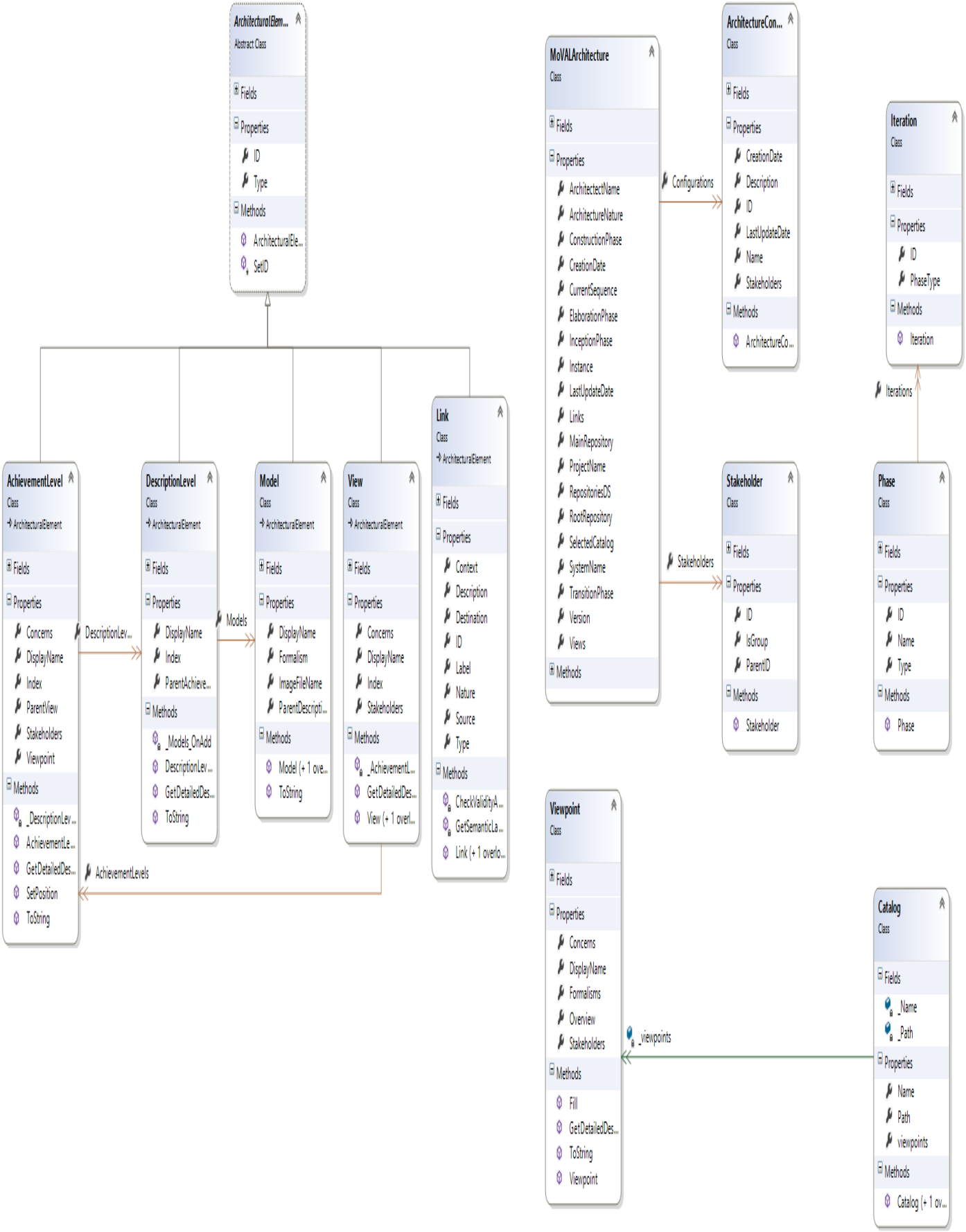


Figure 4.1 Le diagramme de classe simplifié de la couche `MovalPrototypeBusinessRules`

4.3 Les diagrammes de séquence

Afin de présenter les interactions entre les acteurs et le système, nous présentons dans la figure 4.2 le diagramme de séquence [7] associé au scénario ***Add Link*** et dans la figure 4.3 celui associé su scénario ***Create Configuration***.

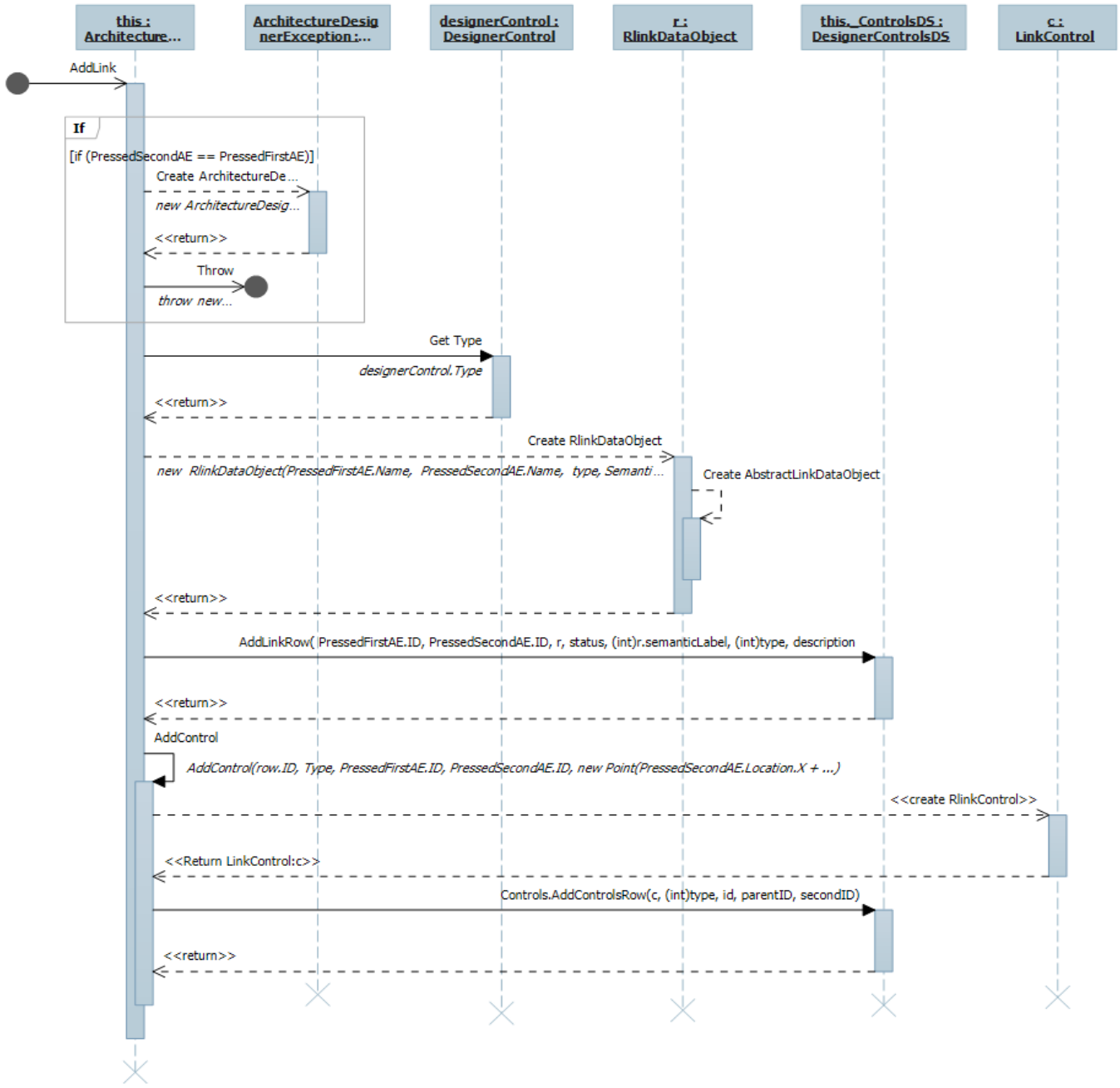


Figure 4.2 Le diagramme de séquence du scénario « ***Add Link*** »

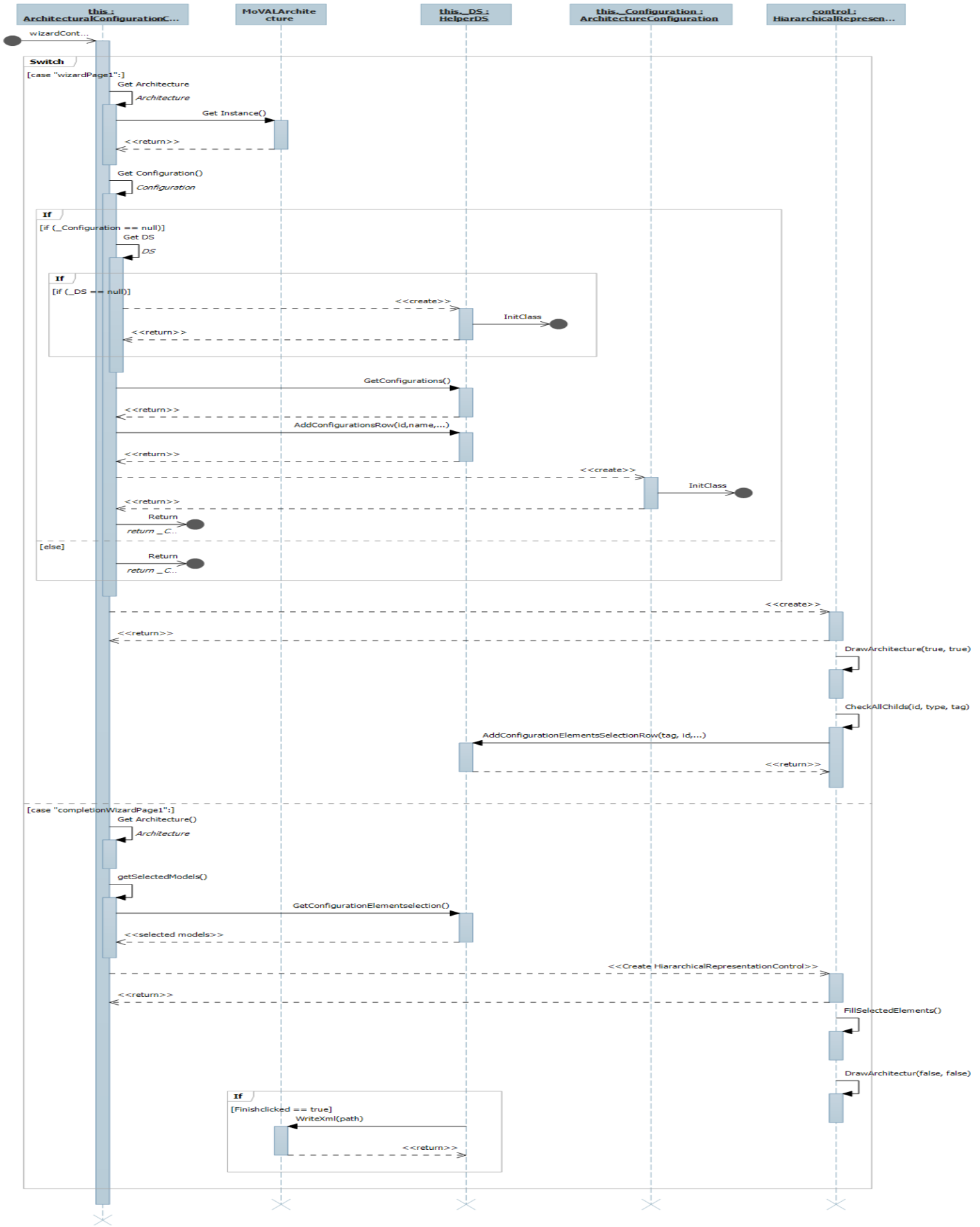


Figure 4.3 Diagramme de séquence du scénario ***« Create Configuration »***

4.4 Le diagramme de composant

Dans cette partie, nous établissons le diagramme de composant [8], illustré dans la figure 4.4, qui montre les dépendances entre les composants principales du système dénotées dans la section 4.2. Ainsi, nous présentons dans ce diagramme l'interaction de ces constituants avec des structures de base de données locales qui lisent et écrivent agilement dans un document XML où se fait le stockage permanent des données.

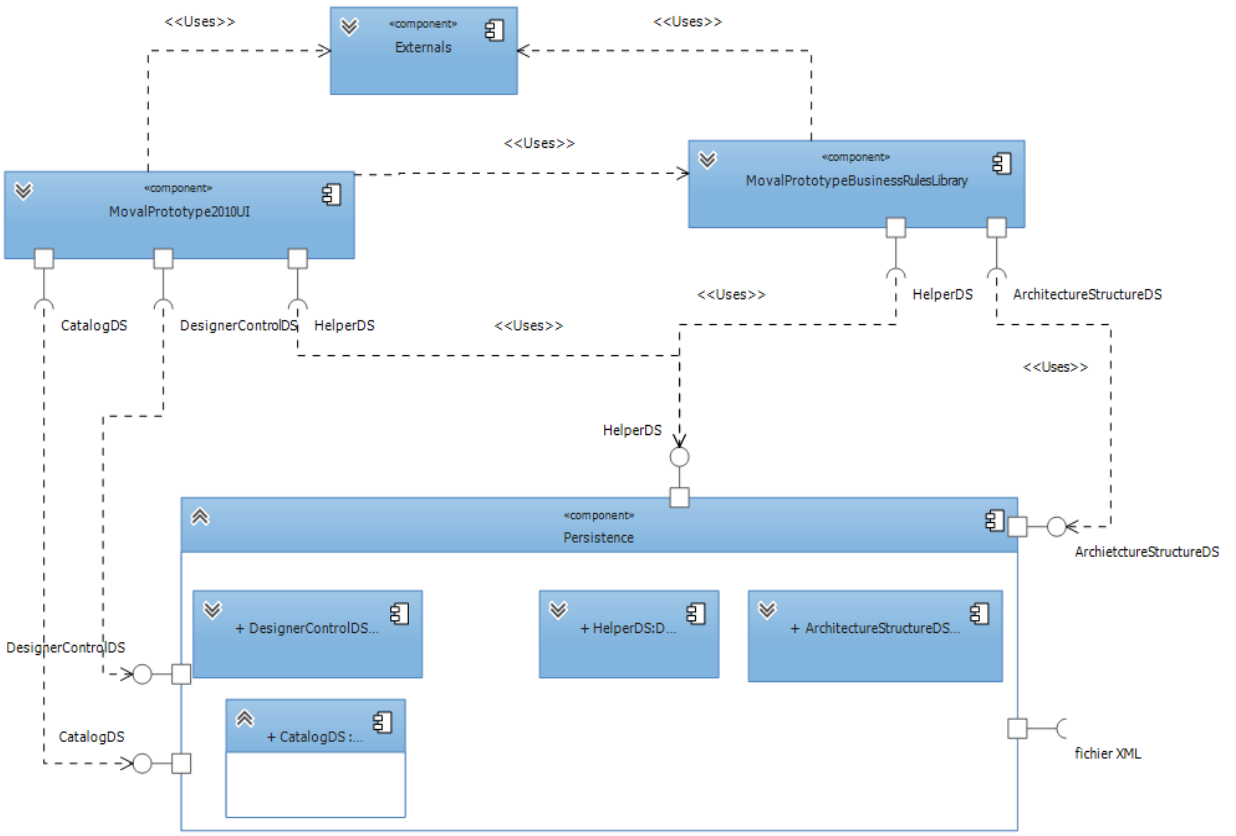


Figure 4.4 Diagramme de composants du système

4.5 Conclusion

Dans ce chapitre, nous avons dégagé les différentes classes UML. Dans le chapitre suivant, nous présentons la vue persistance de notre système.

Chapitre 5 : Persistance de données

5.1 Introduction

Dans ce chapitre, nous dégageons la vue persistance du système. Nous présentons les schémas relationnels des bases de données créés et utilisées par notre logiciel.

5.2 Les schémas relationnels

Les informations utilisées dans l'application sont sauvegardées essentiellement sous la forme de plusieurs fichiers XML. Lors de l'exécution, des structures des bases de données locales seront créées. Nous les détaillons dans les paragraphes suivants.

5.2.1 CatalogDS

Lors de la création d'un nouvel catalogue, une base de données dénotée `CatalogDS` sera générée par le système. Elle est composée de deux entités : (1) `viewPoint` et (2) `CatalogInfo`. Le schéma relationnel de cette structure de base de données est illustré dans la figure 5.1.

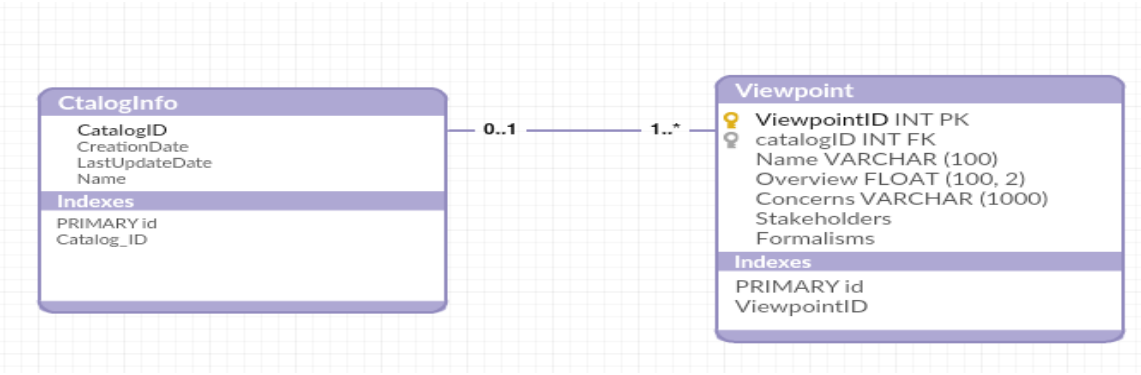


Figure 5.1 Le shéma relationnel de la base de données `CatalogDS`

5.2.2 DesignerControlDS

Le schéma relationnel de la base de données dénotée `DesignerControlDS` est illustré dans la figure 5.2. Cette base de données contient Sept entités, elle sera créée et remplie lors de la construction de l'architecture.

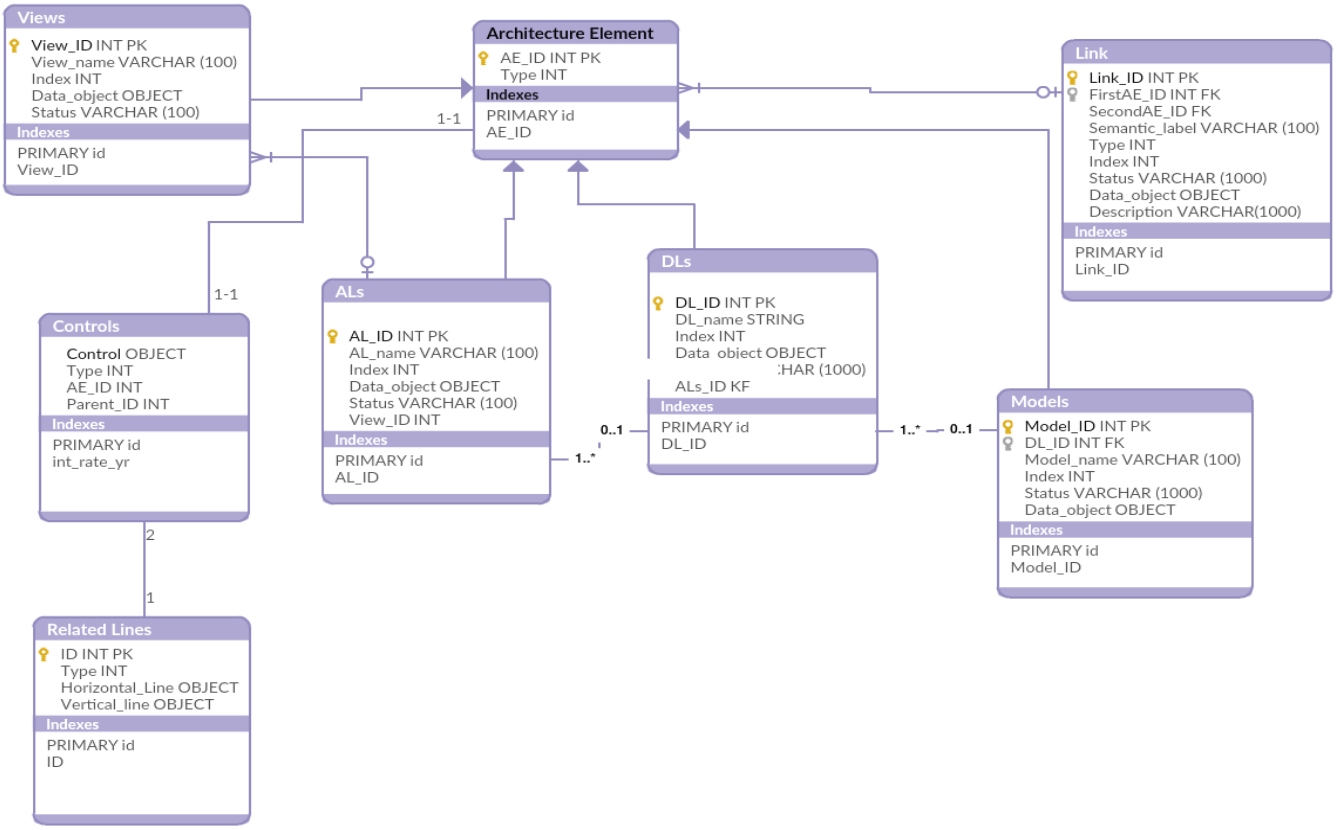


Figure 5.2 Le schéma relationnel de la base de données *DesignerControlDS*

5.3 HelperDs

Le schéma relationnel de la base de données dénotée *HelperDS* est illustré dans la figure 5.3. Dans cette base de données les deux entités initiales seront remplies lors de la création d'une configuration.

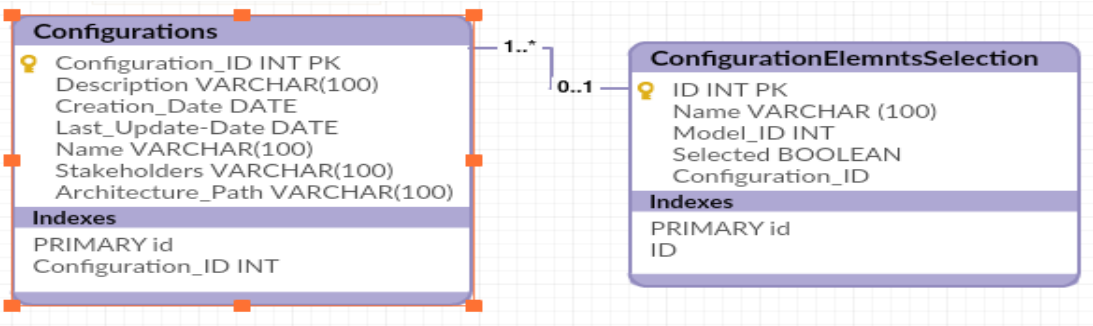


Figure 5.3 Le schéma relationnel de la base de données- *HelperDS*

5.4 ArchitectureStructureDS

La structure de la base de données dénotée *ArchitectureStructureDS* sera généré et remplie juste après le chargement d'une nouvelle architecture. Elle lit directement les informations stockées dans un fichier XML. Elle est composée de huit entités illustrées dans la figure 5.4.

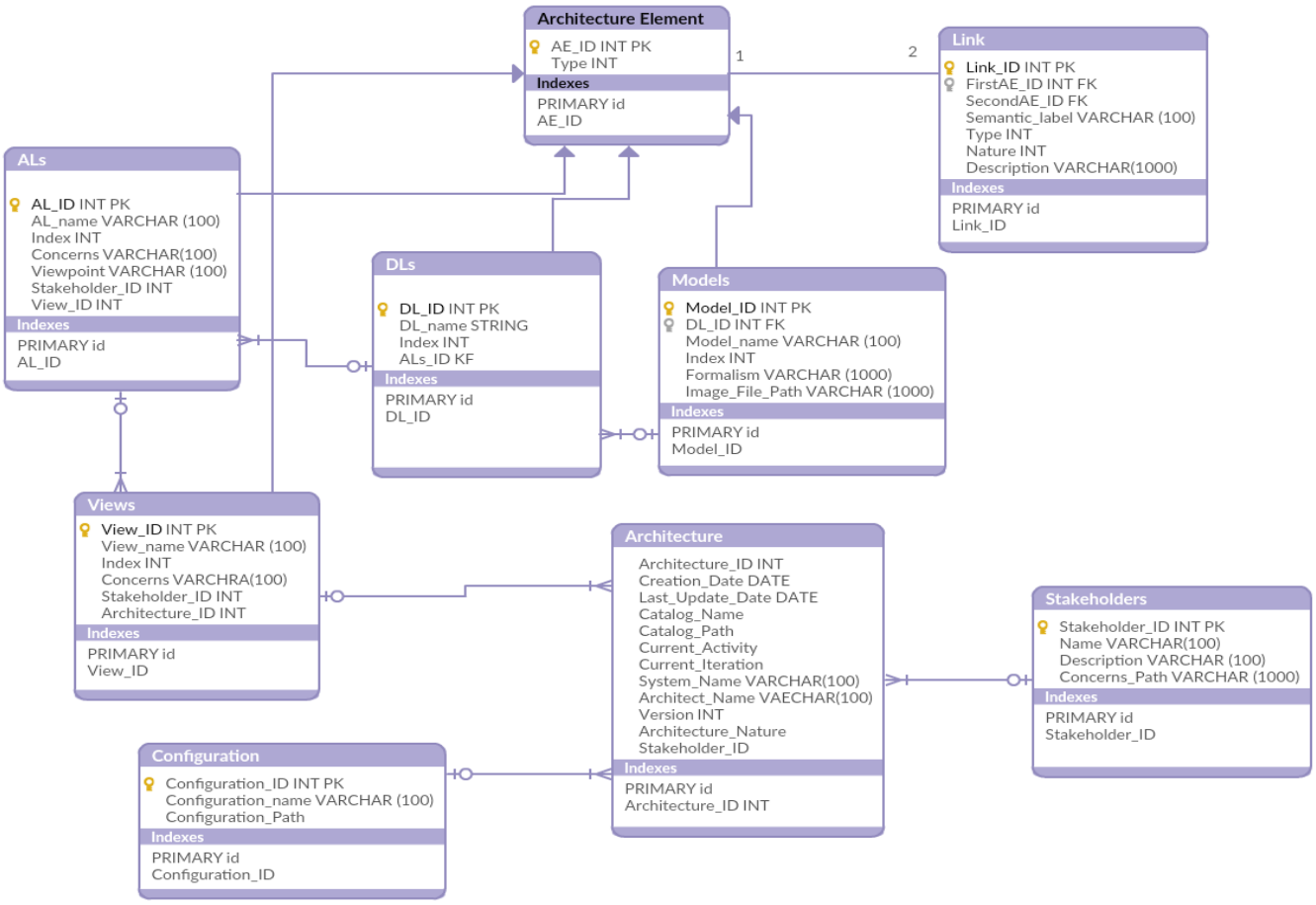


Figure 5.4 La base de données *ArchitectureStructureDataSet*

5.3 Conclusion

Dans ce chapitre, nous avons présenté les différentes entités de la base de données. Dans le chapitre suivant, nous dégageons la vue physique du système. Les avantages de ce stage et nos perspectives.

Chapitre 6 : La vue « Physique du système » et Perspectives

6.1 Introduction

Dans ce chapitre, nous présentons la vue physique du système. Les différentes interfaces graphiques lors de l'exécution. Ainsi, nous dégageons les avantages de ce stage, la conclusion et nos perspectives.

6.2 GUI du système

L'interface graphique de l'outil est présentée en quelques captures d'écran montrant les principales fonctionnalités de l'acteur après que nous corrigeons les erreurs et nous ajoutons quelques améliorations au système.

6.2.1 Le lancement du système

La première fois le système est lancé sur une machine, une nouvelle fenêtre illustrée dans la figure 6.1 lui demande de déterminer l'espace de ces projets. Elle prend par défaut le directoire « *My Documents* » relatif à cette machine.

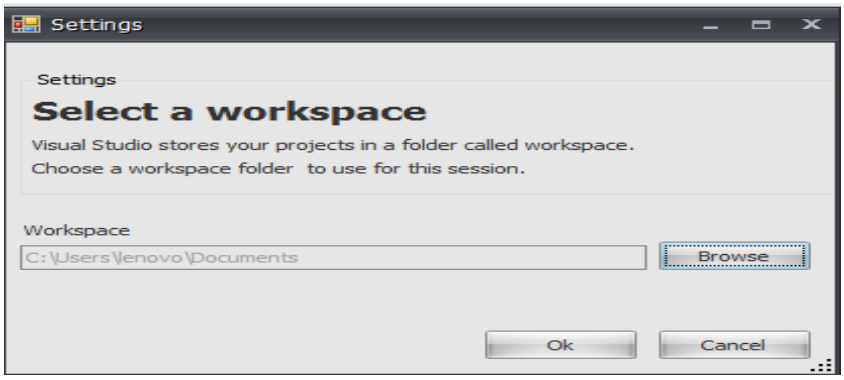


Figure 6.1 Location du projet

L'architecte a toujours accès sur cette fenêtre tout en cliquant sur le sous- menu « *Settings* » pour changer l'espace du travail.

6.2.2 La création d'une architecture

Afin de créer son architecture, l'architecte passe à l'interface graphique illustrée dans la figure 6.2. Il saisit ses informations et le mode de sa création.

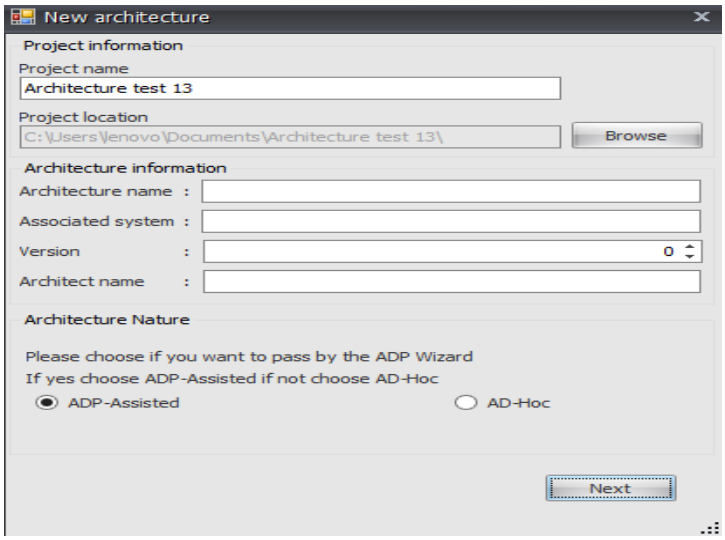


Figure 6.2 GUI pour créer une nouvelle architecture

Lorsque qu'il saisit un nom du projet déjà existant dans la location qu'il saisit, l'architecte sera notifié par le système comme le montre la figure 6.3.

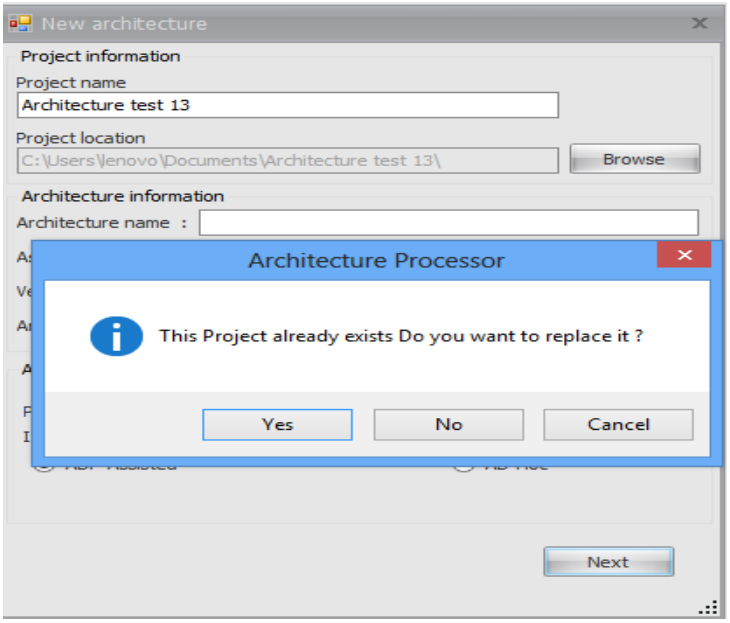


Figure 6.3 GUI pour notifier l'acteur lors de la création de l'architecture

6.2.3 L'éditeur de l'architecture

Lorsque l'architecture est créée en mode *Ad-hoc*, l'architecte clique « *Architecture elements* » puis il clique « *architecture designer* » et passe à l'éditeur de l'architecture. Illustré dans les figure 6.4 et 6.5.

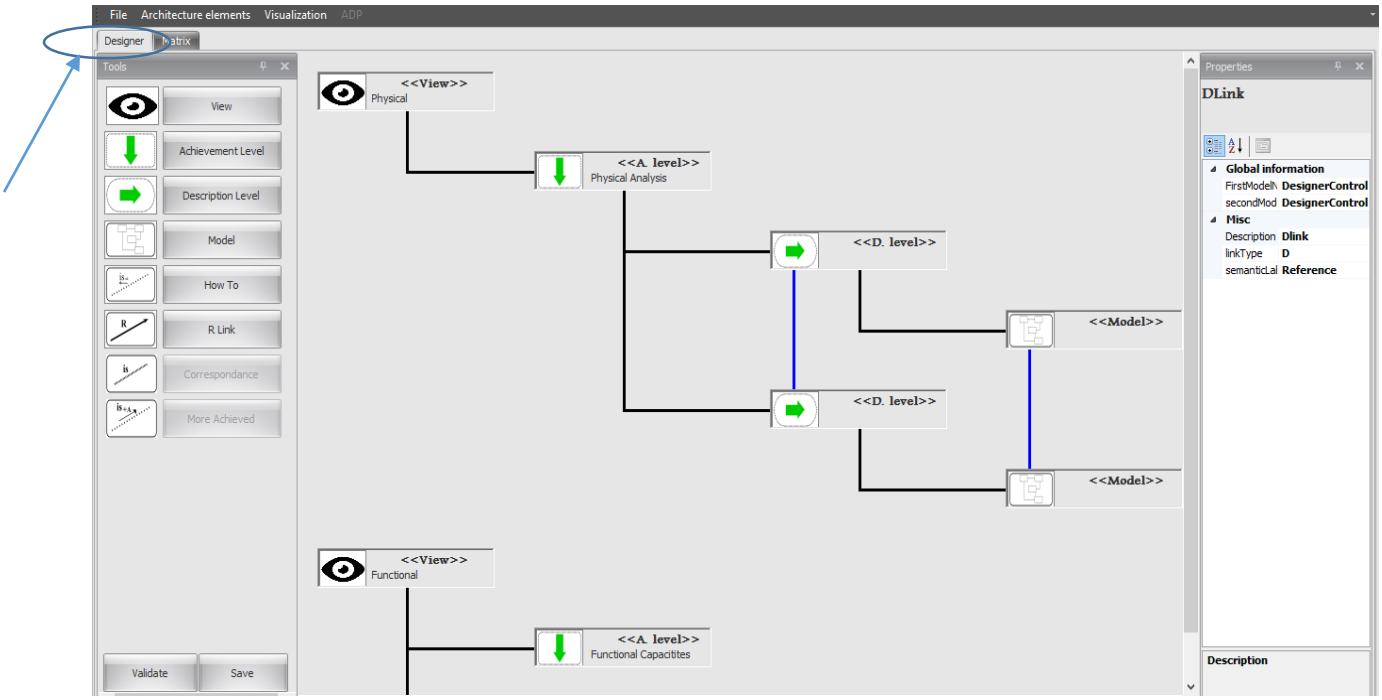


Figure 6.4 GUI pour l'éditeur de l'architecture avec l'onglet *designer* sélectionné

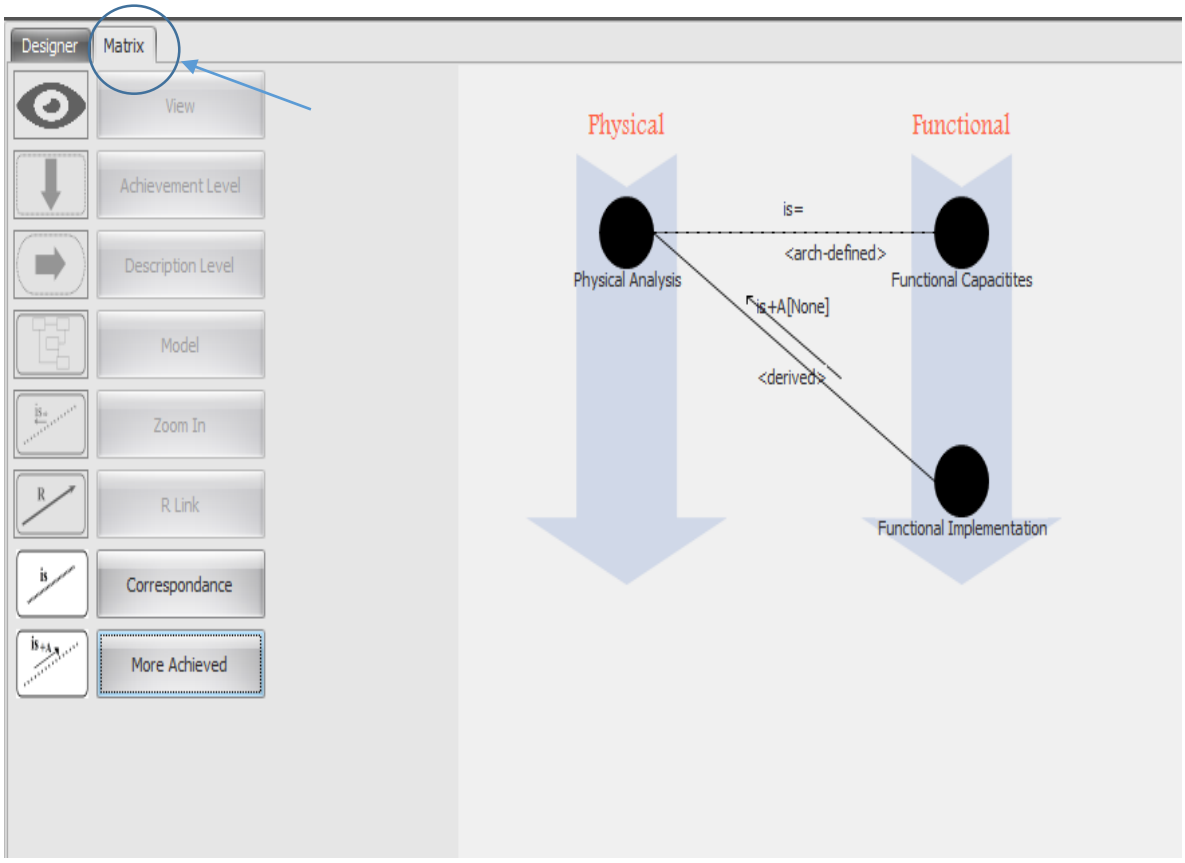


Figure 6.5 Editeur de l'architecture avec l'onglet *Matrix* sélectionné

6.2.4 Visualisation hiérarchique de l'architecture

L'architecture est visualisée soit en mode hiérarchique illustré dans la figure 6.6, soit en mode matriciel comme le montre la figure 6.7.

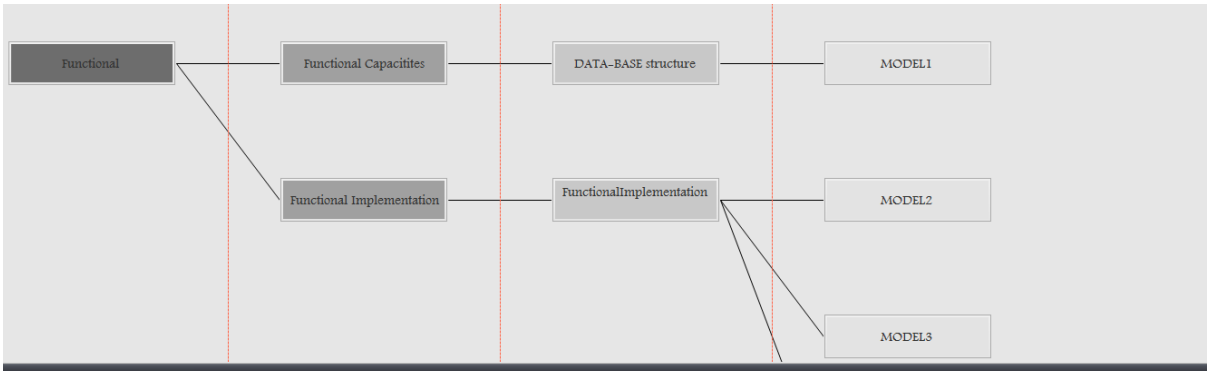
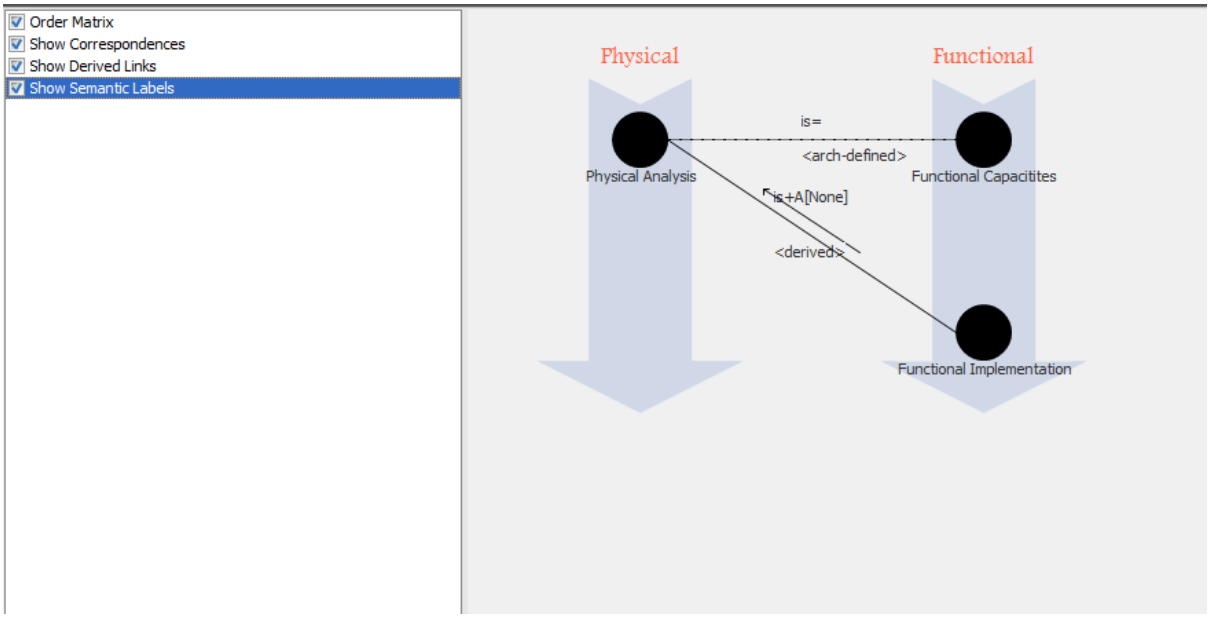


Figure 6.6 GUI pour la visualisation hiérarchique de l'architecture



6.7 GUI pour la visualisation matricielle de l'architecture

6.2.5 La créer d'une configuration

L'architecte clique « *Create configuration* » pour passer à la page illustrée dans la figure 6.8.

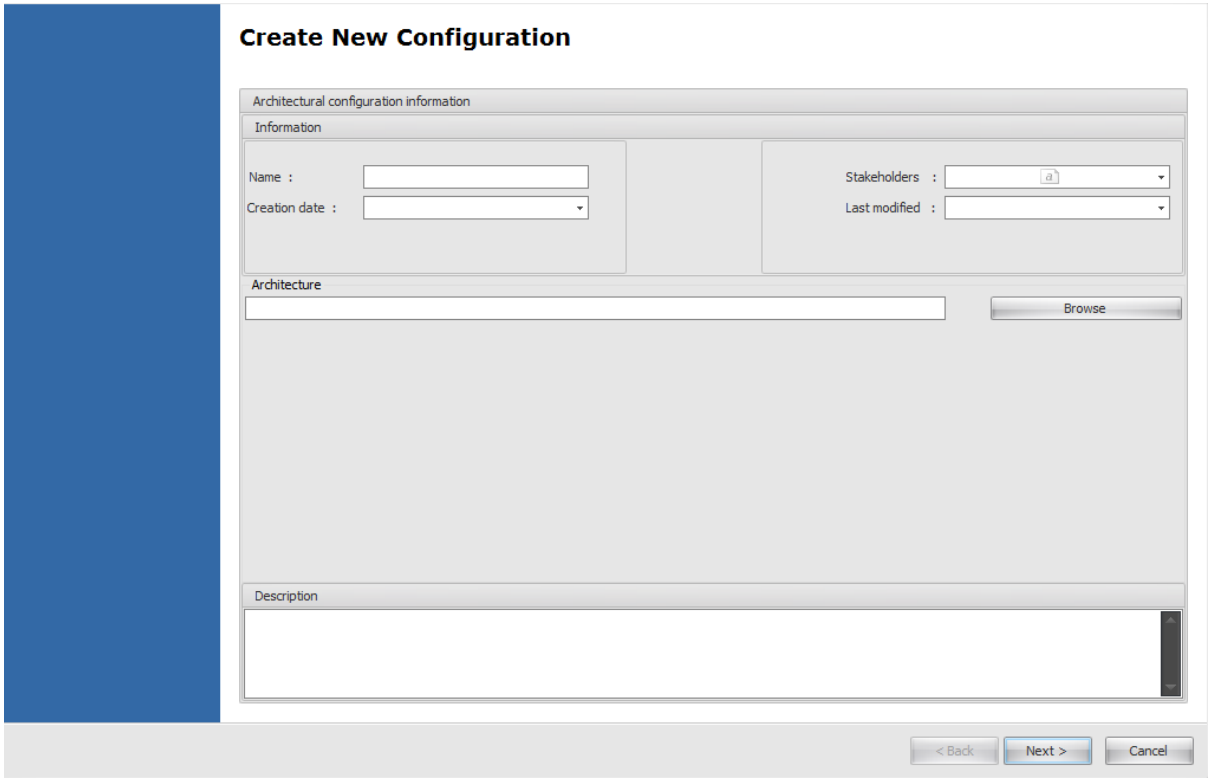


Figure 6.8 Créer une configuration

L'architecte clique sur « *Next* » pour choisir les éléments à ajouter à son configuration comme le montre la figure 6.9.

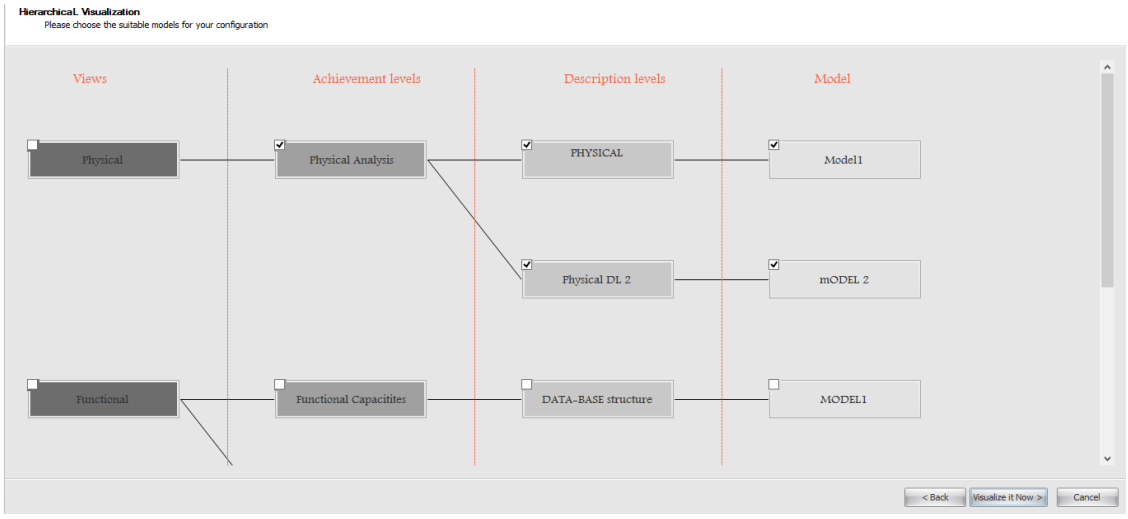


Figure 6.9 GUI pour choisir les modèles d'une configuration

L'architecte clique sur « *Visualize it now* » pour visualiser sa configuration comme le montre la figure 6.10.

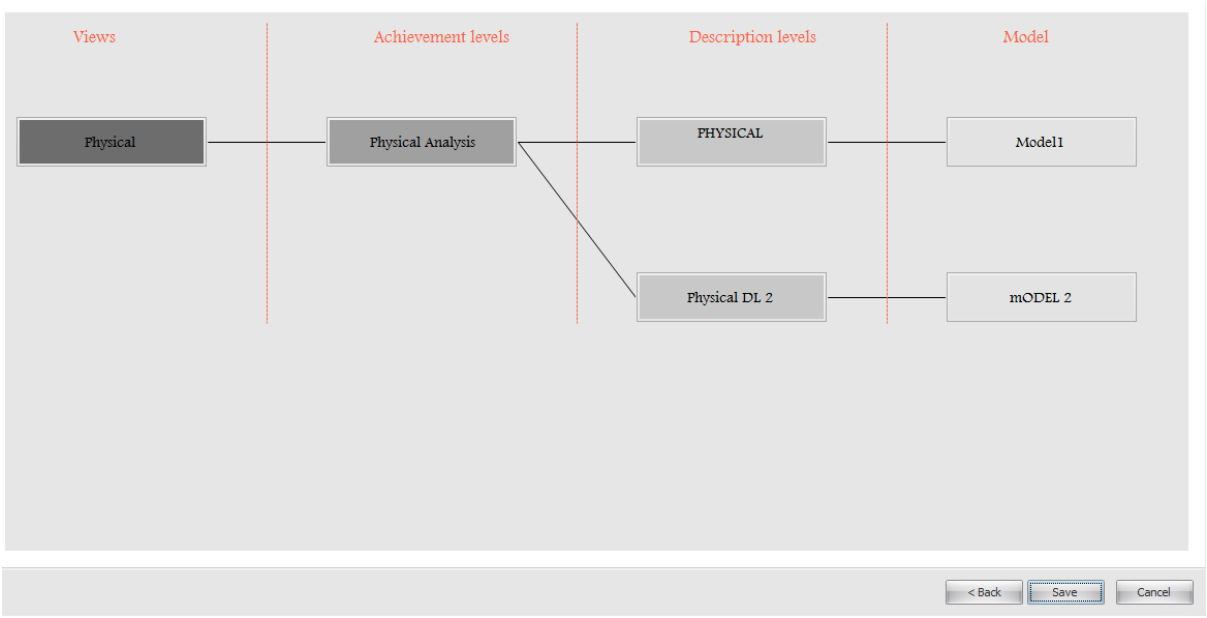


Figure 6.10 GUI de la configuration crée par l'architecte

6.2.6 *ADP-Wizard*

L'*ADP-Wizard* du système est illustré dans la figure 6.11.

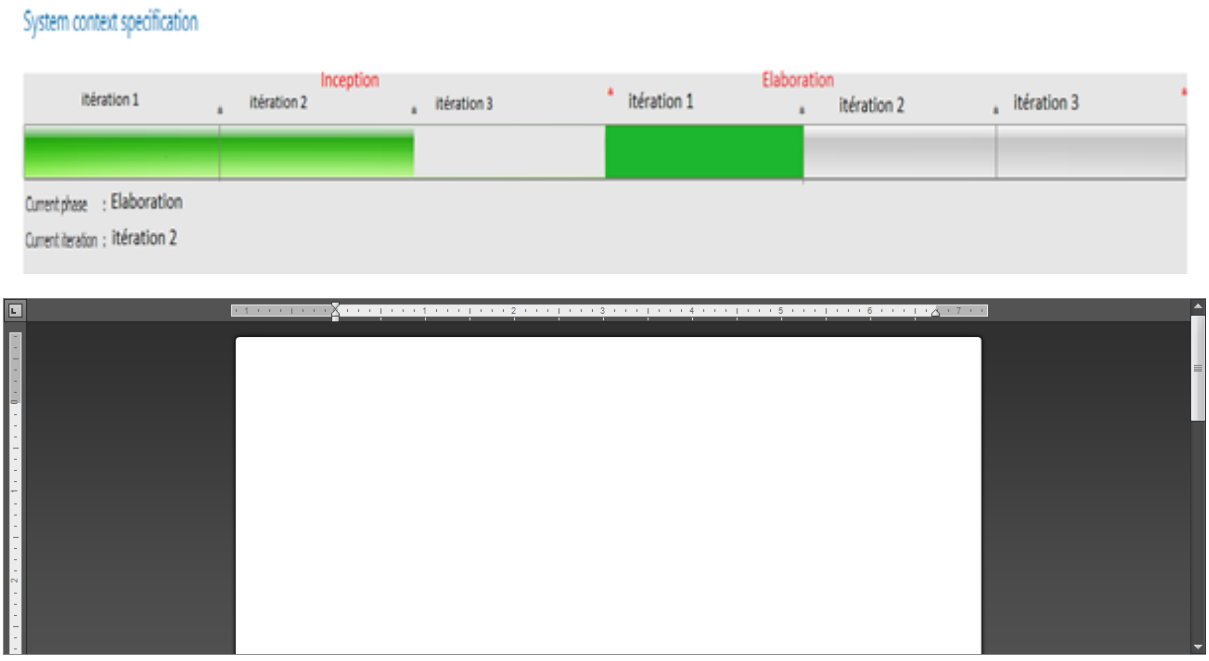


FIGURE 6.11 GUI pour la page *ADP-Wizard*

6.2.7 Visualisation arborescente de l'architecture

L'architecte clique sur « *architecture tree view* » pour explorer les différents constituants de l'architecture sous une forme arborescente comme illustré dans la figure 6.12.

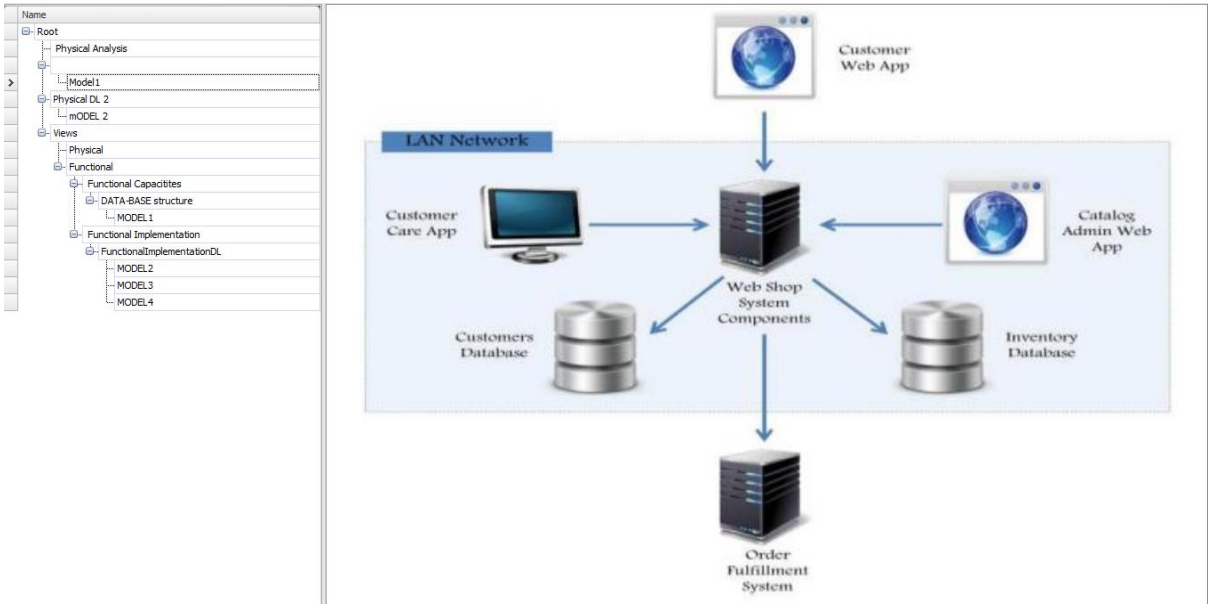


FIGURE 6.12 GUI pour la visualisation arborescente de l'architecture

6.3 Conclusion

Ce stage, objet de ce rapport, que j'ai effectué à la faculté des sciences m'a permis de consolider mes connaissances acquises à l'université surtout celles concernant la plateforme .Net. Il m'a permis d'apercevoir la différence entre le monde théorique et les difficultés rencontrées dans la mise en pratique.

Le travail que j'ai effectué dans le cadre de ce stage a consisté d'abord (1) à tester l'outil *Moval-Tool* en profondeur, (2) à dégager les erreurs et les améliorations que nous jugeons nécessaires, (3) à intégrer notre proposition dans l'outil *Moval-Tool*. Désormais, cet outil est à la main des architectes qui veulent appliquer *Moval* pour la construction de leurs architectures avec plus de fonctionnalités et de fiabilité.

Malgré que le travail à la faculté ne m'ait pas lancé dans un milieu professionnel tel que dans une entreprise, ce stage m'a donné beaucoup d'opportunité. En effet, il m'a permis de tester mes compétences, de mettre à l'épreuve et démontrer mon savoir-faire, d'activer et développer un apprentissage autonome. D'autre part, il m'a donné la possibilité d'étudier les nouveaux horizons de recherche en matière des architectures des logiciels; ce qui est un complément à mes connaissances acquises durant mes études.

Quant à mes perspectives, je suggère que *Moval-Tool* soit développé sur plusieurs plateformes par exemple sur le *Mac Os* afin d'être diffusé sur une échelle plus grande.

Références

[1] KHEIR, Ahmad. MoVAL: *Modélisation multipoints de vue/ multi-granularités d'architectures logicielles* [thèse en ligne]. 148 p. Thèse : Informatique et applications : Laboratoire d'informatique de Nantes-Atlantique (LINA) : 2014. Disponible sur: https://halshs.archives-ouvertes.fr/tel-01146343/documents

[2] KHIER, A., NAJA H., OUSSALAH, M. *MoVAL, a new approach to software architecture and its comparison with existing views based approaches in software engineering* [en ligne]. InfoComp Journal, 13(1), 26-37, 2014. Disponible sur : <http://www.dcc.ufla.br/infocomp/index.php/INFOCOMP/article/view/20/7>

[3] KHIER, A., NAJA H., OUSSALAH, M., TOUT, K. *From Viewpoints and Abstraction Levels in Software Engineering Towards Multi-Viewpoints/Multi-Hierarchy in Software Architecture* [en ligne]. The Eighth International Conference on Software Engineering Advances, Oct 2013, Venice, Italy. pp.478, 2013. Disponible sur: <http://hal.archives-ouvertes.fr/hal-01006222/document>

[4] ROZANSKI Nick, WOODS Eoin. *Software systems architecture: working with stakeholders using viewpoints and perspectives.* Addison-Wesley, Reading, 2011

[5] Introducing .Net Core. *msdn.com*[en ligne]. 4-Dec-2014. Disponible sur: <http://blogs.msdn.com/b/dotnet/archive/2014/12/04/introducing-net-core.aspx>

[6] Grady Booch, James Rumbaugh, Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, Reading, Mass., 1999

[7] UML Diagram Types with Examples for Each Type of UML Diagrams. *creately.com: edition électronique* [en ligne]. 2 February 2012. Disponible sur: < http://creately.com/blog/diagrams/uml-diagram-types-examples/#PackageDiagram>

[8] UML 2 Tutorial - Component diagram. *SparxSystems.com : édition électronique* [en ligne]. 18-June-2015. Disponible sur : <www.sparxsystems.com/resources/uml2_tutorial/uml2_componentdiagram.html>