SHADI LOGHMANI

Migration d'interfaces utilisateurs textuelles de systèmes patrimoniaux vers le Web

Mémoire présenté
à la Faculté des études supérieures et postdoctorales de l'Université Laval
dans le cadre du programme de maîtrise en informatique
pour l'obtention du grade de Maître ès Sciences (M. Sc.)

FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL
QUÉBEC

2013

Résumé

Les interfaces utilisateurs détiennent une importance non négligeable dans tout logiciel applicatif, car elles constituent le premier contact entre les utilisateurs et le logiciel. La migration d'interfaces utilisateurs textuelles de systèmes patrimoniaux vers le Web vise à remédier les problèmes liés à l'accès limité de tels systèmes tout en améliorant l'aspect et la convivialité de ces interfaces. Fréquemment, cela implique à la fois le changement de la plate-forme d'exécution - d'une plate-forme textuelle vers une architecture Web - et la refonte complète des interfaces vers une technologie de présentation Web.

Ce mémoire est une synthèse des méthodes et des techniques de migration d'interfaces utilisateurs textuelles vers le Web. Nous y abordons, dans un premier temps, les notions reliées aux systèmes patrimoniaux et au Web. Cela permet d'établir le contexte de notre étude à savoir l'environnement de départ et l'environnement cible dans une migration d'interfaces utilisateurs vers le Web. Dans un deuxième temps, nous exposons différentes techniques et approches de migration d'interfaces utilisateurs. Finalement, nous terminons notre présentation avec trois exemples de plates-formes de migration d'interfaces utilisateurs.

Abstract

User interfaces possess a significant importance in many software applications. They are the first contact between the user and the software and this contact is crucial in the process of familiarization with the application. The migration of character-based user interfaces of legacy systems aims to address the problems associated with limited access to such systems while improving the look and feel of the interfaces. Frequently, this involves both changing the execution platform - from a character-based platform to a Web architecture platform - and the complete reengineering of the user interfaces to a Web presentation technology.

This thesis is a summary of the methods and techniques of character-based user interfaces migration to the web. In the first place, we discuss the concepts related to legacy systems and the Web. This establishes the context of our study (i.e. the starting point and the target environment in a user interface migration). Next, we present different techniques and approaches to user interfaces migration. Finally, we conclude our presentation with three examples of user interfaces migration platforms.

Avant propos

Je tiens avant tout à exprimer ma profonde reconnaissance au professeur Béchir Ktari, qui a assuré la direction de mon travail de recherche. Son regard critique, sa curiosité intellectuelle et ses conseils précieux m'ont inspiré pour mener à terme cette recherche.

J'exprime ma gratitude aux professeurs Nadir Belkhiter et Bernard Moulin pour avoir accepté de lire et d'évaluer ce travail.

Un grand merci à Camille pour son aide lors de la rédaction du présent mémoire et tous mes amis et collègues du LSFM pour leur sourire et leur bonne humeur permanente.

Merci à Afshar pour son soutien inconditionnel et pour le souffle d'optimisme avec lequel il a su me propulser pour que je puisse donner le meilleur de moi-même. Merci à Kian pour sa patience et ses petits mots d'encouragements.

Finalement, je remercie mes chers parents bien-aimés, ceux à qui je dois tout.

À mon compagnon de vie, Afshar
«Il a été l'étincelle qui a allumé ce feu»

Table des matières

Table des figures

Introduction

Depuis une vingtaine d'années, l'industrie de l'information est dominée par l'introduction de nouvelles technologies informatiques ainsi que par leur évolution constante ; à tel point qu'aujourd'hui, ce sont Internet et plus particulièrement le Web qui dictent les besoins d'affaires.

Malgré la complexité du développement de systèmes pour le Web, du point de vue de l'utilisateur, l'accès aux différents services se fait par l'entremise d'une interface simple, populaire et économique, appelée navigateur.

Alors que le marché a un grand intérêt pour les technologies Web, certains grands fournisseurs de services, des gouvernements, des compagnies d'assurances et des institutions financières ont tendance à être adeptes de leurs vieux systèmes, dits patrimoniaux.

Les systèmes d'information patrimoniaux, hébergés généralement sur les ordinateurs centraux, exécutent littéralement les règles d'affaires pour lesquelles ils sont conçus. Les services offerts par de tels systèmes sont souvent fiables et satisfaisants. L'accès à leurs données est rapide, et ce, malgré un nombre élevé de connexions simultanées [66]. Cependant, contrairement aux interfaces utilisateurs[1] graphiques des applications Web qui sont conviviales et faciles à utiliser, les interfaces utilisateurs des systèmes patrimoniaux sont souvent textuelles (en anglais *character-based*), basiques, et offrent des fonctionnalités très limitées. Elles souffrent notamment des lacunes suivantes [23] :

- accès utilisateur limité - aujourd'hui, les organismes se voient dans l'obligation de fournir un accès externe de leurs services à leurs partenaires ou à leurs clients, via Internet ou via des réseaux intranet. Toutefois, la nature propriétaire des systèmes patrimoniaux fait en sorte que l'accès aux IUs dans un tel système est privé et se limite aux utilisateurs locaux ;

1. Afin d'alléger le texte, nous avons pris la liberté d'employer le terme « IU » dans ce document pour désigner l'interface utilisateur.

- manque de convivialité - les IUs textuelles des systèmes d'information patrimoniaux sont faibles en terme de capacités visuelles. Elles se composent, pour la plupart, de champs de saisie et d'étiquettes. L'utilisateur de tels systèmes est obligé de mémoriser toutes les données d'entrée du système et de les introduire à chaque utilisation. Les interfaces utilisateurs textuelles sont donc non intuitives et difficiles à apprendre ;

- navigation faible - dans les systèmes patrimoniaux, l'espace disponible pour afficher une IU textuelle se limite à la dimension restreinte de l'écran des terminaux : 24 lignes par 80 colonnes. Conséquemment, une IU textuelle offre un modèle de navigation faible et chronophage (de l'anglais *time-consuming*). À titre d'exemple : sur les IUs textuelles, la visualisation complète d'une longue liste de données implique plusieurs opérations de l'utilisateur, alors que dans un environnement graphique, il est possible d'afficher la même liste dans une seule page et de la visualiser à l'aide de la barre de défilement (de l'anglais *scroll bar*), intégrée dans l'environnement.

Les IUs détiennent une importance non négligeable dans tout logiciel applicatif. C'est essentiellement par la voie des IUs que les utilisateurs connaissent un logiciel et qu'ils apprennent à utiliser ses services. C'est pourquoi, aujourd'hui, on insiste de plus en plus pour que la convivialité des IUs soit prise en considération au moment du développement de tout logiciel et pour que les développeurs soient sensibilisés aux conséquences d'une mauvaise conception d'IUs.

Compte tenu de ces faits, les propriétaires des systèmes patrimoniaux manifestent de plus en plus leur intérêt à investir dans des solutions de migration vers des infrastructures Web. Ils désirent donc rendre disponibles leurs services via le Web et, en même temps, améliorer l'aspect et la convivialité des IUs de leurs systèmes, ce qui a comme effet d'attirer un plus grand nombre d'utilisateurs, de rester compétitifs sur le marché ou tout simplement de demeurer en affaires.

La migration de systèmes patrimoniaux a fait l'objet de plusieurs travaux de recherche qui ont permis de proposer des solutions, des techniques et une multitude d'outils d'automatisation pour le processus de telles migrations. Cependant, puisque le code des IUs occupe souvent une place importante - dans certains cas jusqu'à 50% [50] - dans le code d'un système patrimonial, une grande partie des efforts déployés dans une migration de systèmes, concerne le code des IUs. De plus, dans une telle migration, les IUs sont souvent les seuls composants qui font l'objet d'une refonte complète ou de modifications importantes. À cela s'ajoute l'architecture conceptuelle de la majorité

des systèmes patrimoniaux qui est semi-décomposable [2] et qui permet un minimum de séparation entre le code des IUs et le reste du système. Finalement, de nos jours, la séparation du code d'IUs des autres composants figure parmi les éléments clés dans le développement de logiciels. Ces considérations nous convainquent que la migration d'IUs peut et mérite d'être étudiée comme étant un processus indépendant dans la migration de systèmes.

Notre réflexion dans ce mémoire porte sur la migration des IUs des systèmes d'information patrimoniaux. Étant donné l'importance de la place que le Web occupe dans le domaine des affaires, nous nous intéressons plus particulièrement à la migration d'IUs vers le Web. Dans ce document, nous tenterons de faire une synthèse assez complète des solutions et des techniques existantes sur ce sujet. Pour ce faire, nous nous sommes référés à plusieurs travaux de recherche sur la migration de systèmes patrimoniaux et avons focalisé notre attention sur les discussions concernant les IUs.

Notons qu'une tranche importante des travaux de recherches dans le domaine de la migration de systèmes sont âgés de plus de 15 ans, l'époque où l'accessibilité des services via le Web ou via les réseaux interanet ne présentait pas encore un défi sérieux dans le domaine des affaires. Nous avons donc pris la liberté de prendre et d'adapter ces discussions au contexte du Web. Nous croyons que notre synthèse peut faire gagner du temps et de l'effort pour tous ceux qui s'intéressent à ce sujet et peut contribuer à établir une meilleure solution à l'avenir.

Le contenu du mémoire est réparti ainsi :

- le chapitre 1 (*Système d'information patrimonial*) comporte une étude non exhaustive des propriétés des systèmes d'information patrimoniaux : la définition de la notion de système patrimonial, deux types de programmes qu'on peut trouver dans un système patrimonial et trois types d'architecture conceptuelle d'un tel système. Nous y présentons également les activités de l'évolution qu'un système patrimonial subit pendant son cycle de vie. Nous espérons fournir, à travers ce chapitre, les informations dont le lecteur a besoin pour comprendre et pouvoir suivre nos discussions dans ce mémoire ;
- le chapitre 2 (*Web*) décrit l'historique de la création du World Wide Web (WWW) suivi de l'évolution du Web allant des simples pages Web aux applications Web complexes. Les différentes technologies de présentation Web sont également exposées ;
- le chapitre 3 (*Classement des techniques de migration d'IUs textuelles vers le Web*) présente deux classes d'approches de migration d'IUs : approches frontales et ap-

2. L'architecture conceptuelle de systèmes patrimoniaux sera traitée à la section 4.

proche de greffage. Quelques exemples de techniques de chaque classe d'approche y sont également exposés ;
- le chapitre 4 (*Refonte automatisée d'IUs textuelles*) énumère les deux étapes s'inscrivant dans le cadre d'une refonte d'IUs à savoir l'ingénierie-inverse et la conception descendante ;
- le chapitre 5 (*Plate-formes de migration d'IUs*) présente trois grands projets de migration d'IUs et des outils développés dans le cadre de ces projets. Nous terminons le chapitre en récapitulant quelques constats de la pratique de la migration de système et d'interfaces utilisateurs dans l'industrie ;
- dans la Conclusion, nous tentons de récapituler les notions vues dans ce mémoire ;
- finalement, un lexique est consacré à la présentation sommaire des principaux termes techniques utilisés dans ce mémoire.

Chapitre 1

Système d'information patrimonial

> We don't know where to GOTO, if we don't know where we've COME FROM.
>
> Anonymous

Dans ce mémoire, l'essentiel de notre recherche porte sur les IUs. Toutefois, l'étude d'IUs de systèmes d'information patrimoniaux impose une connaissance globale de tels systèmes et cette connaissance n'est généralement pas à la disposition de jeunes chercheurs qui désirent travailler dans ce domaine. Il nous paraît donc approprié de faire une étude des propriétés et des caractéristiques des systèmes patrimoniaux.

Dans ce chapitre nous définissons la notion de système d'information patrimonial. Cette définition sera suivie de la présentation de deux types de programmes - programmes de commande et programmes en-ligne - associés à de tels systèmes. Une attention spéciale est portée aux particularités des programmes en-ligne. Nous présentons ensuite les trois architectures logiciel qu'un système patrimonial peut posséder. Nous terminons le chapitre avec l'analyse de l'évolution d'un système patrimonial dans son cycle de vie. L'acquisition de l'ensemble de ces connaissances favorise la compréhension de nos discussions dans la suite de ce document.

1.1 Définition

Les premiers systèmes d'information commencèrent à se développer et à s'intégrer aux processus d'affaires des gouvernements, des grandes compagnies d'assurances et des institutions financières, il y a au moins 40 ans. Au fil des années, ces systèmes se sont régulièrement raffinés et adaptés aux changements stratégiques des organismes qui en sont propriétaires. Aujourd'hui, de tels systèmes, qui disposent de millions lignes de code et qui sont devenus les pivots de leur organisme, représentent les seules sources permettant d'identifier les processus d'affaires pour lesquels il y a peu (s'il y en a) de documentation.

Malgré l'importance des investissements antérieurs, la majorité des vieux systèmes d'information se voit confrontée aux problèmes suivants [6] :

- ces systèmes s'exécutent généralement sur des machines obsolètes dont l'entretien est très coûteux et n'est souvent plus offert par le fournisseur du matériel [7] [1] ;

- la maintenance du code source de ces systèmes représente une tâche chronophage (de l'anglais *time-consuming*) et risquée. D'une part, le personnel qui entretient le code n'a généralement pas assisté à la réalisation du système et ne le connait pas assez. D'autre part, l'impact d'un petit changement sur l'ensemble du système est difficile à anticiper ;

- l'ajout de nouvelles fonctionnalités à un tel système est un exercice difficile, si ce n'est impossible ;

- le manque de standard commun à l'époque du développement des vieux systèmes d'information fait en sorte qu'aujourd'hui, leurs interfaces systèmes (voir la page 17) ne sont pas conformes aux standards du marché. C'est la raison pour laquelle l'intégration de systèmes patrimoniaux - tels qu'ils sont - à n'importe quel autre système échoue fréquemment ;

- les IUs textuelles de ces systèmes ne sont pas conviviales pour les utilisateurs d'aujourd'hui qui, ont l'habitude de travailler avec des interfaces graphiques et d'utiliser la souris et d'autres périphériques pour pouvoir bénéficier d'une meilleure expérience utilisateur.

1. La série 360 (S/360) et la série 370 (S/370) d'IBM figurent parmi les premiers ordinateurs centraux - fabriqués et mis sur le marché par la compagnie IBM - dont l'entretien matériel n'est plus offert par le fabriquant.

Dans [10], un *système d'information patrimonial* (de l'anglais *legacy information system*) [2] est défini comme étant un système d'information qui résiste de façon significative aux moindres modifications imposées par les nouveaux besoins et exigences de son domaine d'affaires. De nos jours, le besoin de mises à jour régulières du logiciel et du matériel des systèmes d'information - rendu nécessaire à cause de l'avancement rapide de la technologie - vient s'ajouter aux exigences du domaine d'affaires auxquelles il faut adapter un tel système.

Brodie et Stonebraker [10] stipulent que tous les systèmes d'information patrimoniaux partagent les caractéristiques suivantes :

- ils sont âgés de plus de 30 ans ;
- ils contiennent des millions de lignes de code ;
- ils sont développés en utilisant des langages de programmation de troisième génération (de l'anglais *3GL* ou *third-generation programming language*) comme Cobol, Rpg, Fortran 77, PL/I, Basic, Pascal, etc. ;
- ils utilisent soit de simples modèles de données, comme les systèmes de fichiers, soit des bases de données patrimoniales comme ISAM (pour *Indexed Sequential Access Method*), VSAM (pour *Virtual Storage Access Method*), Codasyle [39], etc. ;
- ils sont généralement autonomes, c'est-à-dire qu'ils possèdent peu d'interfaces avec d'autres systèmes.

Dit de façon simplifiée, un système patrimonial est un grand système obsolète auquel on ne se sait pas comment faire face, mais qui est vital à un organisme [5].

Selon certains auteurs, l'environnement immédiat d'un système d'information - des développeurs ainsi que des procédures administratives - est inclus dans la définition d'un système patrimonial. Avec le temps, modifier un système patrimonial devient de plus en plus difficile. Il en est de même pour changer la façon de penser des gens qui ont longtemps travaillé avec un tel système.

Les programmes de commande et les programmes en-ligne sont deux différents types de programmes que l'on retrouve dans un système d'information patrimonial. Dans la suite de cette section, nous présentons ces deux types de programmes plus en détail.

2. Les termes LIS pour «*Legacy Information System*» et EIS pour «*Enterprise Information System*» sont fréquemment utilisés dans la littérature, afin de désigner les systèmes d'information patrimoniaux.

1.1.1 Programme de commande

Les premiers systèmes d'information se composaient de programmes de commande (de l'anglais *batch program*), développés pour être exécutés sur des machines gigantesques, appelées *ordinateurs centraux* (de l'anglais *mainframes*) ou *hôtes* (de l'anglais *hosts*). Les ordinateurs centraux sont réputés pour leur grande capacité d'effectuer des traitements par lots (de l'anglais *batch processing*). Le *traitement par lots* est un type de traitement de données dans lequel on traite des entrées fournies de façon séquentielle, et ce, avec un minimum d'intervention humaine.

Un *programme de commande* est conçu pour réaliser des traitements par lots. Les entrées d'un programme de commande proviennent de fichiers en lecture (de l'anglais *input files*) - soit des fichiers de mouvements (de l'anglais *transaction files*), soit des fichiers de paramètres. À chaque programme de commande, un ou plusieurs fichiers en lecture sont associés. Le programme fait donc appel à ces fichiers pour se procurer les données d'entrée. Ces entrées sont séquentiellement interprétées par le programme et servent à paramétrer la lecture des données de système à partir de bases de données ou de fichiers indexés. Les résultats de ces traitements de données sont ensuite envoyés aux imprimantes, sous forme de rapports, ou aux fichiers de sortie (de l'anglais *output files*) pour y être enregistrés [67].

Les programmes qui permettent aux banques et aux autres institutions financières d'effectuer des traitements de fin de trimestre et de produire les rapports nécessaires, pour leurs clients - exemple : les états de compte trimestriels ou les relevés de pension - ou le gouvernement - exemple : résultats financiers -, sont des exemples de programmes de commande effectuant des traitements par lots.

Pendant l'exécution, les traitements par lots n'imposent aucune interaction avec les utilisateurs. C'est la raison pour laquelle ces types de traitements sont aussi désignés par *traitements hors-ligne.*

Contrairement à un programme de commande qui ne nécessite aucune interface en vue d'interagir avec des utilisateurs, le fonctionnement d'un programme en-ligne est basé sur des interactions directes avec des utilisateurs. La suite de cette section est consacrée à la présentation des programmes en-ligne et des technologies qui peuvent être employées pour faciliter leur développement.

1.1.2 Programme en-ligne

L'arrivée des terminaux au début des années 70, donna naissance aux traitements transactionnels en-ligne[3]. À l'opposé des traitements hors-ligne, dans les *traitement en-ligne*, l'interaction avec utilisateur est instantanée, c'est-à-dire que le temps écoulé entre la saisie des données d'entrée et leur traitement se limite à quelques instants.

Comme son nom indique, un *programme en-ligne* est un programme qui permet de réaliser des traitements en-ligne. Un tel programme est aussi appelé *interactif*. Contrairement à un programme de commande, il est possible d'accorder plus d'une tâche à un programme en-ligne ou d'arrêter ses traitements pendant l'exécution [67].

La figure 1.1 illustre le schéma d'un système transactionnel. L'idée est de servir plus d'un utilisateur à la fois, et ce, en connectant plusieurs terminaux à un ordinateur de grande puissance de traitement : l'ordinateur central. Dans ce contexte, les terminaux n'ont aucune capacité de traitements logiques. Ils assument uniquement la saisie des informations de la part des utilisateurs, leur envoi à l'ordinateur central et l'affichage des résultats provenant de ce dernier. Leur nomination en anglais, « dump terminal » fait référence à cette caractéristique. Dans la littérature, on voit aussi la présence des *terminaux commutés* qui utilisaient une ligne commutée (de l'anglais *dial-up line*), afin de se connecter à l'ordinateur central[4].

Comme l'illustre la figure 1.1, c'est par l'entremise des terminaux que les données d'entrée sont saisies et envoyées aux programmes hébergés sur l'ordinateur central. Cette communication de données se fait soit en mode blocs, soit en mode caractères. L'architecture matérielle d'un terminal détermine la façon dont les données circulent entre celui-ci et des programmes en-ligne. Entre un *terminal orienté bloc* et l'ordinateur central, les données circulent en forme de blocs - souvent des blocs de textes. Les terminaux IBM 3270 sont probablement les plus répandus possédant une architecture orientée bloc. En revanche, un *terminal orienté caractère* communique les données un caractère à la fois. Dans ce contexte, un *protocole de communication* met en place une spécification de plusieurs règles, selon le mode d'envoi et de réception de données.

Afin de coordonner la gestion de la communication de données et de faciliter le développement des programmes en-ligne - dans un système transactionnel - on a généralement

3. Les premiers terminaux sont conçus et mis sur le marché par de grands fournisseurs de matériel tels qu'IBM.

4. Le lecteur est invité à se référer à l'adresse `http://publib.boulder.ibm.com/infocenter/zos/basics/topic/com.ibm.zos.zmainframe/toc.htm` pour obtenir davantage d'information sur les concepts des ordinateurs centraux.

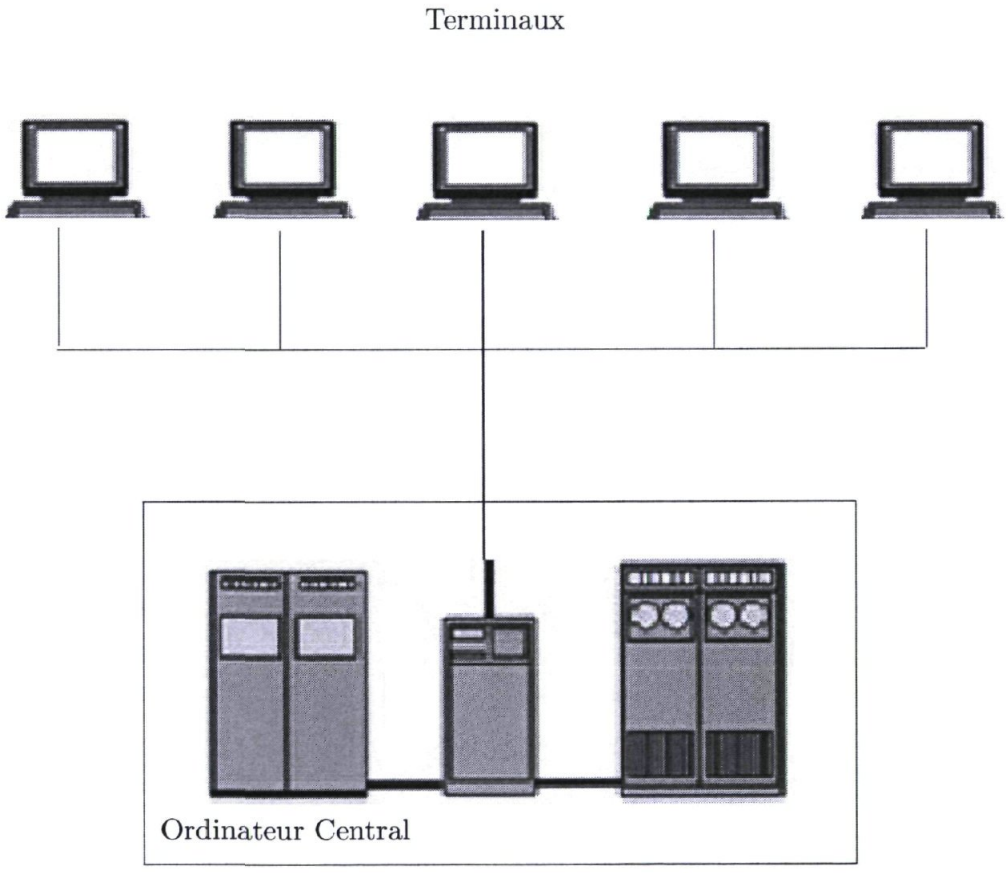

FIGURE 1.1 – Architecture d'un système transactionnel.

recours aux services d'un moniteur de télétraitement.

Dans la migration des IUs textuelles d'un système patrimonial, il est souvent indispensable de connaître le moniteur de télétraitement utilisé par le système pour comprendre ou simuler la communication entre des programmes en-ligne et des terminaux. La définition d'un moniteur de télétraitement, ses tâches dans un système transactionnel et deux exemples de tels moniteurs sont exposés dans la section suivante.

1.1.2.1 Moniteur de télétraitement

À l'instar d'un système d'exploitation, un *moniteur de télétraitement* est un logiciel complexe faisant partie de la plate-forme logicielle d'un ordinateur central. Les tâches suivantes sont souvent confiées à un moniteur de télétraitement [33] :

- gestion des accès, lorsque plusieurs terminaux se connectent au même hôte ;
- gestion de l'attribution des ressources de l'ordinateur central - mémoire, temps d'exécution, etc. - aux programmes exécutés simultanément ;
- libération des ressources, lorsqu'elles ne sont plus utilisées ;
- gestion de la réception et de l'envoi des données provenant des terminaux en utilisant un protocole de communication ;
- garantie de la sécurité et de l'intégrité du système.

Dans un système d'information transactionnel, un moniteur de télétraitement intervient à plusieurs niveaux en tant que coordinateur. Un des rôles d'un moniteur de télétraitement se situe au niveau de la communication entre l'ordinateur central et les terminaux qui lui sont associés. La gestion de la communication entre des programmes en-ligne - installés sur l'ordinateur central - et des terminaux implique la gestion de protocoles de communication, de mémoires tampons de programmes (de l'anglais *input/output buffer*) ainsi que de traitements assez complexes de chaînes de caractères et de données. Dans un système transactionnel, la *mémoire tampon d'entrée* d'un programme sert à emmagasiner les messages reçus des terminaux. Quant à la *mémoire tampon de sortie*, elle emmagasine les messages destinés à des terminaux. Dans les deux cas, le traitement des messages doit se faire par ordre chronologique d'entrée (de l'anglais *First In, First Out* ou *FIFO*).

L'utilisation de moniteur de télétraitement - comme médiateur [5] - entre des terminaux et des programmes, pour gérer la communication entre eux et pour organiser et sérialiser

5. Traduction du terme *middleware* selon l'Office québécois de la langue française (OQLF).

les données de communication, simplifie grandement le développement des programmes en-ligne d'un système transactionnel. En d'autres termes, un programme en-ligne n'a pas à se soucier du formatage des données qu'il produit, ni de la gestion de protocole de communication pour envoyer ses données aux terminaux. C'est le moniteur de télétraitement qui est en charge du formatage ainsi que de l'envoi et de la réception de données entre les programmes et les terminaux, et ce, via des interfaces préalablement programmées, dites *descriptions d'écran* (de l'anglais *screen masks*).

Tel qu'illustré à la figure 1.2, un programme en-ligne - lorsqu'il est appelé - s'exécute et produit ses données de sortie (de l'anglais *output data*). Il les communique ensuite au moniteur de télétraitement en mentionnant le nom d'une description d'écran. Trouver la description d'écran - demandée par le programme - dans un référentiel de descriptions d'écran et y mapper les données de sortie devient du ressort du moniteur de télétraitement.

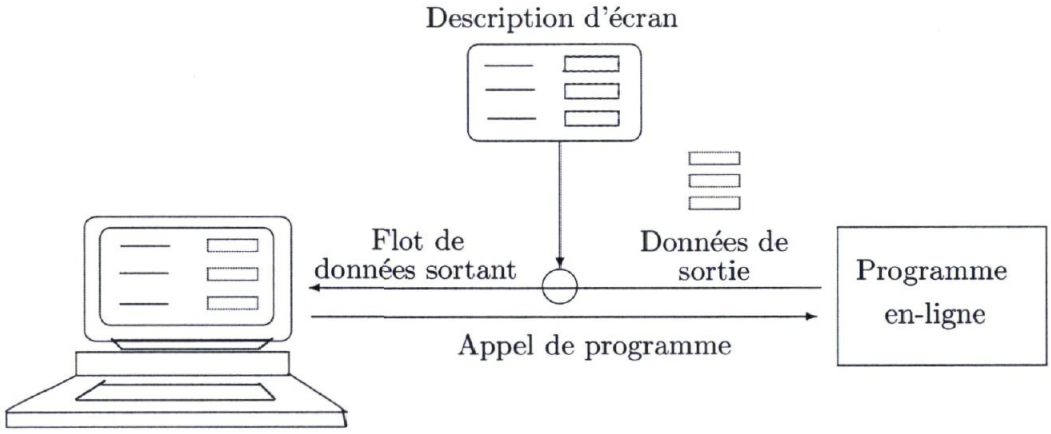

FIGURE 1.2 – Utilisation des descriptions d'écran.

Tel qu'expliqué, une description d'écran est un ensemble d'éléments qui sert à formater les données de sortie d'un programme en-ligne, avant de les envoyer au terminal. Notons que chaque moniteur de télétraitement est muni d'un langage de programmation propriétaire pour développer des descriptions d'écran.

La figure 1.3 est un exemple de définition d'un élément ainsi que ses propriétés dans une description d'écran. Comme nous pouvons le constater, la couleur de fonte, la position, la longueur, la hauteur et plusieurs autres informations sur l'apparence de l'élément sur le terminal sont définies dans cette présentation. Tel qu'illustré, les différentes catégories regroupent les propriétés d'un élément. Les lignes qui n'ont pas été indentées représentent ces catégories. À titre d'exemple, FOREGROUND_ATTRIBUTES com-

```
FIELD &
   NAME = "TRANS-LINE"
INPUT_ATTRIBUTES &
   REPLENISH_NOTIFICATION = NO &
   DATA_ENTRY = OPTIONAL &
   BLANK_INPUT = BLANK &
   ALLOW_QUERY = NO &
   VALUE_CHECK = NONE
SECURITY &
   LEVEL = 1 &
   TYPE = SOFT &
   FAIL_ACTION = NORMAL
FOREGROUND_ATTRIBUTES &
   COLOR = WHITE &
   INTENSITY = NORMAL
BACKGROUND_ATTRIBUTES &
   COLOR = BLACK
DIMENSION &
   HEIGHT = 1 &
   WIDTH = 50 &
   VARIABLE_WIDTH = NO
POSITION_IMAGE &
   ROW = 1 &
   COLUMN = 1 &
   VARIABLE_COLUMN = YES &
   VARIABLE_ROW = YES
TEST &
   DEPENDENT = NO
OUTPUT_ATTRIBUTES &
   CONTENTS = STORAGE
STORAGE &
   DATA_TYPE = ALPHA_NUMERIC
GENERAL_EDIT &
   PROTECTED = NO &
   SET_MODIFIED = NO &
   JUSTIFICATION = NONE &
   CASE = EITHER &
   USAGE = INPUT_OUTPUT
HIGHLIGHT &
   TYPE = NONE
TAB &
   TYPE = NONE
EMPHASIS &
   TYPE = NONE
CHARACTER_ATTRIBUTES &
   FONT_INDEX = 0
END_FIELD
```

FIGURE 1.3 – Définition d'un champ de saisie dans une description d'écran.

prend la couleur : COLOR et l'intensité : INTENSITY d'avant-plan de l'élément [6] ou encore la catégorie DIMENSION contient la hauteur : HEIGHT et la largeur : WIDTH de l'élément sur l'écran. La valeur YES pour la propriété VARIABLE_WIDTH, contenu dans la catégorie DIMENSION, signifie que la largeur de l'élément peut être modifiée par le programme qui est associé à cette description d'écran, alors que la valeur NO ne donne pas cette possibilité au programme. Les explications complètes sur les différentes catégories et propriétés sont disponibles dans [19].

Le *mappage* - mise en correspondance - entre les données de sortie d'un programme et les éléments de la description d'écran qui correspond à ces données, se fait en associant à chaque donnée, les propriétés qui lui sont accordées par la définition d'écran. Par exemple, dans la figure 1.3, la valeur de la propriété NAME détermine le nom de la donnée à laquelle ces propriétés seront appliquées. L'ensemble des données de sortie d'un programme et sa description d'écran forme un *flot de données sortant* (de l'anglais *outbound data stream*) qui est envoyé au terminal déclencheur de la communication.

En recevant un tel flot de données sortant, le terminal interprète les propriétés de chaque élément et affiche le résultat. Ce dernier est connu en tant qu'*interface utilisateur textuelle* (de l'anglais *Character-based User Interface* ou *CUI*). Les interfaces utilisateurs textuelles des systèmes patrimoniaux se composent généralement de champs de saisie (de l'anglais *input fields*) et de champs de sortie protégés en écriture (de l'anglais *output fields*). Les deux groupes de champs ont les mêmes caractéristiques. Ils se différencient uniquement par le fait que les champs de saisie sont modifiables (de l'anglais *editable*) par les utilisateurs du système, alors que les champs de sortie ne le sont pas. Un exemple d'une IU textuelle est présenté à la figure 1.4.

Il est intéressant de noter que la notion de mappage, tel que présentée précédemment, correspond assez fidèlement à la notion plus récente de pages Web dynamiques générées par les serveurs Web. En effet, de telles pages correspondent à l'union de gabarits - masks - de pages Web et de données provenant de sources de données, telles que les bases de données, hébergées généralement sur des serveurs.

Notons que la plate-forme logicielle de certains ordinateurs centraux n'est pas équipée de moniteur de télétraitement. Dans ce contexte, il est du ressort de chaque programme en-ligne du système transactionnel de gérer ses propres envois et réceptions de données.

6. Selon l'Office québécois de la langue française, la couleur d'avant-plan : se dit de la couleur attribuée aux objets ou aux caractères affichés à l'écran choisie à partir d'une palette de couleurs disponible.

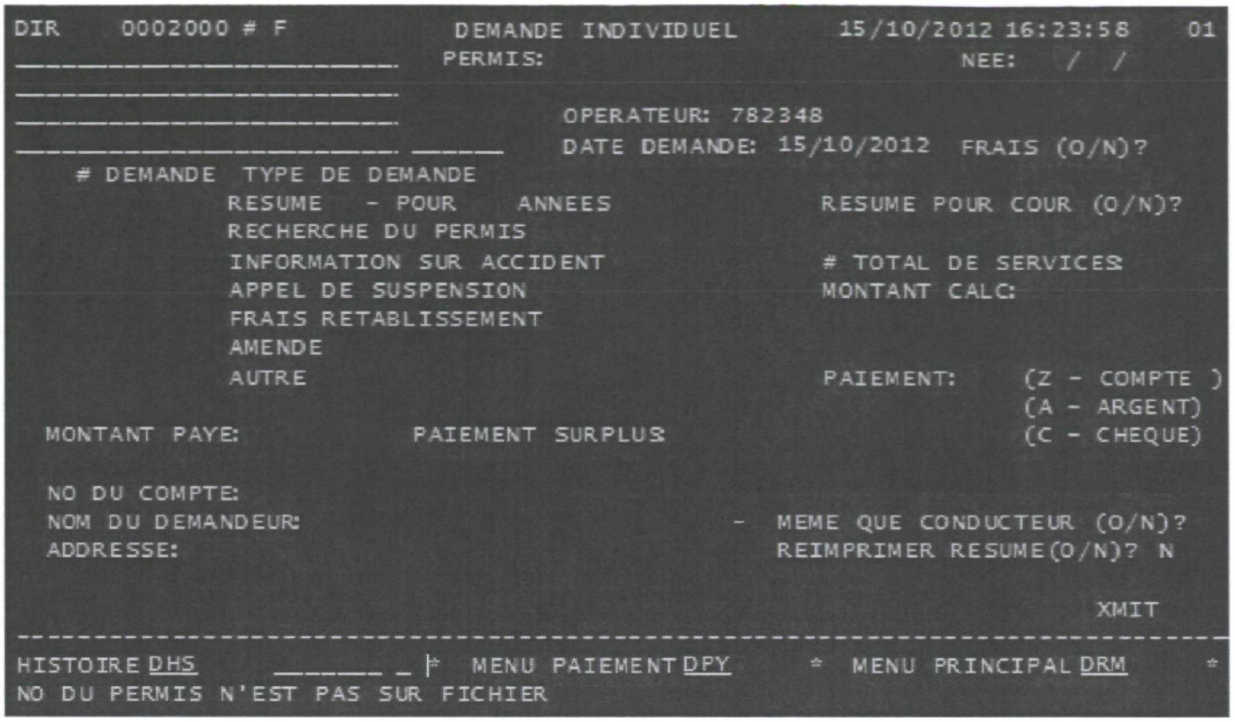

FIGURE 1.4 – Exemple d'une interface utilisateur textuelle.

CICS (pour *Customer Information Control System* ou Système de Contrôle des Informations Client, en français) [15], ainsi qu'IMS/DC (pour *Information Management System/Data Communications*) [32] sont deux moniteurs de télétraitement développés par la compagnie IBM à la fin des années 60 [57] [7]. CICS utilise le langage de description d'écran BMS (pour *Basic Mapping Support*) et IMS/DC propose MFS (pour *Map Format Service*) comme langage de description d'écran.

Côté utilisation, un moniteur de télétraitement est une bibliothèque de classes (de l'anglais *class library*). Un programme en-ligne fait donc usage des services d'un moniteur de télétraitement par le biais de macros, d'interfaces d'appel spéciales ou encore d'instructions qui y sont disponibles. Ces instructions sont dispersées dans le code du programme et se répètent à chaque fois qu'un appel aux services du moniteur de télétraitement est nécessaire. Les instructions utilisées pour appeler des fonctionnalités d'un moniteur de télétraitement changent selon le langage de programmation utilisé pour le développement des programmes en-ligne.

7. Puisque tous les logiciels et matériels des ordinateurs centraux sont de nature propriétaire, chaque fournisseur de matériel a nécessairement sa propre solution en terme de moniteur de télétraitement. On entend par *système propriétaire* un système, logiciel et matériel conjointement, qui appartiennent à un fournisseur unique et qui nécessitent des développements particuliers.

La figure 1.5 illustre un exemple d'un appel au moniteur de télétraitement CICS dans un code Cobol. Dans cette figure, la commande *EXEC CICS SEND* est utilisée pour l'envoi de données au terminal. L'option *FROM* spécifie le nom de la variable qui contient les données. L'option *LENGTH* correspond à la longueur totale des données, et finalement l'option *ERASE* demande au terminal d'effacer l'écran, avant d'afficher de nouvelles données.

```
EXEC CICS SEND
        FROM(WS_OUTPUT)
        LENGTH(WS-MSG-LENGTH)
        ERASE
END-EXEC.
```

FIGURE 1.5 – Exemple d'appel au moniteur de télétraitement CICS dans un code Cobol.

Nous verrons à la section 4.1 que les résultats de l'ingénierie-inverse des descriptions d'écran d'un système transactionnel fournissent les informations sur la structure visuelle des IUs textuelles. Dans la refonte des IUs textuelles vers le Web, ces informations constituent les spécifications statiques des IUs qui sont récupérées afin d'établir la mise en page (de l'anglais *layout*) des nouvelles IUs (voir la page 68).

Dans le chapitre 3, nous montrerons que toute migration d'IUs vers le Web nécessite l'hébergement de nouvelles IUs, et d'autres composants du système patrimonial faisant l'objet de migration, dans une architecture Web. L'architecture conceptuelle du système à l'étude joue un rôle clé dans le choix de l'architecture Web de migration. La section suivante établit la notion de l'architecture conceptuelle d'un système patrimonial et présente trois types d'architectures sur lesquels repose le développement d'un tel système.

1.2 Architecture

Chaque système patrimonial a une fonction principale. Celle-ci se divise en quelques sous-fonctions individuelles ; chacune de ces fonctions se divise, à son tour, en quelques sous-fonctions et ainsi de suite. Les fonctions d'un système, tel qu'exprimé, forment une hiérarchie de fonctions, que l'on nomme la *décomposition fonctionnelle* du système. Ainsi, l'architecture conceptuelle d'un système patrimonial est constituée de modules et de composants, chacun associé à un type de fonctions. Le regroupement des composants d'un système, selon les types de fonctions auxquels ces composants sont associés, donne trois différents groupes de composants :

- composants d'interface (interfaces utilisateurs et interfaces systèmes) ;
- composants de logique d'affaires (ou logique d'application) ;
- composants de données.

Les *composants d'IU* ont comme fonction de faire communiquer le système et ses utilisateurs. Quant aux *interfaces systèmes*, elles font communiquer les systèmes patrimoniaux entre eux. Les *composants de logique d'affaires* implémentent les règles d'affaires, et les composants de données, quant à eux, comprennent à la fois les *données de système* et les composants qui sont responsables de la communication entre la logique d'affaires et les données de système. On les nomme *composants de services de bases de données*. La façon dont l'implémentation de chaque groupe de composants est répartie dans le code source, détermine l'architecture conceptuelle du système. La suite de cette section expose trois types d'architecture conceptuelle de systèmes patrimoniaux présentés par Brodie et Stonebraker [10] :

- architecture décomposable ;
- architecture semi-décomposable ;
- architecture non décomposable.

1.2.1 Architecture décomposable

La figure 1.6 illustre l'architecture d'un *système décomposable* dont les composants sont distincts et facilement identifiables. Il s'agit donc d'un système modulaire. Tel qu'illustré, le système se compose d'une collection de modules d'application. Chaque module interagit indépendamment avec les services de bases de données, les interfaces utilisateurs ainsi que les interfaces systèmes. Pour que l'architecture d'un système patrimonial soit considérée comme étant décomposable, il est nécessaire que les modules d'application soient indépendants les uns des autres et qu'il n'y ait aucune hiérarchie entre eux.

1.2.2 Architecture semi-décomposable

La figure 1.7 illustre l'architecture d'un système *semi-décomposable*. Contrairement à une architecture décomposable, seules les interfaces systèmes et les IUs sont identifiables et séparables. Le reste du système, les composants de logique d'affaires et de services de bases de données, sont considérés comme formant une seule unité non décomposable. Comme nous l'avons mentionné, les systèmes d'information patrimoniaux

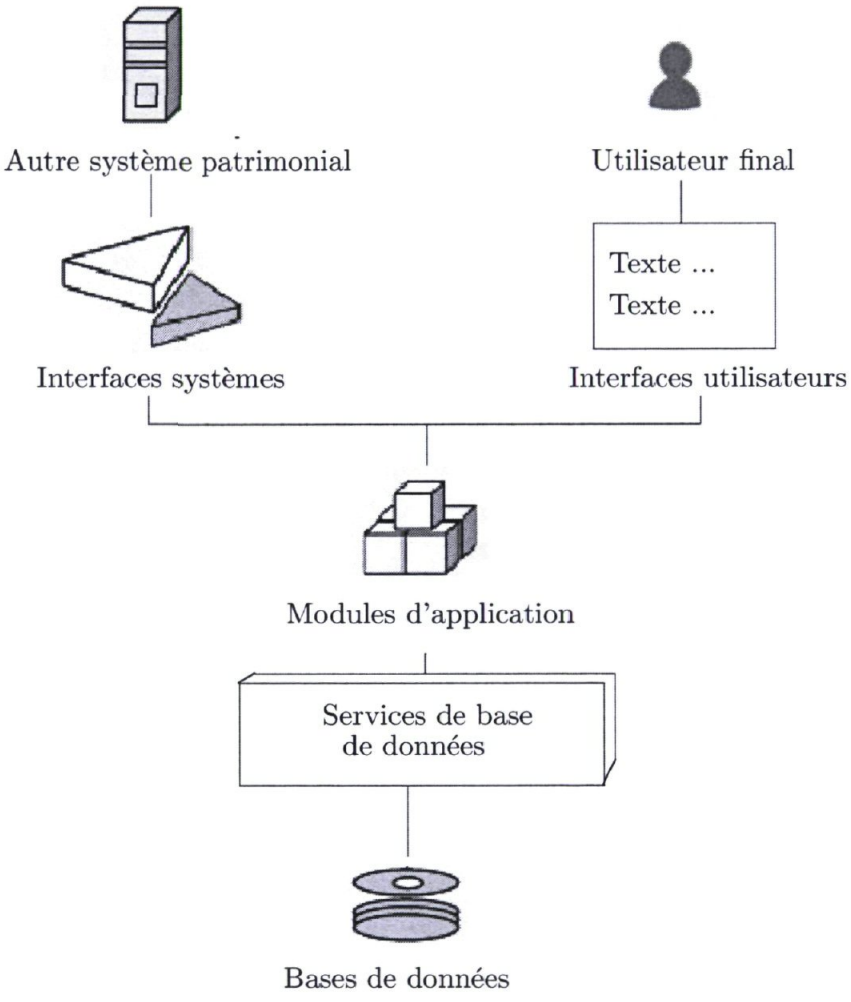

FIGURE 1.6 – Architecture d'un système patrimonial décomposable.

qui utilisent les services d'un moniteur de télétraitement possèdent une architecture semi-décomposable dans laquelle une grande partie des traitements qui concernent les IUs - la mise en page des IUs, par exemple - est séparée et se réalise par le moniteur de télétraitement.

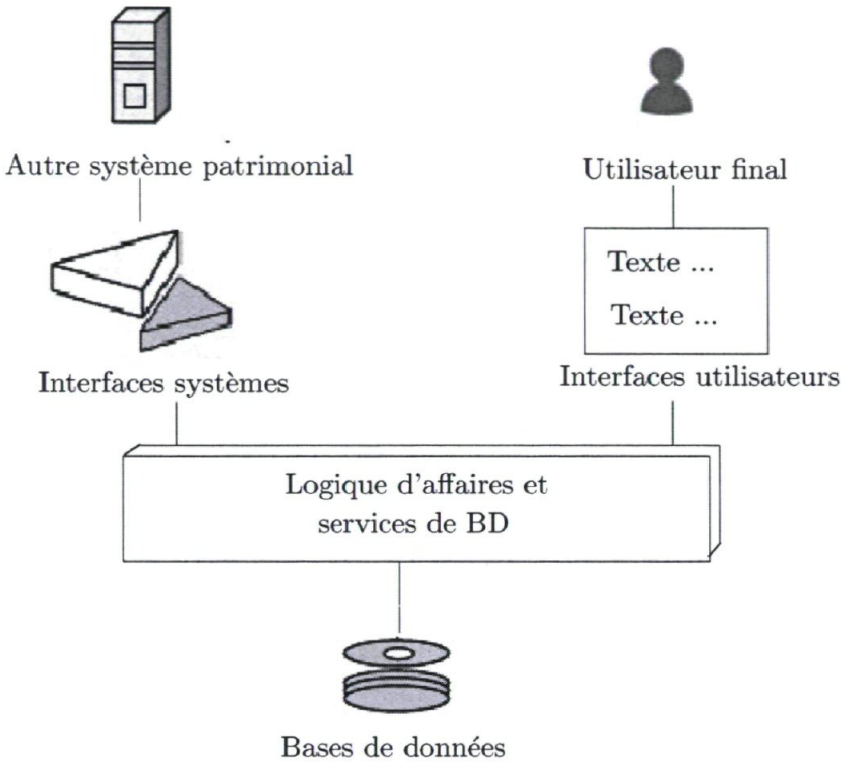

FIGURE 1.7 – Architecture d'un système patrimonial semi-décomposable.

1.2.3 Architecture non décomposable

La plupart des systèmes d'information patrimoniaux qui ont été développés entre les années 50 et 70 possèdent une architecture *non décomposable* (tel qu'illustré à la figure 1.8) dans laquelle les IUs sont généralement de type question-réponse (de l'anglais *prompt and response*). Dans un programme de type question-réponse, les entrées du programme sont demandées à l'utilisateur au fur et à mesure que le programme en a besoin. Ainsi, l'exécution du programme est arrêtée à chaque fois qu'une saisie de données devient nécessaire. Une fois que l'entrée est saisie, le programme continue son exécution jusqu'à la prochaine saisie et ainsi de suite. Dans ce contexte, le code source gérant les interactions avec l'utilisateur se trouve noyé dans le code monolithique du système. Le manque de frontière claire entre le code des différents composants fait en

sorte que dans une architecture non décomposable, aucun module ou composant ne peut être identifié ou extrait facilement.

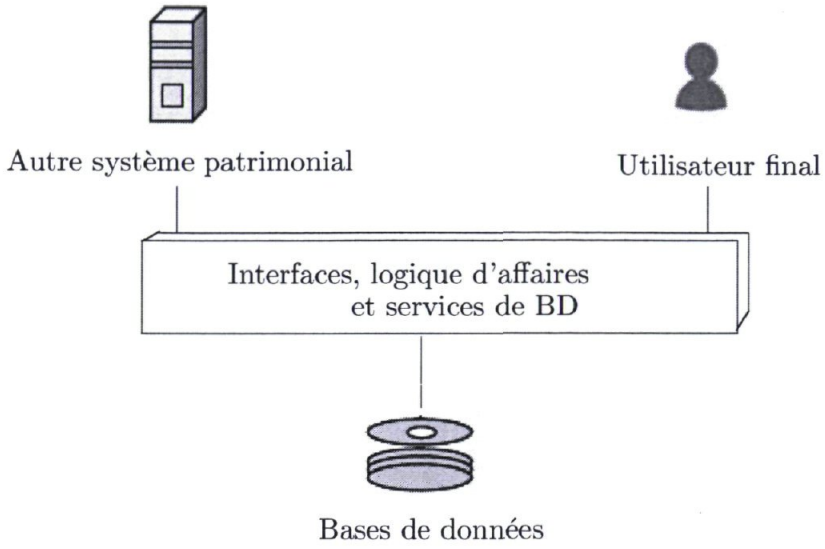

FIGURE 1.8 – Architecture d'un système patrimonial non décomposable.

Comme nous l'avons précédemment mentionné, nos discussions dans ce mémoire concernent essentiellement les activités de migration des IUs textuelles de systèmes patrimoniaux. Puisque ces activités s'inscrivent dans le cadre des activités de l'évolution d'un système, nous profitons du contexte de système patrimonial pour établir la notion d'évolution d'un tel système. Nous décrivons également le rôle et déterminons les objectifs des activités de migration d'IUs dans l'évolution de systèmes.

1.3 Évolution d'un système d'information

Comme il a été vu à la section précédente, un système patrimonial fait l'objet de plusieurs modifications durant son cycle de vie. Ces modifications que l'on nomme les *activités d'évolution de système*, couvrent un continuum allant de l'ajout d'un champ aux bases de données, au remplacement complet du système. Les activités d'évolution d'un système se déclinent en trois catégories : la maintenance, la migration et le remplacement [16].

La figure 1.9 donne un aperçu du cycle de vie d'un système d'information et les activités d'évolution qu'il subit pendant son existence. La ligne en pointillé représente les besoins croissants en affaires, alors que la ligne en continu désigne les fonctionnali-

tés offertes par le système. Comme nous pouvons le constater, grâce aux activités de maintenance, le système est en mesure de suivre l'évolution du domaine d'affaires sur une période limitée de temps. Cependant, avec le temps et l'avancement de la technologie, il devient de plus en plus difficile d'adapter le système aux nouveaux besoins d'affaires. À ce stade, la migration du système est indispensable. La migration représente des activités d'évolution plus importantes ; autant au niveau du code touché par des changements qu'au niveau du temps nécessaire pour la réalisation de ces changements [8]. Finalement, lorsque le vieux système ne peut aucunement évoluer, il est temps de le remplacer complètement par un nouveau système.

Le projet Renaissance est un grand projet de recherche financé par ESPRIT (pour *European Union's information technologies RTD program*), dans la ré-ingénierie et l'évolution des logiciels [54]. Renaissance favorise la transformation d'un système patrimonial en un système évolutif. Un *système évolutif* est un système flexible aux changements, capable de supporter les activités d'évolution sur une période plus longue que la normale.

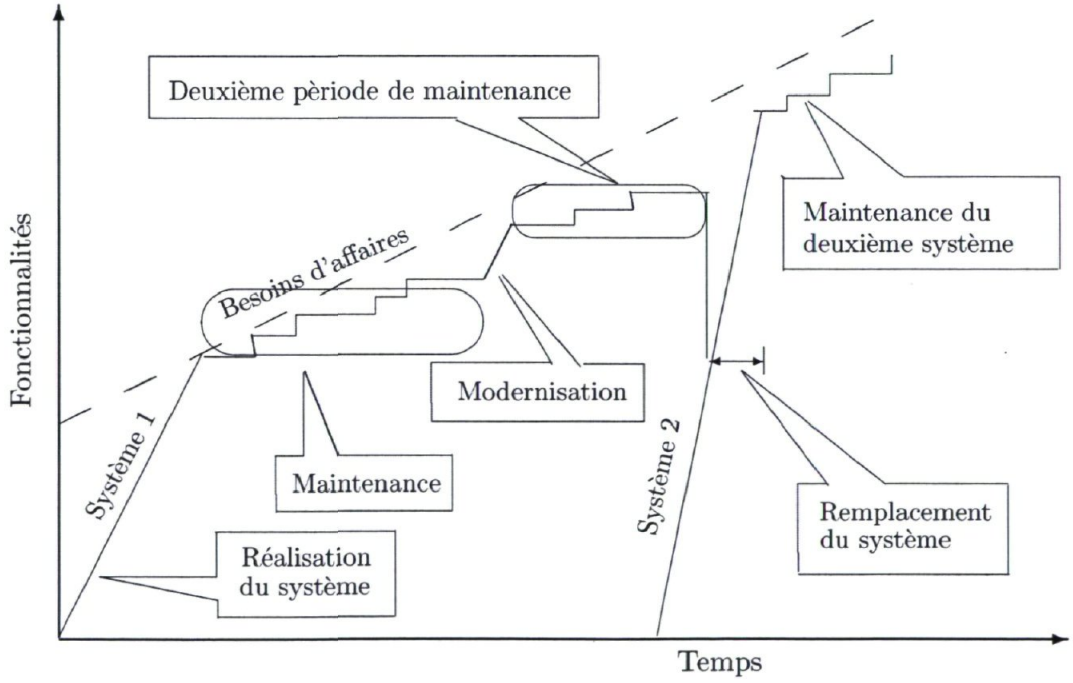

FIGURE 1.9 – Cycle de vie d'un système d'information [16].

Dans la suite de cette section, nous définissons les notions de maintenance, de migration et de remplacement [9].

8. Bien que certains chercheurs font référence au terme modernisation [16] pour désigner ces activités, dans ce document nous favorisons l'usage du terme migration. Dans la littérature, les deux termes sont utilisés de façon interchangeable.

9. Dans l'état de l'art, on constate que les définitions données aux termes migration et ré-ingénierie

1.3.1 Maintenance

L'institut national des normes et de la technologie (de l'anglais *The National Institute of Standards and Technology (NIST)* ou *National Bureau of Standards (NBS)* entre les années 1901 et 1988) définit la *maintenance* comme étant tout travail effectué sur un système après sa mise en production [40]. Lientz et Swanson utilisent cette définition pour décliner les activités de maintenance en quatre catégories [38] :

- activités de maintenance correctives ;
- activités de maintenance adaptives ;
- activités de maintenance d'amélioration (de l'anglais *perfective maintenance*) ;
- activités de maintenance valorisantes (de l'anglais *enhancive maintenance*).

Toutes les activités de maintenance impliquent des changements de code, de test et probablement de design. Cependant, l'objectif des changements diffère d'une activité à l'autre. Notons que toutes ces activités se font sur le système, pendant qu'il est en marche et qu'il ne résiste pas encore à des modifications.

Dans la littérature, le terme « ré-ingénierie » est utilisé de façon interchangeable avec les activités de migration d'un système patrimonial [63]. Dans le présent document, nous réservons le terme « ré-ingénierie » aux activités envisageant uniquement la restructuration de code. Cela nous permet de classer la ré-ingénierie dans les activités de maintenance de système patrimonial.

1.3.1.1 Ré-ingénierie

La *ré-ingénierie* concerne toute restructuration du code d'un système patrimonial ayant comme principal objectif d'augmenter sa qualité. Un exemple des activités de ré-ingénierie est la restructuration des programmes procéduraux - les programmes écrits en Cobol, PL/I ou Fortran - en y supprimant l'instruction GOTO. Cela augmente la lisibilité du code et facilite potentiellement sa maintenance. Dans ce contexte, on s'attend à ce que la restructuration ne perturbe aucunement le fonctionnement du code touché. Toutefois, la ré-ingénierie de code entraîne des risques. Harry Sneed fait une synthèse de 13 projets de ré-ingénierie afin de mettre en avant des risques qu'un projet de ré-ingénierie peut engendrer [65].

varient selon les chercheurs [6, 13, 16, 69]. Nos définitions ici reposent sur des travaux de Harry Sneed [69].

1.3.2 Migration

Devant les restrictions imposées par un système patrimonial, à savoir les IUs textuelles, l'accès local en raison de la nature propriétaire du système, le manque d'expertise pour maintenir le système, etc., un changement technologique est parfois le seul recours afin de pouvoir suivre l'évolution constante de la technologie et celle du domaine d'affaires. On nomme *migration* les activités d'évolution imposant un changement de technologie à un système patrimonial. Lors de la migration d'un système patrimonial, malgré les changements technologiques, on désire souvent pouvoir réutiliser des composants du système et reproduire les anciennes fonctionnalités autant que possible.

La migration d'un système patrimonial peut être faite à tous les niveaux, tant au niveau matériel qu'au niveau logiciel. Cependant, des changements matériels impliquent souvent des changements logiciels. Prenons l'exemple d'un système patrimonial s'exécutant sur un ordinateur central, la migration de la plate-forme matérielle de ce système requiert également des changements technologiques au niveau du logiciel - langage de programmation, technologie des bases de données, etc.

Les activités de migration déployées sur un système patrimonial transactionnel comprennent naturellement les efforts accomplis au niveau de ses IUs textuelles. Autrement dit, dans presque toute activité de migration sur un système transactionnel, les IUs sont touchées de près ou de loin. Ci-dessous, nous définissons la notion de migration d'IUs et nous énumérons les conditions sous lesquelles les résultats d'une telle migration sont satisfaisants.

1.3.2.1 Migration d'IUs textuelles

On nomme le changement de la plate-forme d'exécution des IUs d'un système patrimonial *la migration d'IUs textuelles*. Autrement dit, dans une migration d'IUs la plate-forme d'exécution est transférée d'un terminal, lequel présente les sorties uniquement sous forme textuelle et dispose simplement d'un clavier pour les entrées, vers d'une plate-forme graphique. La migration d'IUs textuelles est fréquemment étudiée dans le cadre de la migration complète d'un système patrimonial. Cependant, l'importance des IUs dans un système, d'une part, et l'architecture semi-décomposable de la majorité des systèmes patrimoniaux, d'autre part, font en sorte qu'une telle migration peut constituer un objectif en soi.

La *migration d'IUs textuelles vers le Web* à laquelle nous faisons référence dans ce

document, est une sous-catégorie de la migration d'IUs textuelles en général. Elle a comme objectif de migrer la plate-forme d'exécution d'IUs vers le Web.

Dans la migration d'IUs textuelles vers le Web, le changement de technologie - objectif intime de toute migration - impose souvent la refonte complète des IUs textuelles dans une technologie de présentation Web. Néanmoins, il existe des solutions à court terme qui ne nécessitent pas une telle refonte ; par *refonte* d'IUs textuelles, nous entendons le processus de génération des IUs Web équivalentes des IUs textuelles qui font l'objet de migration. Le chapitre 3 est entièrement consacré à la présentation des différentes techniques de migration d'IUs textuelles vers le Web et le chapitre 4 expose les étapes et les techniques de refonte automatisée d'IUs.

En général, dans une migration d'IUs textuelles vers le Web, il convient de satisfaire les conditions suivantes :

- automatiser le processus de migration - dans le processus de migration d'IUs, l'automatisation réduit considérablement les coûts de l'opération. Il est donc souhaitable que les différentes étapes de migration soient automatisées ;

- conserver les anciennes fonctionnalités du système - il est possible que la migration des IUs d'un système change partiellement ou complètement l'aspect et la convivialité des IUs textuelles originelles. Cependant, après la migration, les IUs graphiques générées doivent continuer d'offrir les anciennes fonctionnalités ;

- générer des IUs multiplate-formes - dans une migration d'IUs vers le Web, les nouvelles IUs sont destinées à s'intégrer aux navigateurs Web. Cependant, l'organisme qui investit dans une telle migration peut éventuellement souhaiter offrir l'accès à ses services via des téléphones portables, des tablettes ou d'autres plate-formes. C'est la raison pour laquelle un tel organisme favorise la mise en place de solutions de migration étant en mesure de générer des IUs multiplate-formes ;

- générer du code maintenable - la migration des IUs d'un système patrimonial donne une nouvelle vie à ce dernier. Malgré que la durée de vie d'un système migré est normalement moins longue que celle du système originel, il demeure nécessaire de pouvoir maintenir le code et d'y apporter des modifications demandées.

1.3.3 Remplacement

Lorsque ni les activités de maintenance, ni celles de migration ne remédient au retard technologique d'un système patrimonial et ne l'aident à suivre les nouveaux besoins techniques et organisationnels, le *remplacement* ou la *ré-implémentation* (aussi désigné par *big bang* ou *cold turky* [10]) du système s'impose. Cela implique le développement du système patrimonial à partir de zéro, généralement, en utilisant les dernières technologies du marché. Le remplacement d'un système d'information patrimonial entraîne des problèmes et des risques insoutenables [65]. Parmi ces problèmes, nous mentionnons les suivants :

- la ré-implémentation d'un système patrimonial occasionne généralement des coûts et des risques importants :
  - des coûts importants reliés à la formation des utilisateurs afin qu'ils apprennent à utiliser le nouveau système [10] ;
  - de risques d'échec du projet en raison du manque de respect du budget et des délais.

- le nouveau système ne se prête généralement pas bien à la reproduction des anciennes fonctionnalités offertes par le système patrimonial. On fait souvent face à un manque de documentation pour le système patrimonial, et il est difficile d'identifier précisément les règles d'affaires dans le code du système [68] ;

- pour arriver au même niveau de robustesse et de fiabilité, il est crucial d'effectuer de nombreux tests sur le nouveau système ;

- compte tenu de l'ampleur de la plupart des systèmes d'information patrimoniaux, leur remplacement risque de s'étendre sur une longue période de temps. Considérant l'avancement rapide des technologies dans le domaine de l'information, il se peut qu'après la livraison, les technologies utilisées dans le développement du nouveau système soient obsolètes et dépassées et ne répondent plus aux besoins du domaine d'affaires [7].

Pour diminuer la probabilité d'échec d'un projet de remplacement d'un système patrimonial, les risques et les coûts reliés au remplacement doivent être soigneusement évalués à l'avance [7, 17]. Un processus complet de remplacement d'un système patrimonial est expliqué par Seng et Tsai, dans [58].

10. On constate fréquemment une forte résistance à l'apprentissage d'un nouveau système chez des utilisateurs qui ont longtemps travaillé avec un système existant.

Une solution alternative pour la ré-implémentation d'un système patrimonial est de le remplacer par un progiciel prêt à utiliser (de l'anglais *Commercial Off-The-Shelf system ou COTS system*). L'utilisation de composants prêts à utiliser réduit des coûts de conception et de maintenance du nouveau système. Cependant, il faut personnaliser ces composants aux propres besoins de l'organisme qui en fait usage. Cela implique des risques considérables, notamment lorsque le système patrimonial représente un système critique offrant des fonctionnalités très spécialisées.

1.4 Conclusion

Lors du présent chapitre, nous avons défini la notion de système d'information patrimonial. Nous avons vu qu'un tel système est constitué de deux types de programmes : programme de commande et programme en-ligne. Bien que nous ayons présenté le fonctionnement de chaque type de programme, notre étude était focalisée sur le fonctionnement des programmes en-ligne, lesquels servent des IUs textuelles afin de communiquer avec des utilisateurs. Nous avons également décrit trois architectures conceptuelles des systèmes patrimoniaux en soulignant le fait que la majorité de ces systèmes possède une architecture semi-décomposable ou non décomposable. Les activités d'évolution d'un système patrimonial exposées dans ce chapitre nous permettent de situer les activités de migration d'IUs que nous présenterons dans les prochains chapitres.

Le chapitre suivant expose la naissance et l'évolution du Web. Nous y présentons également quelques techniques de présentation Web. À l'opposé des notions exprimées au présent chapitre, le Web et les technologies employées dans le contexte du Web sont récents et l'information qui les concerne est donc plus accessible. C'est la raison pour laquelle, au lieu de faire une étude exhaustive de ces notions, nous limitons notre présentation aux sujets auxquels nous faisons référence dans ce document.

Chapitre 2

Web

Vague but exciting...

A commenté Mike Sendall, lors de la présentation de World Wide Web par Tim Berners-Lee.

À ses débuts, le Web a servi de moyen de partage de contenus statiques. Depuis, il a énormément évolué, de telle manière qu'il est devenu l'infrastructure déterminante pour le développement de systèmes distribués axés sur le service (de l'anglais *service-based*), dans lesquels le principal objectif consiste à favoriser la réutilisabilité et l'intégrabilité des composants. Au niveau conceptuel, ces systèmes sont développés à partir des standards ouverts définis en association avec de larges consortiums tels que le World Wide Web Consortium (W3C) [77] et l'organisation pour l'avancement des normes structurées de l'information (OASIS) [51]. L'utilisation des standards ouverts augmente la propriété d'interopérabilité et de réutilisabilité de tels systèmes, c'est-à-dire leur capacité de communiquer, de collaborer et de s'intégrer avec d'autres systèmes. Grâce aux standards ouverts, il est donc devenu possible de développer une application Web et de l'utiliser sur des environnements hétérogènes. Dans ce contexte, la migration de systèmes patrimoniaux vers le Web vise à intégrer de tels systèmes dans des architectures Web pour leur permettre, entre autres, de pallier les problèmes associés à l'accessibilité limitée.

Dans ce chapitre, nous définissons les notions du Web et des technologies de présentation Web. Dans le contexte de nos discussions sur la migration d'IUs, ces notions représentent l'environnement cible dans lequel des nouvelles IUs seront développées. Il nous apparait donc approprié de présenter les différents choix de technologies qu'il est possible d'utiliser dans le processus de migration vers le Web.

2.1 Évolution du Web

Comme nous l'avons vu à la section 1.2, l'architecture conceptuelle d'un système patrimonial peut être décomposable, semi-décomposable ou non décomposable. La migration des IUs textuelles d'un tel système vers le Web impose l'intégration du système dans une architecture Web. Comme nous verrons dans la suite de cette section, à la base, le Web n'a pas été conçu pour pouvoir héberger des applications complexes. Cependant, il a rapidement évolué et s'est transformé en une plate-forme de déploiement d'applications.

Dans cette section, nous tentons d'étudier brièvement cette évolution en mentionnant des différentes solutions architecturales qui ont été progressivement proposées pour améliorer cette plate-forme de déploiement. Bien qu'aujourd'hui, on se dirige de plus en plus vers la conception des applications par composants, certaines architectures comme l'architecture Web basée CGI ou encore les pages Web dynamiques s'harmonisent bien avec l'architecture conceptuelle de systèmes patrimoniaux. C'est la raison pour laquelle on a tendance à vouloir choisir ce type d'architecture pour la migration de systèmes patrimoniaux.

2.1.1 Naissance du Web

Dans les années 60, le département de la Défense des États-Unis (*United States Department of Defense*, abrégé par DoD ou par DOD) définit le projet de conception d'un grand réseau d'ordinateurs. L'idée fut de faciliter la communication entre les différents centres de recherche où les projets de la Défense se réalisaient. La robustesse constituait l'un des objectifs importants visés par ce projet. En d'autres termes, le réseau devait continuer à offrir ses services aux utilisateurs, même si certains noeuds du réseau cessaient à fonctionner.

L'ARPA (pour *Advanced Research Projects Agency* en anglais) finança la conception du premier réseau du même type connectant une douzaine de laboratoires de recherche de la Défense, en 1969. Malgré les premiers objectifs définis pour ce réseau, la communication textuelle par courriel fut le seul service offert par le réseau ARPAnet. Étant donné que l'utilisation d'ARPAnet se limita aux centres de recherche et des universités partenaires des projets de la Défense, certaines universités développèrent parallèlement leurs propres réseaux de communication. BITNET (pour *Because It's Time Network*) fut conçu par l'Université de New York (*City University of New York*) pour les com-

munications électroniques et le transfert de fichiers. CSNET (pour *Computer Science Network*) qui connecta l'Université de Delaware, l'Université de Purdue, l'Université de Wisconsin et quelques compagnies de recherche, fut également conçu pour effectuer des communications électroniques. Notons que BITNET et CSNET n'ont jamais été utilisés à grande échelle.

En 1986, un nouveau réseau national fut conçu aux États-Unis. Financé par la NSF (pour *National Science Foundation*), il connecta cinq centres de recherche universitaires. Le NSFnet se mondialisa peu de temps après son établissement. Avec 1 million noeuds, en 1992, le NSFnet forma le plus grand réseau à travers du monde. *Internet* est le nom que l'on donna au réseau NSFnet depuis 1995, lorsqu'il a été servi, pour la première fois, comme étant un réseau de recherche. Internet est un terme qui fait référence à la fois à l'infrastructure physique et l'infrastructure logique du réseau.

Chaque ordinateur membre du réseau, que l'on nomme l'*hôte*, possède une adresse réseau unique, appelée l'*adresse IP* (pour *Internet Protocol Address* ou *IP address*). Matériellement, les hôtes sont connectés les uns aux autres par des câbles réseaux. Le protocole TCP/IP (pour *Transmission Control Protocol/Internet Protocol*) est le moyen de communication logiciel entre les différents hôtes. Grâce à TCP/IP, Internet offre plusieurs services à ses utilisateurs. Ci-dessous, nous énumérons quelques-uns de ces services [1] :

- e-mail - SMTP ,POP3/IMAP, MIME ;
- système de noms de domaine (pour *Domain Name Service* ou DNS) - DNS est un système distribué de bases de données et de serveurs, qui assure la traduction des noms de domaine utilisés par les internautes en numéros Internet utilisables par les ordinateurs, ceci pour permettre la transmission des messages d'un site à l'autre du réseau. Le système de noms de domaine a été mis au point pour permettre aux internautes d'utiliser des noms dans la rédaction des adresses, ce qui est beaucoup plus facile à manipuler que des suites de chiffres ;
- protocole FTP (pour *File Transfer Protocol*) - FTP est un protocole de transfert de fichiers qui permet de télécharger une copie de données choisies par l'internaute, d'un ordinateur à un autre, selon le modèle client-serveur. Les données téléchargées peuvent prendre la forme de logiciels, de fichiers de toute nature - textes, graphiques, images, sons, etc. ;
- protocole Telnet - Telnet est un protocole d'émulation de terminal qui permet à l'internaute d'entrer en communication avec un hôte Internet à partir de son propre ordinateur pour utiliser des programmes ou consulter des données qui y sont stockées ;

1. Les définitions ci-dessous proviennent du site de Wikipédia.

– Web (pour *World Wide Web*).

Le Web est probablement le plus connu des services offerts par Internet.

En 1989, Tim Berners-Lee et son équipe au CERN (pour *Conseil Européen pour la Recherche Nucléaire*) proposèrent un nouveau protocole pour Internet - protocole HTTP - ainsi qu'un système d'accès aux documents utilisant ce protocole. L'objectif de ce nouveau système qu'ils appelèrent *World Wide Web* - désormais désigné par Web - fut de permettre aux scientifiques à travers le monde d'utiliser Internet afin de partager les documents décrivant leurs travaux. Ainsi, un scientifique, peu importe son emplacement sur Internet, fut en mesure de faire une recherche d'un document en spécifiant son URL (pour *Uniform Resource Locator*) et de le récupérer des bases de données se trouvant sur différents ordinateurs sur Internet - dans l'état de l'art, les appellations *page Web* et *ressource* ont également été données au terme «document» [56].

2.1.2 Pages Web

Du point de vue conceptuel, l'architecture de base du Web est composée d'un client et un serveur Web. Les deux parties se trouvent généralement sur deux ordinateurs distincts sur le Web et communiquent par l'entremise du protocole HTTP. Le *serveur Web* est un logiciel s'exécutant sur un hôte Internet. Tout serveur Web, lorsqu'il est exécuté, est prêt à recevoir des requêtes de la part des clients Web et de les traiter. Le *client Web* - un navigateur - est l'élément déclencheur de la communication entre les deux parties. La figure 2.1 illustre le cas le plus basique d'une communication client-serveur Web. Le serveur Web maintient un contenu statique référentiel (de l'anglais *static content repository*), tel qu'un système de fichiers ou des ressources d'information. Lors de la réception d'une demande (requête) de la part du client, le serveur Web récupère le document demandé, appelé *page Web*, du contenu référentiel et l'envoie au client [85].

Le contenu d'une page Web est en format *langage de balisage hypertexte* (de l'anglais *Hypertext Markup Language* ou HTML) ; nous présentons HTML, plus loin dans ce chapitre. Un document hypertexte peut contenir des liens vers d'autres documents pour permettre une navigation non séquentielle. Le protocole utilisé pour le transfert d'information entre le client et le serveur est le protocole HTTP (pour *Hypertext Transport Protocol*). Le *protocole HTTP* est un protocole basé texte qui fonctionne au sommet du protocole TCP/IP. L'interopérabilité et la facilité d'utilisation sont les deux caractéristiques importantes de ce protocole de transport [79]. La diffusion de l'information par le Web, tel qu'expliqué, se nomme *Web d'information* (de l'anglais *information Web*).

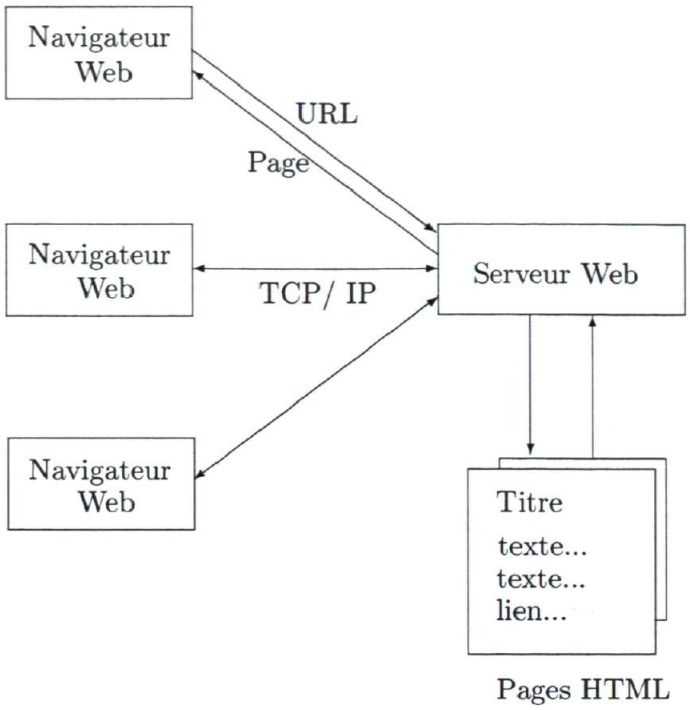

FIGURE 2.1 – Architecture conceptuelle de pages Web.

2.1.3 Applications Web

Une *application Web* se dit d'un logiciel applicatif manipulable grâce à un navigateur Web [2]. Notons que toutes les applications Web partagent le fait d'avoir une structure client-serveur. Une application type client-serveur comprend une partie client, parfois appelée frontale (de l'anglais *front-end*) et une partie serveur, parfois appelée dorsale (de l'anglais *back-end*). La partie client est composée de composants d'IU pour communiquer avec des utilisateurs et la partie serveur contient le reste des composants, c'est-à-dire des composants de logique d'affaires et de bases de données. Contrairement au contenu statique d'une page Web, le contenu d'une application Web est dynamique et change selon la demande du client et les données stockées dans les bases de données. Dans la suite de cette section, nous passons en revue l'évolution de l'architecture logicielle des applications Web.

2. La définition de l'application Web selon le site de Wikipédia.

2.1.3.1 Interface de passerelle commune (CGI)

Bien qu'à ses débuts, le Web a servi de moyens d'affichage des informations statiques stockées sur un serveur Web, l'utilisation de Java et des standards comme CGI [12][3] (pour *Common Gateway Interface*) ont fait évoluer le Web de sorte qu'il est devenu possible de générer des pages Web à l'exécution et de manière dynamique. L'architecture Web basée CGI étend l'architecture de base du Web en fournissant un moyen d'exécuter un programme, afin de générer les pages Web dynamiquement. *CGI* est une des premières approches permettant de développer des pages Web dynamiques. Elle définit un standard pour invoquer des programmes du côté serveur. La génération dynamique d'une page Web se déroule comme suit : un client Web envoie une demande d'exécution d'un programme au serveur Web. À la suite de la réception de la demande, le serveur Web traite la demande et appelle le programme concerné. Les paramètres destinés au programme y sont envoyés via une interface CGI. Les résultats de l'exécution du programme sont des fichiers HTML générés dynamiquement qui sont envoyés au client demandeur.

La figure 2.2 illustre les composants d'une architecture basée CGI. La popularité de CGI s'explique par son indépendance de tout langage et plate-forme logicielle. Il est donc possible d'utiliser tout langage de script-serveur, afin de créer dynamiquement des pages Web. Les langages de script sont très répandus pour la création des programmes CGI. Cependant, des langages de programmation comme C/C++ et Java peuvent également être utilisés pour écrire des programmes CGI.

2.1.3.2 Génération de pages Web dynamiques

Une *page Web dynamique* est une page Web statique qui comprend des parties dynamiques de code. Avant d'envoyer une page Web dynamique au client, le serveur Web doit interpréter et exécuter ses parties dynamiques par un moteur de génération de pages Web dynamiques (de l'anglais *Dynamic Page Generation Engine*). Des moteurs de génération de pages Web sont développés par des fournisseurs des serveurs Web comme *Netscape's LiveWire* ou *Microsoft Active Server Pages*. Le code dynamique comprend généralement certains traitements. Les traitements dans les parties interprétées peuvent varier entre les traitements de base et l'accès aux bases de données et traiter les données accédées. La figure 2.3 illustre les composants d'une architecture de génération de pages

3. La traduction française du terme « Common Gateway Interface» est « Interface de passerelle commune ». Cependant, ce terme est rarement utilisé en pratique. C'est la raison pour laquelle nous favorisons l'utilisation du terme « CGI » dans ce document.

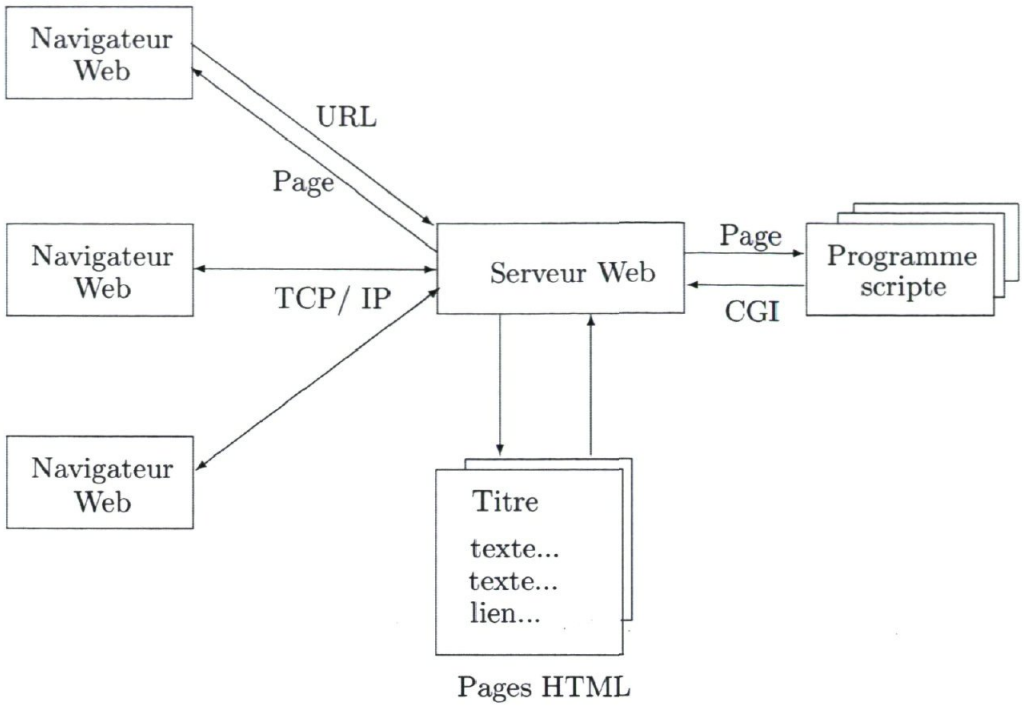

FIGURE 2.2 – Architecture Web basée CGI.

Web dynamiques. Les parties dynamiques des pages Web sont fréquemment écrites avec les langages PHP, JSP ou ASP.

2.1.3.3 Application par composants

L'importance et la nécessité de la réutilisation de code ont amené une nouvelle tendance en développement de logiciels : celle du développement de logiciels par composants. Un *composant* est une partie réutilisable et indépendante de logiciel qui est exposée par une interface standard bien définie. L'idée de base provient des composants matériels prêts à l'emploi (de l'anglais *plug-and-play hardware components*) [83] et l'objectif est de pouvoir concevoir une application en réutilisant les services des composants déjà développés. La réutilisation de code rend le temps de développement plus court, réduit les coûts et étend la durée de vie de l'application.

Par ailleurs, une architecture par composants facilite la communication entre les composants hétérogènes, c'est-à-dire les composants écrits en utilisant différents langages de programmation. Une telle architecture met en place un environnement où les composants peuvent communiquer entre eux, peu importe le langage de programmation employé dans leur développement. Pour ce faire, il convient que tous les composants fournissent une interface conforme aux standards prédéfinis par l'environnement. Les

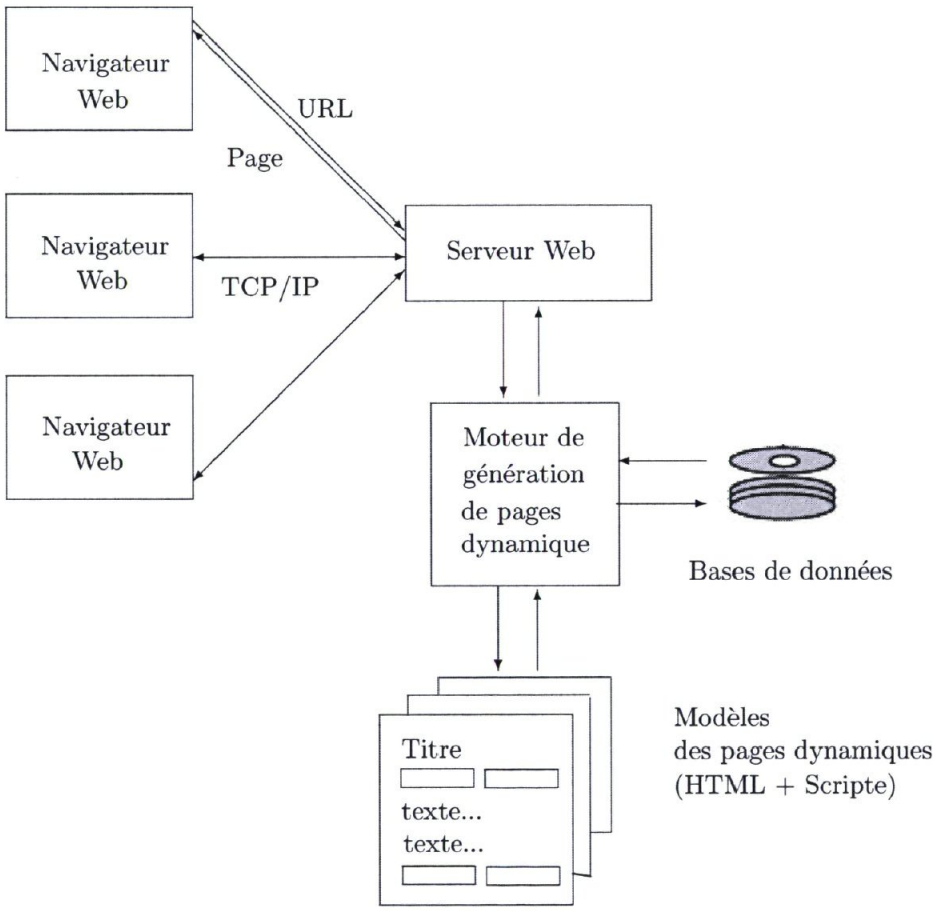

FIGURE 2.3 – Architecture de génération de pages Web dynamique.

courtiers d'objets (de l'anglais *Object Brokers*) simplifient la communication entre composants dans une architecture par composants.

CORBA (pour *Common Object Request Broker Architecture*) offre des spécifications standardisées d'un courtier d'objets. Il a été développé par le groupe OMG (pour *Object Management Group*). Il fait en sorte de faire communiquer de façon transparente divers composants. Nous référons notre lecteur au [31] pour des informations détaillées sur l'élaboration complète de CORBA.

Comme nous verrons à la section 4.2, dans la refonte d'IUs, l'étape de conception descendante consiste à générer de nouvelles IUs dans une technologie de présentation Web. La suite de cette section présente quelques technologies de présentation Web qui pourraient être utilisées dans la conception des nouvelles IUs dans une refonte d'IUs.

2.2 Outils de conception d'IUs Web

On nomme *outils de conception d'IUs* les logiciels ou les bibliothèques de classes qui aident à la conception d'IUs. Dans [30], les outils de conception d'IU se divisent en trois grandes catégories :

- outils de conception d'IUs procéduraux (de l'anglais *procedural UI tools*) ;
- outils de conception d'IUs déclaratifs (de l'anglais *declarative UI tools*) ;
- outils de conception d'IUs hybrides (de l'anglais *hybrid UI tools*).

Un *outil de conception procédural* décrit une librairie de programmation (de l'anglais *programming library*). AWT (pour *Abstract Window Toolkit*) est une bibliothèque graphique pour concevoir des IUs en Java. Cette bibliothèque a été introduite dès les premières versions de Java. Depuis Java 2, la bibliothèque de gestion de fenêtre officielle est Swing. Toutefois, AWT sert encore de fondement à Swing, dans la mesure où de nombreuses classes Swing héritent de classes AWT[4]. Les outils procéduraux sont utilisés par des langage impératifs, tel que Java. Étant donné la nature propriétaire des outils procéduraux, l'utilisation de tels outils est bénéfique seulement si les IUs générées sont destinées à s'exécuter sur des environnements homogènes. Aujourd'hui, avec le grand succès du Web et des appareils mobiles, l'affichage des IUs sur des environnements hétérogènes est rendu indispensable. Il est donc intéressant de pouvoir concevoir des IUs qui fonctionnent sur les différentes plate-formes avec peu de modifications. C'est la raison pour laquelle, les outils de conception déclaratifs sont devenus de plus en plus répandus.

4. Cette définition provient du site de Wikipédia.

Comparativement aux outils de conception procéduraux, les *outils de conception déclaratifs* possèdent un niveau sémantique plus élevé. Ils sont donc généralement plus faciles à apprendre et à utiliser. Le langage HTML (pour *HyperText Markup Language*) est un exemple des outils de conception déclaratifs [5].

Quant aux outils de conception d'IU hybride, ils tirent profit des propriétés des deux catégories d'outils, précédemment mentionnés.

Dans la suite de cette section, nous présentons quelques outils de conception d'IU utilisés dans le contexte du Web.

2.2.1 Applet Java

Une *applet* [3] est un code compilé en format bytecode Java qui s'intègre et s'exécute dans un navigateur Web, par le biais d'une machine virtuelle Java (de l'anglais *Java virtual machine* ou JVM). L'utilisation du langage Java dans le développement des IUs permet de concevoir des IUs complexes et de faire de l'animation sur les navigateurs Web hébergant une JVM.

Une applet Java est une solution procédurale de conception d'IUs où l'environnement de développement Java offre une boîte à outils complète pour le développement d'IUs. De plus, l'utilisation de Java comme langage de programmation permet de concevoir du code modulaire et réutilisable. Cependant, la courbe d'apprentissage du langage est généralement longue. De plus, tel que mentionné, les applets Java nécessitent un environnement spécial - JVM - pour s'exécuter sur les navigateurs Web.

2.2.2 Outils Web basés Java

L'idée de base dans la conception des *outils Web basés Java* était d'offrir un même environnement pour le développement des IUs et pour le développement des autres composants d'une application. Le fait que les IUs soient développées en Java permet de pouvoir les déboguer plus facilement. Cela est considéré comme un des avantages importants des outils Web basés sur Java, car en raison du manque d'outils de débogage, le débogage des langages de programmation script comme JavaScript est une tâche chronophage (de l'anglais *time consuming*).

5. Dans la littérature, ces outils de conception sont désignés aussi par langages descriptifs d'IUs (pour *User Interface Description Languages* ou UIDL).

GWT (pour *Google Web Toolkit*) [6] et la plate-forme Echo3 [7] sont deux exemples de ce type d'outils. Les outils Web basés sur Java se classifient dans les outils procéduraux et partagent sensiblement les mêmes avantages et les mêmes inconvénients que nous avons énumérés pour les applets Java. Les outils Web basés Java se différencient toutefois par le fait que leur code est compilé en DHTML ou Ajax (voir la section 2.2.6), avant de s'exécuter dans un navigateur Web. Cela élimine le besoin d'héberger une JVM sur le navigateur Web.

2.2.3 HTML

HTML (pour *HyperText Markup Language*) est le langage de présentation le plus utilisé sur le Web. HTML est la version simplifiée d'un méta-langage de balisage, appelé SGML (pour *Standard Generalized Markup Language*). Un méta-langage de balisage permet de définir d'autres langages de balisage de façon formelle. L'idée de base dans la conception des langages de balisages était de rajouter de l'information supplémentaire aux documents textes de manière à ce que leur structure soit compréhensive et traitable par les ordinateurs. XML et HTML sont deux exemples de langages de balisage dérivant du langage SGML. HTML présente une série de balises standardisées permettant de définir la structure, le contenu et la présentation des pages Web.

Puisqu'à la base, le langage HTML est défini pour structurer le contenu de pages Web, les aspects visuels de contenu n'ont pas été pris en considération lors de la définition du langage. Cependant, le succès rapide du Web a créé le besoin d'avoir des pages Web plus riches en termes d'aspects visuels. De ce fait, les fournisseurs des navigateurs Web ont commencé à développer et à intégrer leurs propres balises dans le langage HTML pour améliorer la présentation de contenu Web.

La dernière version standardisée de HTML est HTML4. Ce dernier a introduit la notion de *feuilles de style en cascade* (de l'anglais *Cascading Style Sheets* ou CSS). Grâce à des feuilles de style en cascade, il est possible de définir avec exactitude la police, la taille, la couleur et la disposition des éléments d'une page Web les uns par rapport aux autres. Ces feuilles de style permettent de séparer la présentation du contenu d'une page Web. Ainsi, il est possible de partager une seule feuille de style entre plusieurs fichiers HTML. De plus, l'analyse d'une page HTML qui ne contient pas de balises de présentation s'avère plus facile.

6. Google Web Toolkit est disponible sur `https://developers.google.com/web-toolkit`
7. Echo3 est disponible à l'adresse suivante : `http://echo.nextapp.com/site/echo3`

2.2.4 DHTML

Le DHTML (Pour *Dynamic HyperText Markup Language*) ou encore HTML dynamique est une extension du langage HTML. Le DHTML est un ensemble de technologies Internet associées afin de fournir des pages HTML plus interactives. Le *DHTML* a été introduit pour pallier les problèmes d'utilisabilité et d'extensibilité des pages HTML, il représente un outil hybride qui bénéficie des avantages des outils procéduraux et déclaratifs à la fois. Alors qu'il est nécessaire de rafraichir une page HTML à partir du serveur Web pour tout changement de contenu, le contenu d'une page DHTML peut être modifié après le chargement, grâce à des événements - mouvements de la souris, survol d'un objet par le curseur, etc. - sans nécessiter un rafraichissement. Tel qu'il est illustré à la figure 2.4, les pages Web DHTML ont la possibilité de contenir certaines règles d'affaires - règles de validation des éléments de l'IU, par exemple.

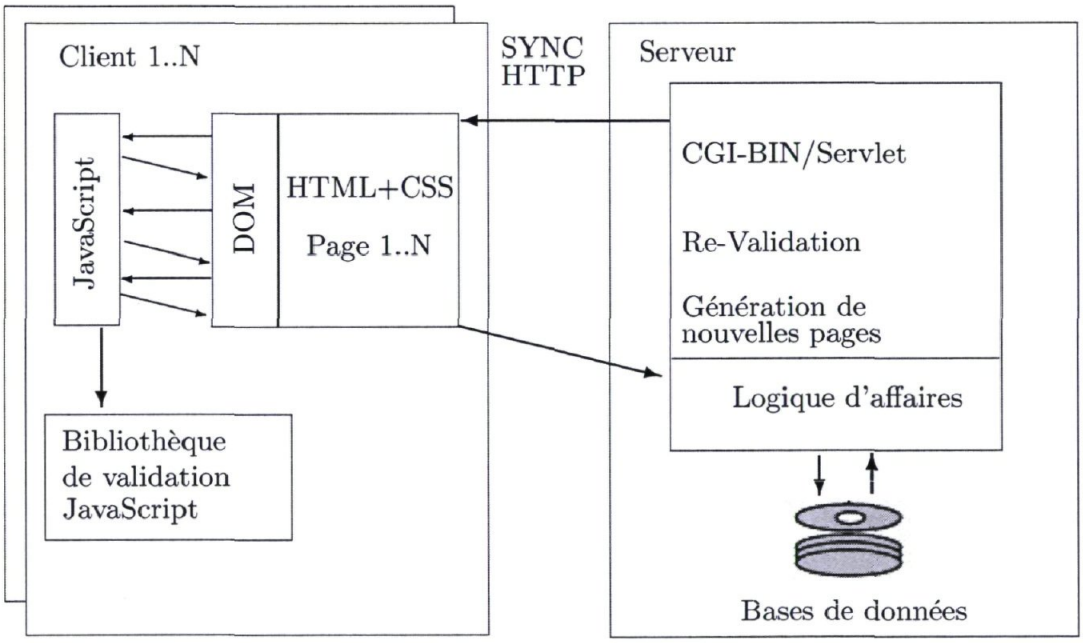

FIGURE 2.4 – Architecture des pages DHTML.

Le DHTML met en pratique les technologies suivantes :

- (X)HTML - les balises HTML permettent de créer des pages Web et leurs éléments pour l'utilisation des feuilles CSS et le modèle objet de document ;
- le modèle objet de document (de l'anglais *Document Object Model* ou DOM) - DOM permet aux divers éléments HTML d'une page d'être manipulés par des langages de script tels que, le JavaScript ;
- les feuilles de styles (de l'anglais *Cascading Style Sheets* ou CSS) - les feuilles CSS augmentent le contrôle et la disposition du contenu des pages Web. Elles

permettent, entre autres, la gestion des polices de caractères et de l'espacement ;
- JavaScript - JavaScript est un langage de scripte qui permet de définir des événements utilisateurs.

2.2.5 XML

XML (pour *Extensible Markup Language*)[8] est un outil de conception d'IUs déclaratif. À l'instar de HTML, XML dérive du langage de balisage SGML. Cette syntaxe est dite extensible, car elle permet de définir différents espaces de noms, c'est-à-dire des langages avec chacun leur vocabulaire et leur grammaire, comme XHTML, XSLT, RSS, SVG. L'objectif de XML était de définir un langage aussi générique que SGML, mais plus simple à utiliser et à comprendre. Il est possible, dans XML, de rajouter des balises.

XSL (pour *eXtensible StyleSheet Language*) [81] est un langage recommandé par le W3C (pour *World Wide Web Consortium*) pour effectuer la représentation des données de documents XML. XSL est lui-même défini avec le formalisme XML ; cela signifie qu'une feuille de style XSL est un document XML bien formé. XSL est un langage permettant de définir des feuilles de style pour les documents XML au même titre que les CSS (pour *Cascading StyleSheets*) pour le langage HTML. Contrairement aux CSS, XSL permet de retraiter un document XML afin d'en modifier totalement sa structure, ce qui permet, à partir d'un document XML, d'être capable de générer d'autres types de documents (PostScript, HTML, RTF, etc.) ou bien un fichier XML de structure différente. Ainsi, la structuration des données (définie par XML) et leur représentation (définie par un langage tel que XSL) sont séparées. Il est donc possible, à partir d'un document XML, de créer des documents utilisant différentes représentations (HTML pour créer des pages web, WML (pour *Wireless Markup Language*) pour les mobiles WAP (pour *Wireless Application Protocol*), etc.). Cela permet de réutiliser les IUs générées en XML sur différentes plate-formes.

2.2.6 Ajax

Ajax (pour *Asynchronous JavaScript and XML*) est une extension de DHTML pour la conception des applications Web interactives [29]. En plus des technologies (X)HTML, CSS, DOM et JavaScript, utilisées par DHTML, Ajax fait l'usage de XMLHttpRequest

8. Pour avoir plus d'information, nous invitons notre lecteur à consulter :`http://www.ibm.com/developerworks/xml/newto`

afin d'effectuer les appels asynchrones aux serveurs Web. Ajax a été introduit pour remplir l'écart d'interactivité entre des applications Web et des applications de bureau (de l'anglais *desktop application*). Les technologies employées dans une application Ajax font en sorte de cacher les délais des interactions entre le client et le serveur Web des utilisateurs.

Dans une application Web traditionnelle, les interactions entre le client et le serveur Web se font de façon synchrone. C'est-à-dire qu'une grande partie des interactions d'un utilisateur avec l'application déclenche une requête HTTP vers le serveur Web. À la suite de la réception d'une requête, le serveur Web la traite et envoie une réponse au client Web, qui à son tour, affiche la réponse obtenue. Chaque traitement de requête, du côté serveur, entraine des délais et pendant ces délais, l'utilisateur n'a qu'à attendre que le traitement se termine et que la page se rafraichisse complètement.

Pour diminuer ces temps d'attente, Ajax permet d'effectuer des interactions asynchrones. Dans ce contexte, les requêtes HTTP se déclenchent par le biais du moteur Ajax qui a pour fonction de générer des requêtes HTTP en JavaScript (XMLHttpRequest) et de les envoyer au serveur Web, sans que l'utilisateur n'ait fait la demande. Autrement dit, le moteur Ajax récupère progressivement du serveur toutes les informations nécessaires du côté client. Cette récupération se fait de manière asynchrone pendant que l'utilisateur est en train de travailler avec l'application. La figure 2.5 compare le fonctionnement d'une application Web traditionnelle et une application Ajax.

Google Suggest [9] et Google Maps [10] sont deux exemples de projets Ajax réalisés par la compagnie Google.

2.2.7 HTML5

HTML5 (pour *HyperText Markup Language 5*) est un outil déclaratif de conception d'IUs. Il est la prochaine révision majeure de HTML - format de données conçu pour représenter les pages web. Cette version est en développement en 2012. HTML5 spécifie deux syntaxes d'un modèle abstrait défini en termes de DOM : HTML5 et XHTML5. Le langage comprend également une couche application avec de nombreuses API, ainsi qu'un algorithme afin de pouvoir traiter les documents à la syntaxe non conforme. Le travail a été repris par le W3C en mars 2007 après avoir été lancé par le WHATWG (pour *Web Hypertext Application Technology Working Group*). Les deux organisations

9. Google Suggest est disponible à l'adresse : `https://www.google.com/webhp?complete=1`
10. Google Maps est disponible à l'adresse : `https://maps.google.com/`

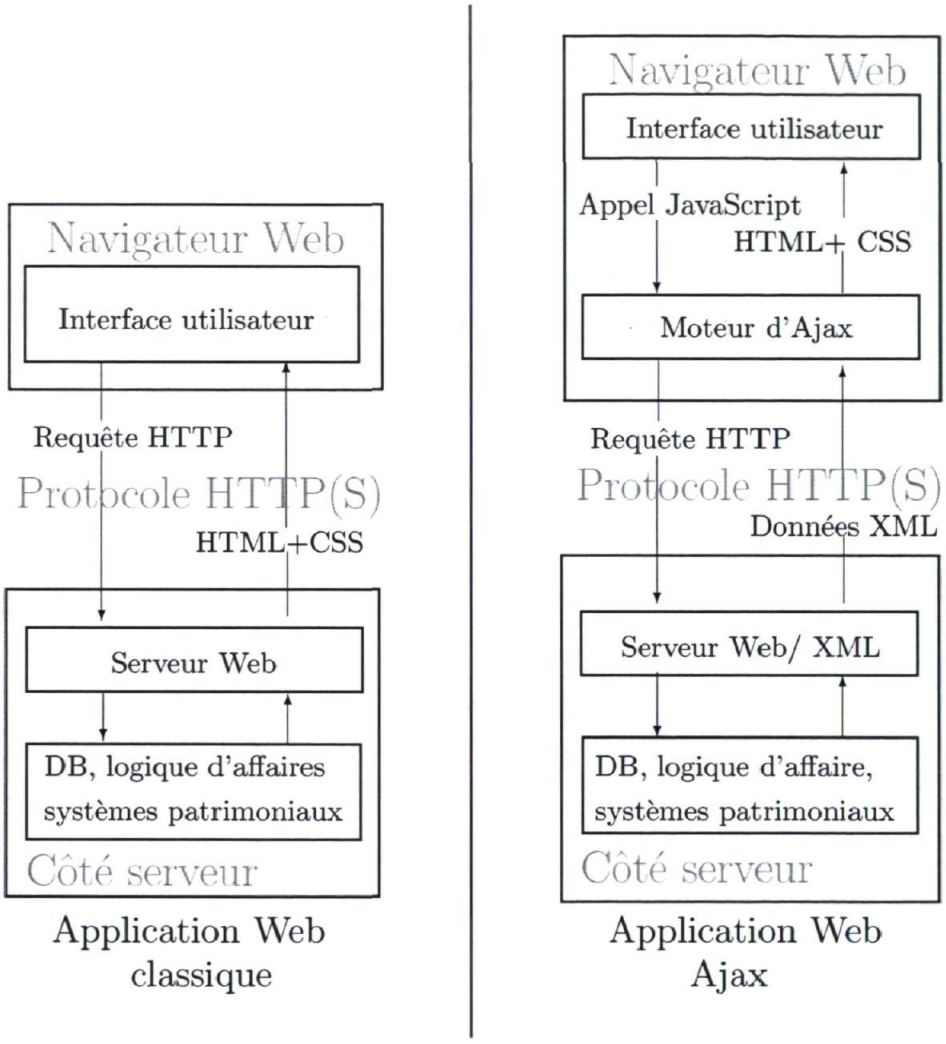

FIGURE 2.5 – Comparaison de l'architecture d'une application Web classique et d'une application Ajax [1].

travaillent en parallèle sur le même document afin de maintenir une version unique de la technologie. Le W3C vise la clôture des ajouts de fonctionnalités le 22 mai 2011 et une finalisation de la spécification en 2014, et encourage les développeurs Web à utiliser HTML 5 dès maintenant [11].

Les développeurs de l'HTML5 tentent de réunir la syntaxe des langages HTML4.0 et XHTML1.1 dans un langage déclaratif et simple à lire pour les humains et pour les ordinateurs - les analyseurs, les navigateurs Web, etc. De plus, la possibilité d'intégrer le multimédia est aussi considérée dans le développement du langage, et ce, sans la nécessité de télécharger des modules d'extension.

2.3 Conclusion

Ce chapitre a donc fourni des informations concernant l'environnement cible d'une migration d'IUs vers le Web. Nous avons brièvement présenté des différentes technologies de présentation Web - outils de conception d'IUs - dans lesquelles, il est possible de développer de nouvelles IUs. Parmi les outils présentés, l'utilisation des outils déclaratifs est plus avantageuse dans la conception d'IUs, car les IUs conçues à l'aide de ces outils peuvent être utilisées sur différentes plate-formes avec peu de modifications.

Maintenant, que nous connaissons l'environnement de départ et l'environnement cible d'une migration d'IUs textuelles vers le Web, nous pouvons aborder le sujet, lui-même. Pour ce faire, nous regroupons les techniques de migration d'IUs dans deux classes d'approches que nous présentons dans le chapitre suivant. Pour chaque classe d'approches, quelques exemples de techniques viendront éclaircir les notions définies. Nous clôturons nos discussions par l'énumération des avantages et des inconvénients de chaque classe d'approches.

11. Cette définition provient du site de Wikipédia

Chapitre 3

Classement des techniques de migration d'IUs textuelles vers le Web

> By the time you will finish your thesis, the systems, tools and/or prototypes that you have developed will be legacy systems and you will need to reverse engineer them in order to understand, migrate and/or reengineer them.
>
> Anonymous

La migration des IUs d'un système patrimonial vers le Web fait souvent partie de la migration complète du système, laquelle aboutit généralement à la conception d'une *application Web client léger* dont la partie client n'exécute aucune règle d'affaires. Tel que mentionné à la section 1.1.2.1, étant donné l'architecture semi-décomposable ou non décomposable de la plupart des systèmes d'information patrimoniaux et l'analogie de fonctionnement entre ces systèmes et les applications Web client léger, ce choix de conception semble être pertinent (voir la page 14).

Dans une application Web client léger, le code des IUs est séparé de celui des autres composants. Comme stipulé au chapitre précédent, les IUs sont développées à l'aide des technologies de présentation Web, à savoir : HTML, XML, XAML, JavaScript, etc. et sont destinées à s'intégrer à des navigateurs Web. Les autres composants, quant à eux, sont développés en utilisant des technologies et des standards côté serveur (de l'anglais *server side technology*) telles que, ASP (pour *Active Server Pages*), ASP.NET, ColdFusion, JSP (pour *JavaServer Pages*), PHP (pour *Hypertext Preprocessor*), Python, Ruby

on Rails, etc. et sont exécutés par des serveurs Web ou sur des serveurs d'applications. Le constat de ces différences, tant au niveau de la technologie de développement que de la plate-forme d'exécution, nous convainc que la migration d'IUs textuelles de systèmes patrimoniaux vers le Web peut être étudiée comme une étape indépendante du processus de migration de systèmes. Néanmoins, l'étude de plusieurs travaux de recherche dans le domaine de la migration de systèmes patrimoniaux nous révèle que la majorité d'entre eux ne traite pas le sujet de migration d'IUs vers des plate-formes graphiques en général et vers le Web en particulier, de manière distincte du processus de migration de systèmes.

Dans le cadre de nos recherches sur la migration d'IUs, nous nous mettons au défi de rassembler et de présenter les méthodes et les techniques de migration d'IUs vers le Web, proposées dans la littérature [1]. Puisque, la présentation de ces techniques, de manière structurée, nécessite un regroupement de ces dernières, nous les classons en deux classes d'approches : la classe d'approches frontales et la classe d'approches de greffage. Ce chapitre expose notre classement des techniques de migration d'IUs vers le Web. À la fin du chapitre, nous résumons nos discussions en citant les avantages et les inconvénients de chaque classe d'approches.

3.1 Techniques frontales

Dans la migration d'IUs textuelles vers le Web, une *technique frontale* cherche à fournir un accès Web aux utilisateurs d'un système patrimonial et, en même temps, à conserver les IUs textuelles ainsi que les autres composants du système patrimonial sur leurs environnements originels - tel qu'illustré à la figure 3.1. Le résultat est donc une application client-serveur Web dans laquelle l'intégrité du système patrimonial constitue un des composants principaux - souvent le serveur d'applications.

La mise en place d'une technique frontale est généralement simple et peu coûteuse. La simplicité de l'implémentation fait en sorte que les techniques frontales ont été principalement pratiquées par l'industrie avant d'intéresser les chercheurs universitaires [9, 23].

1. Notons que dans le cadre du projet Soare (pour *State of the Art in Re(verse) Engineering and Transcoding of UIs*) les chercheurs du laboratoire LILab (Pour *Louvain Interaction Laboratory*) à l'Université Catholique de Louvain en Belgique ont également présenté des travaux de recherche sur la migration d'IUs [71]. Cependant, cette présentation ne traite pas un type particulier de migration d'IUs ; elle concerne plutôt la migration d'IUs de manière générale.

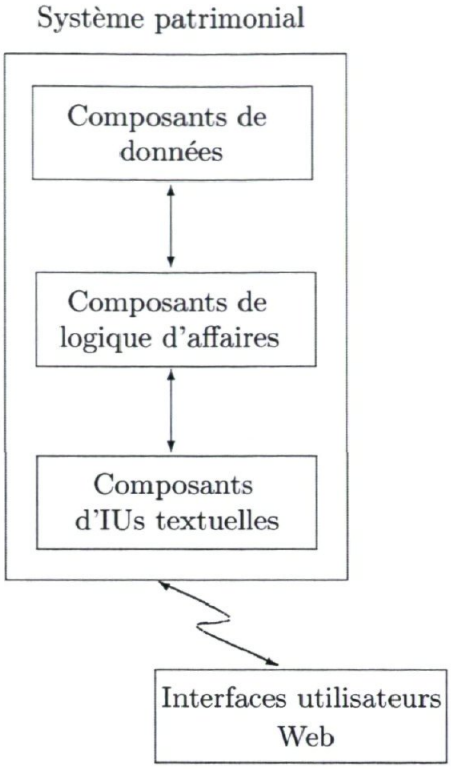

FIGURE 3.1 – Classe d'approches frontales.

Dans la migration des IUs textuelles d'un système, lorsque l'on opte pour une des techniques frontales, une condition parmi les conditions suivantes ou une combinaison d'entre elles sont remplies :

- on cherche un moyen rapide et peu coûteux pour accéder aux IUs textuelles du système via le Web ;

- les langages de programmation utilisés pour développer les différents composants du système patrimonial sont acceptés par l'organisation propriétaire du système et ses employés. Par exemple, l'organisation dispose de développeurs Cobol afin de maintenir des programmes patrimoniaux ;

- le fabriquant du matériel du système offre le support technique nécessaire pour le matériel en marche ;

- la performance, la fiabilité (de l'anglais *reliability*) et l'extensibilité (de l'anglais *scalability*) du système patrimonial sont satisfaisantes et compensent la maintenance coûteuse du matériel [24] ;

- les fonctions du système patrimonial sont critiques et on désire éviter toute modi-

fication du code source qui pourrait éventuellement perturber le fonctionnement du système ;

- aucun changement majeur de fonctionnalité n'est espéré [24] ;
- le système patrimonial possède une architecture non décomposable et les coûts et les risques associés à la séparation des IUs ne sont pas justifiables ;
- le code source du système patrimonial n'est pas disponible.

L'émulation d'hôte sur le Web, la capture d'écrans et le mappage d'écrans sont trois techniques, principalement employées par l'industrie, qui se situent dans la classe d'approches frontales. Dans la suite de ce chapitre, nous nous attardons sur la définition et sur les particularités de chacune de ces techniques.

3.1.1 Émulation d'hôte sur le Web

Tel qu'expliqué à la page 9, dans un système transactionnel, les interactions entre le système hébergé sur l'ordinateur central et les utilisateurs se font via des terminaux muets. Bien qu'adéquats et utiles à leur époque, ces terminaux sont jugés inefficaces et peu qualifiés après l'arrivée des ordinateurs personnels (de l'anglais *Personal Computer* ou PC). C'est la raison pour laquelle, dans l'architecture matériel des systèmes transactionnels, des PC ont progressivement remplacé des terminaux. Dans la nouvelle architecture, pour que la communication entre l'ordinateur central et les PC puisse continuer de fonctionner de façon transparente et sans la nécessité d'altérer le système, la notion d'émulateur de terminal a été introduite. Un *émulateur de terminal* est un logiciel qui s'installe sur l'ordinateur personnel, remplaçant du terminal. L'émulateur a comme fonction de simuler le fonctionnement du terminal ainsi que le protocole de communication entre l'ordinateur central et le terminal. Ainsi, l'émulateur est en mesure de prendre les flots de données sortants (de l'anglais *outpout data stream*) (voir la page 14) provenant de l'ordinateur central, et de les afficher comme les afficherait un terminal. On nomme cette technique d'émulation qui préserve tous les aspects visuels et les fonctionnalités des IUs originelles l'*émulation d'hôte.*

L'*émulation d'hôte sur le Web* (de l'anglais *Webulation*) est une extension de l'émulation d'hôte. Elle donne donc un accès Web aux IUs textuelles d'un système patrimonial, sans nécessiter la refonte des IUs dans une technologie de présentation Web. La seule nouveauté est que, dans l'émulation d'hôte sur le Web, l'émulateur s'exécute soit sur un navigateur Web, soit sur un serveur Web [23].

La figure 3.2 illustre une émulation d'hôte sur le Web dans laquelle l'émulateur est un module d'extension (de l'anglais *plug-in*) qui - à la demande de l'utilisateur - s'installe sur un navigateur. Il s'agit d'une applet Java (voir la section 2.2.1) qui est téléchargée et exécutée sur l'environnement d'exécution Java (de l'anglais *Java Runtime Environnement* ou *JRE*), intégré au navigateur Web. Pour la communication de données, l'applet utilise le protocole TELNET (pour *TELecommunication NETwork*) : elle établit une connexion TCP avec un serveur TELNET et lui envoie les données saisies par l'utilisateur en format ASCII - texte en clair. Le serveur TELNET reçoit les données et les communique au système patrimonial hébergé sur l'ordinateur central, à l'aide du protocole de communication originel - SNA (pour *System Network Architecture*). La communication dans le sens inverse se fait de la même manière. Cet exemple ainsi que quelques autres exemples de l'émulation d'hôte sur le Web sont présentés dans [76].

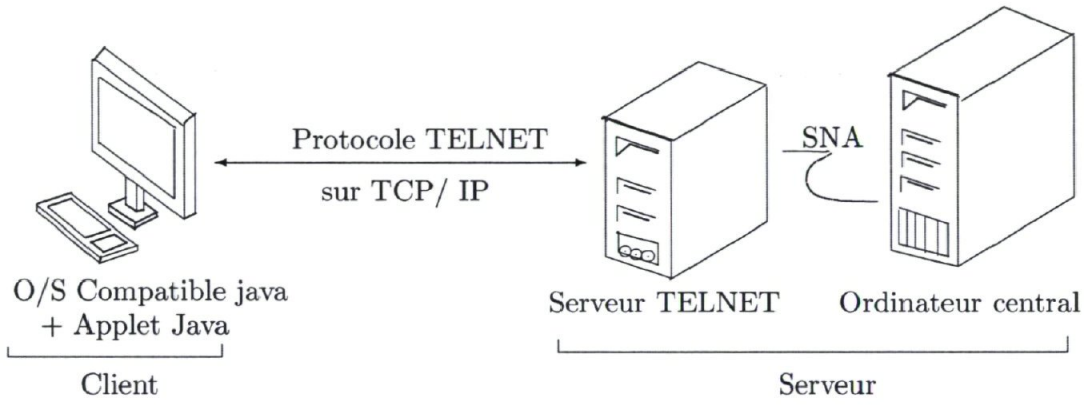

FIGURE 3.2 – Émulation d'hôte sur le Web.

Comme nous l'avons mentionné, l'émulation d'hôte sur le Web n'implique pas la refonte des IUs textuelles du système à l'étude. Par conséquent, malgré le fait que cette technique permet d'intégrer un système patrimonial au Web, elle ne s'inclut que rarement parmi les techniques de migration d'IUs.

3.1.2 Capture d'écrans

La technique de la *capture d'écrans*[2](de l'anglais *screen scraping*, *screen scratching* [66] ou encore *refacing*) est une extension de l'émulation d'hôte sur le Web pour accéder aux IUs d'un système patrimonial via le Web. Il s'agit d'une technique frontale pour la refonte d'IUs textuelles, en utilisant des technologies de présentation Web et des éléments graphiques dont ces technologies disposent.

2. Traduction du terme *screen scraping* selon l'Office québécois de la langue française (OQLF).

La technique de capture d'écrans conserve une relation un-à-un entre les IUs textuelles et les IUs Web correspondant à ces dernières, c'est-à-dire qu'une IU Web représente une seule IU textuelle du système patrimonial [23]. Étant donné que cette technique ne vise pas à modifier le code source du système à l'étude, il s'avère nécessaire d'avoir recours aux services d'un émulateur de terminal afin de communiquer avec l'ordinateur central et de récupérer des flots de données sortants. Ceux-ci fournissent les informations nécessaires pour la refonte des IUs textuelles dans la technologie de présentation de choix. Dans ce contexte, la refonte des IUs textuelles se fait à l'exécution, soit à la volée, soit en générant une interface personnalisée pour chaque IU textuelle.

Pour la refonte d'une IU textuelle à la volée, il convient de connaitre les éléments textuels qu'un flot de données sortant peut contenir et de déterminer les éléments graphiques de la technologie de présentation Web, qui peuvent leur correspondre. Il faut ensuite définir un ensemble de règles de traduction. Ces règles permettent de traduire tous les éléments de l'IU à l'exécution. Le processus de refonte - si l'émulateur réside sur le serveur Web - se déroule comme suit :

- l'utilisateur demande l'affichage d'une IU à l'aide de son URL ;
- le serveur Web communique cette demande à l'émulateur ;
- l'émulateur transforme la demande HTTP en une demande d'une IU textuelle et la communique à l'ordinateur central ;
- l'émulateur récupère ensuite le flot de données sortant représentant l'IU textuelle demandée ;
- sur le serveur Web, on applique les règles de traduction sur chacun des éléments du flot de données reçu de l'ordinateur central.

C'est ainsi que l'on obtient l'équivalent graphique de l'IU textuelle, laquelle est envoyée au client Web - navigateur - pour être affichée. Notons que les IUs Web générées à la volée se composent des éléments graphiques de base de la technologie de présentation pour laquelle les IUs textuelles sont interprétées.

Prenons l'exemple de la refonte d'une IU textuelle - présentée sous forme d'un flot de données sortant destiné à un terminal IBM 3270 - sur une page HTML. L'application de la technique de capture d'écrans sur un tel flot de données, transforme les champs de saisie en zones de texte (de l'anglais *text box*). Quant aux champs de sortie protégés en écriture (de l'anglais *output fields*), ils se transforment en des étiquettes (de l'anglais *label*). De même, il est possible de transformer les touches de fonction (de l'anglais *function key*) - $F1, ..., F24$ - disponibles sur toutes IUs textuelles d'un terminal 3270 en boutons [23].

Rappelons que, pour un système transactionnel muni d'un moniteur de télétraitement, les données qui se trouvent dans une description d'écran fournissent de l'information sur les aspects visuels de l'IU textuelle correspondant. Il est donc possible, dans la technique de capture d'écrans, de remplacer les descriptions d'écran du système par les IUs Web en HTML ou en XML. Cela a comme avantage de pouvoir enlever l'implication du moniteur de télétraitement sur la nouvelle structure de migration. Néanmoins, dans ce contexte, en plus d'émuler le fonctionnement du terminal, celui du moniteur de télétraitement doit aussi être émulé. Considérant toutes les tâches d'un moniteur de télétraitement dans un système transactionnel (voir la section 1.1.2.1), son émulation n'est pas triviale [66].

La figure 3.3 est un exemple de l'utilisation de la technique de capture d'écrans à la volée pour la migration d'une IU textuelle. L'image en avant-plan correspond à une IU textuelle dans un système IBM et l'image en arrière-plan est la conversion de cette IU sur le Web. Comme nous pouvons le constater, la migration a permis d'améliorer l'aspect et la convivialité de l'IU patrimoniale, de façon limitée. Par exemple, sur la nouvelle IU, il est possible de se servir du lien « *Home* », dans le coin supérieur droit, afin de revenir à la page principale de l'application ; ou encore de se servir du lien « *Site Map* » (du français carte de site) afin de consulter toutes les pages de l'application en une seule clique[3].

La passerelle de transaction CICS (de l'anglais *CICS transaction gateway*) est un produit commercial de la compagnie IBM qui, en utilisant la technique de capture d'écrans, permet aux clients d'IBM d'accéder à leurs systèmes transactionnels CICS à partir des navigateurs Web [14].

Comme nous l'avons vu, la refonte d'IUs dans la technique de capture d'écrans est une transformation élémentaire d'IUs textuelles, laquelle n'apporte pas de valeur ajoutée aux aspects visuels des IUs. Quant à la capture d'écrans personnalisée d'IUs, elle tente de bonifier l'aspect et la convivialité des IUs textuelles sur la technologie de présentation, et ce, à l'aide d'éléments graphiques plus élaborés - logos, cases à cocher, cases d'option, listes déroulantes, etc. Pour ce faire, la migration des IUs d'un système utilisant la génération personnalisée d'écran impose que l'on personnalise (de l'anglais *customize*) la refonte de chaque IU textuelle séparément. Dans l'industrie, cela est fait souvent par des développeurs, de façon manuelle. Ainsi, des développeurs se basent sur l'apparence visuelle des IUs textuelles pour concevoir les IUs Web personnalisées qui leur correspondent. Puisque la personnalisation de chaque IU se fait manuellement, il est à la discrétion des développeurs de trouver des éléments graphiques de la technologie

3. Ces exemples nous sont fournis par Fujitsu Canada à Québec. Nous avons obtenu leur autorisation pour inclure ces exemples dans ce document.

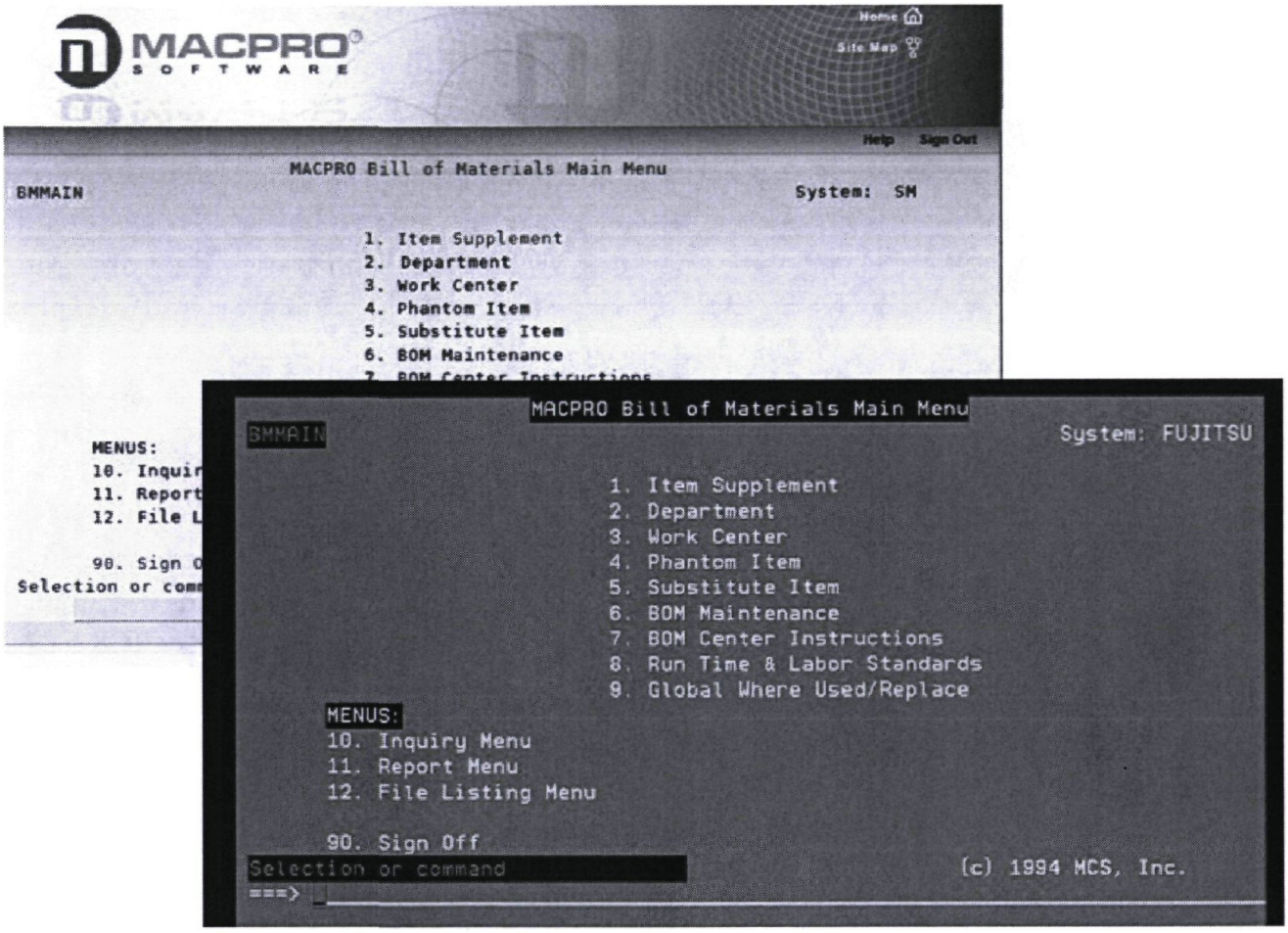

FIGURE 3.3 – Un exemple de migration d'IUs à l'aide de la technique de capture d'écrans.

de présentation qui correspondent le mieux aux éléments de l'IU faisant l'objet de la refonte. Une fois que la refonte de toutes les IUs est terminée, les nouvelles IUs sont répertoriées et conservées sur un serveur Web. À l'exécution, en recevant un flot de données sortant sur le serveur Web, on identifie l'IU textuelle, l'objet de l'affichage, et on la remplace pas son équivalent graphique. Le résultat est ensuite envoyé au navigateur Web [24].

3.1.3 Mappage d'écrans

Le *mappage d'écrans* (de l'anglais *screen mapping*) est une pratique de refonte d'un sous-ensemble des IUs textuelles d'un système patrimonial pour générer une IU Web basée sur la tâche (de l'anglais *task-oriented Web-based GUI*). Le mappage d'écrans est une extension de la capture d'écrans. Il vise donc à minimiser les modifications du code source d'un système patrimonial, y compris ses IUs.

Contrairement à l'émulation d'hôte et à la capture d'écrans, le mappage d'écrans crée une relation plusieurs-à-un entre les IUs textuelles et l'IU Web générée correspondant à ces dernières. Ainsi, il est possible de réunir les IUs textuelles ayant une relation logique entre elles - les IUs qui concernent les tâches d'un utilisateur - dans une seule IU Web. En bonifiant la convivialité des IUs, la technique du mappage d'écrans offre une meilleure expérience utilisateur. Dans l'industrie, la refonte des IUs textuelles est réalisée, encore une fois, par des analystes et des développeurs, de façon manuelle. Étant donné la grande taille de la plupart des systèmes patrimoniaux, la réalisation manuelle de refonte des IUs textuelles est très longue et n'est donc pas économiquement justifiable. C'est la raison pour laquelle, au niveau universitaire, certains chercheurs ont proposé d'automatiser la technique du mappage d'écrans au niveau de la refonte des IUs. Nous présentons, à la section 5.3, un exemple de ces travaux de recherche réalisé par la professeure Eleni Stroulia et ses collègues, à l'Université d'Alberta [74]. Les techniques et les étapes de refonte automatisée, quant à elles, sont exposées au chapitre 4.

3.2 Techniques de greffage

Dans la migration d'IUs textuelles de systèmes patrimoniaux vers le Web, les techniques et les méthodes qui visent à intégrer un système patrimonial au Web et, en même temps, à remplacer les composants d'IU d'un système patrimonial par des IUs Web se classifient dans la classe d'approches de *greffage* [11, 25, 43, 50, 52]. Tel qu'illustré à la

figure 3.4, l'objectif de la refonte d'IUs dans une telle migration est à la fois la séparation des IUs d'un système patrimonial des autres composants du système et la refonte de ces dernières dans une technologie de présentation Web.

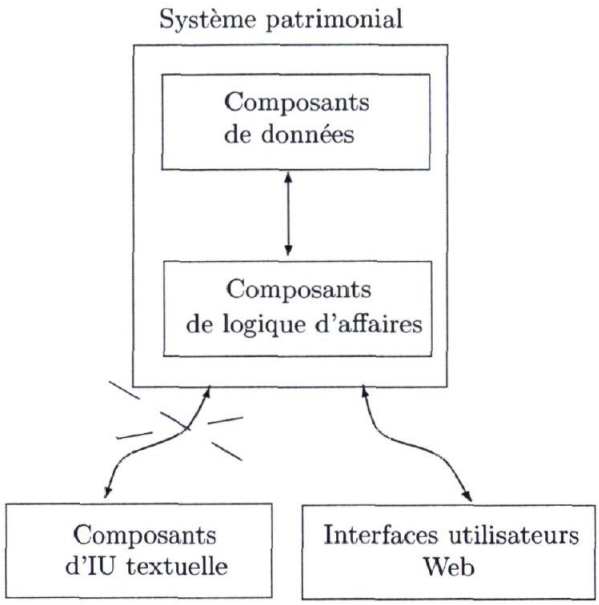

FIGURE 3.4 – Classe d'approches de greffage.

Dans le chapitre 5, nous étudierons en détail deux plate-formes qui utilisent des techniques de greffage dans la migration d'IUs. Ces plate-formes ont été proposées par Melody Moore [50] et Merlo et al. [44].

Nous avons mentionné à la section 1.2 que l'architecture relativement modulaire des systèmes patrimoniaux semi-décomposables et décomposables permet de séparer leurs composants d'IU plus facilement. La séparation du code des IUs dans un système patrimonial est en effet la *décomposition* du système au niveau de la logique d'affaires.

La complexité de la décomposition d'un système patrimonial est directement liée au degré de la décomposabilité de son architecture [10]. Bien qu'en théorie, il est possible de décomposer tout système patrimonial, en pratique, la décomposition d'un système non décomposable, en raison des risques et des coûts importants que cela engendre, n'est souvent pas réalisable [11]. Considérant cela, la séparation des composants d'IU des autres composants dans un système n'est justifiée que pour la migration des IUs des systèmes semi-décomposables ou décomposables. Ajoutons que, même dans un système patrimonial modulaire, il est rare d'identifier une frontière claire entre le code des composants d'IU et celui des autres composants. C'est la raison pour laquelle, avant la décomposition d'un tel système, il est souvent nécessaire de procéder à la restruc-

turation du code source pour permettre de séparer le code des composants d'IU sans perturber le fonctionnement du système [57].

La réalisation manuelle de la séparation des composants d'IU nécessite qu'un expert du domaine analyse tout le code source du système pour en extraire le code des composants d'IU. Cependant, étant donné la grande taille de la plupart des systèmes patrimoniaux, une telle séparation est pratiquement impossible. C'est pourquoi, dans la littérature, on constate que le processus de séparation d'IUs est souvent semi-automatisé et, dans certains cas, complètement automatisé. Les techniques de *slicing* de code figurent parmi les techniques d'automatisation de la décomposition de systèmes patrimoniaux. Dans [4, 8, 11], des algorithmes de *slicing* [78], basés sur des graphes de dépendance de contrôle, sont utilisés afin d'extraire le code des composants d'IU de systèmes patrimoniaux.

Le slicing traditionnel sert à examiner des tranches sélectionnées du code d'un programme (*slices*), afin d'analyser leurs comportements et de récupérer les décisions de conception. Ces tranches concernent un type ou un objet-données (de l'anglais *object data*) en particulier et contiennent les traces des manipulations de l'objet-données, tout au long du programme. Les outils de *slicing* représentent généralement les résultats de l'analyse des tranches en graphes de flot de contrôle ou de flot de données. Cela permet de reproduire les tranches. Le *slicing* est notamment utilisé pour la compréhension de programmes, les métriques de qualité de programme, l'analyse de portabilité, la parallélisation, etc. Il se révèle aussi d'une grande utilité en ingénierie-inverse [4].

Tel qu'illustré à la figure 3.5, la décomposition d'un système patrimonial au niveau de la logique d'affaires se fait d'une des manières suivantes [11] :

- séparer les composants d'IU des autres composants (3.5a) ;
- séparer les composants d'IU ainsi qu'une partie de la logique d'affaires des autres composants du système (3.5b).

La séparation des IUs textuelles d'un système patrimonial - la partie (a) de la figure 3.5 - et leur refonte en utilisant des technologies de présentation Web, aboutissent à la conception des IUs Web qui n'exécutent pratiquement aucune règle d'affaires. En revanche, le résultat obtenu de la refonte des IUs d'un système patrimonial, tel qu'illustré à la partie (b) de la figure 3.5, est des IUs Web qui comprennent une partie de la logique d'affaires du système, par exemple, certaines validations de données d'entrée. Notons que la tâche de diviser la logique d'affaires en deux parties - même dans les

4. Le *slicing* représente un domaine de recherche florissant. Notre lecteur est invité à enrichir ses connaissances sur ce sujet en consultant [78].

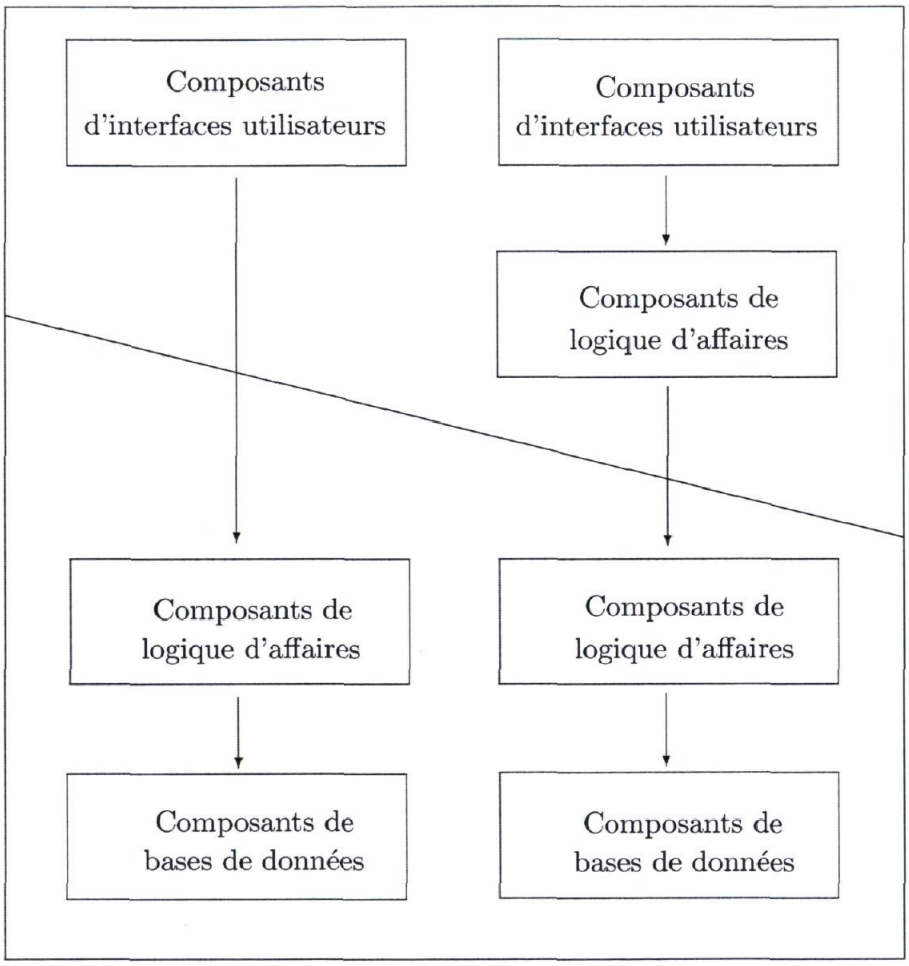

FIGURE 3.5 – Répartition des composants d'un système patrimonial sur une architecture client-serveur.

systèmes semi-décomposables et décomposables - s'avère difficile.

Comme expliqué précédemment, une partie significative du code des systèmes d'information concerne les traitements de données. Pour les données d'entrée du système, il s'agit, entre autres, de la vérification et de la validation de leur contenu, compte tenu des règles d'affaires définies dans le code [66]. Certaines règles de validations sont basées sur des données de système se trouvant dans des fichiers indexés ou des bases de données. Cette dépendance aux données de système détermine la possibilité de transmettre une règle d'affaires, du côté des IUs.

Dans la migration des IUs d'un système patrimonial à l'aide d'une technique de greffage, on doit faire le choix d'une architecture Web. Le choix de l'architecture peut varier selon le niveau de décomposabilité du système et les besoins techniques de l'organisme qui en fait la demande. Une analyse approfondie du système patrimonial permet de déterminer l'architecture Web la mieux adaptée et de diminuer les risques reliés au processus de migration.

Dans la migration d'IUs à l'aide d'une technique de greffage, la refonte des IUs textuelles mène à la conception des IUs Web qui constituent la partie client de la nouvelle architecture. La partie serveur, quant à elle, est composée des autres composants du système patrimonial : les composants de logique d'affaires et les composants de services de bases de données. La refonte d'IUs textuelles revêt une grande importance dans une migration d'IUs. Contrairement aux techniques frontales de migration d'IUs, dans les techniques de greffage, la refonte d'IUs est souvent semi-automatisée, voire complètement automatisée. Nous aborderons ce sujet plus en détail, dans le chapitre 4. Les sections suivantes, quant à elles, exposent trois approches différentes pour la migration des composants de logiques d'affaires et de services de bases de données d'un système patrimonial vers une architecture Web [16]. L'importance de ces discussions dans un document sur les IUs est que le choix d'approche pour faire migrer les composants de logique d'affaires et de services de bases de données vers une architecture Web, a un impact sur le choix de la technologie de présentation dans le processus de refonte des IUs textuelles.

3.2.1 Emballer l'intégrité de la logique d'affaires

À la page 44, nous avons décrit les caractéristiques d'un système patrimonial faisant l'objet d'une migration d'IUs frontale. Parmi ces caractéristiques, on peut noter que l'architecture conceptuelle d'un tel système est non décomposable et que l'ensemble du système, à l'exception de ces IUs, est accepté par l'organisme propriétaire du système.

Ces points constituent les raisons principales pour lesquelles, lors de la migration des IUs, on a tendance à conserver l'intégrité du système patrimonial sur sa plate-forme originelle et de se contenter d'accéder à ses IUs textuelles via le Web en utilisant une des techniques frontales de migration d'IUs. Considérons maintenant un système patrimonial semblable dont l'architecture conceptuelle est semi-décomposable. Dans ce contexte, puisque l'architecture conceptuelle du système permet de séparer les IUs, il est possible de remplacer les IUs textuelles par des IUs Web et de conserver les autres composants, sans changement.

Le résultat d'une telle migration est illustré à la figure 3.6. Comme nous pouvons le constater, le résultat de la migration est une application Web client léger dans laquelle, la partie serveur comprend les composants de logique d'affaires et de services de données qui sont emballés dans une boîte noire, appelée *enveloppeur* (de l'anglais *wrapper*). L'enveloppeur a comme fonction de faire communiquer le système patrimonial et l'extérieur. Dans ce contexte, les composants emballés conservent le langage de programmation utilisé pour leur développement ainsi que leur structure conceptuelle. Afin d'intégrer les composants emballés à l'architecture Web, il convient uniquement d'altérer leurs interfaces systèmes. On entend par interface système, la façon dont un composant est utilisé de l'extérieur. Ainsi, la partie client de l'application Web - composée d'IUs Web - est en mesure de communiquer avec les autres composants sur la partie serveur, de façon transparente. Dans ce contexte, les données du système patrimonial peuvent être transférées sur l'architecture Web en demeurant sur leur plate-forme originelle - exemple : IDS (pour *Informix Dynamic Server*). Une autre possibilité, c'est de faire migrer ces données sur une nouvelle plate-forme de données - exemple : migrer des fichiers indexés d'un système patrimonial vers une base de données relationnelle - et ensuite de les intégrer à l'architecture Web.

Dans un de ses travaux de recherche, Harry Sneed propose l'utilisation d'une telle architecture pour la migration d'IUs vers le Web. Dans son article [67], il souligne le fait que le monde patrimonial est différent du monde technologique d'aujourd'hui : les systèmes patrimoniaux possèdent pour la plupart une architecture procédurale et monolithique dans laquelle la réutilisabilité et l'intégrabilité de composants n'ont pas été prises en considération ; alors que, de nos jours, le respect de la réutilisabilité et de l'intégrabilité des composants dans le développement de tout logiciel est devenu crucial. En considérant cela, il estime qu'il est préférable que les systèmes patrimoniaux demeurent sur leur plate-forme originelle et qu'on les fasse communiquer avec d'autres systèmes par l'entremise de médiateurs. Le médiateur proposé par Sneed est celui d'une passerelle XML (de l'anglais *XML gateway*).

M. Sneed s'intéresse aux systèmes patrimoniaux semi-décomposables utilisant les ser-

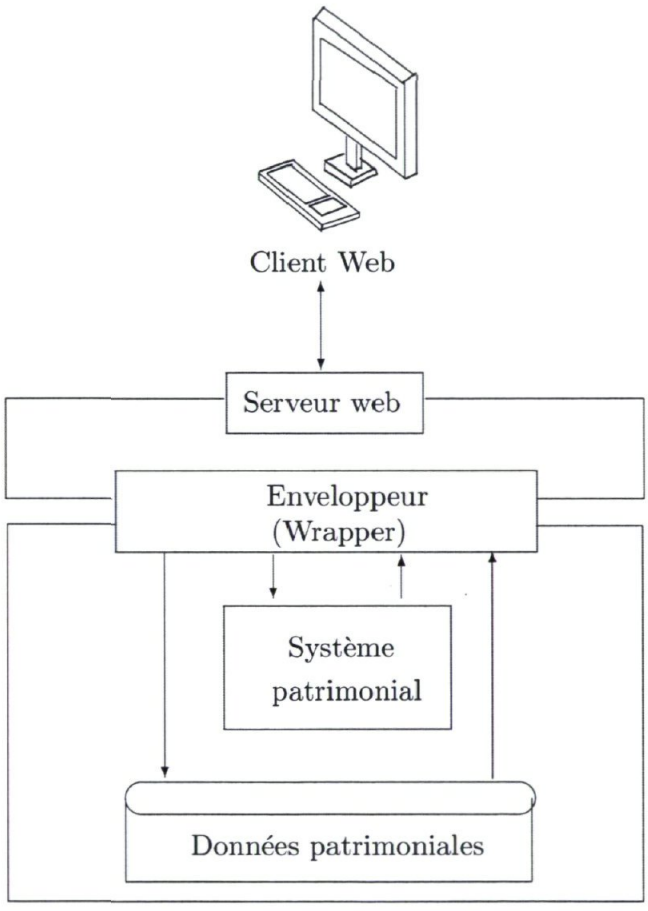

FIGURE 3.6 – Enveloppeur *(wrapper)*.

vices d'un moniteur de télétraitement. Il propose des techniques et des outils[5], afin d'emballer automatiquement les différents types de programmes de tels systèmes : les programmes en-ligne et les programmes de commande. Dans sa solution, les programmes en-ligne sont emballés de manière à pouvoir recevoir leurs entrées et produire leurs sorties, en format XML. Étant donné que les programmes patrimoniaux s'exécutent sur leur environnement originel, un médiateur doit convertir les flots de données sortant provenant des programmes patrimoniaux en des fichiers XML et *vice-versa*. Dans ce contexte, les fichiers XML représentent le contenu des IUs Web. Quant à la mise en page des IUs Web, elle se fait par des fichiers de style XML (de l'anglais *XML-style sheet*) et des fichiers JavaScript, et ce, en se basant sur les fichiers de descriptions d'écran de l'ancien système. Les détails du processus de migration sont présentés dans [67].

Emballer les composants de logique d'affaires d'un système patrimonial, tel qu'expliqué, est une solution peu coûteuse et peu risquée de migration d'IUs de systèmes patrimoniaux vers le Web. À la fin des années 90, ce choix de technique de migration a connu un succès remarquable au niveau de l'industrie. Cependant, aujourd'hui, il est de plus en plus difficile d'entretenir des systèmes patrimoniaux sur leur environnement originel [70], entre autres, à cause du manque d'expertise pour maintenir le vieux code. Il est donc devenu indispensable pour les organismes qui ont opté pour cette technique de migration, de faire appel à d'autres solutions de migration plus à long terme.

Une des solutions proposées dans la littérature est celle de traduire le langage de programmation des programmes patrimoniaux emballés. Cela permet d'une part de se débarrasser du logiciel et du matériel du vieux système en les migrant vers de nouvelles technologies et, d'autre part, de ne pas courir le risque de perturber le fonctionnement du système en modifiant le code existant.

3.2.2 Emballer des objets

La technologie orientée objet fut sans doute la tendance prédominante du développement de logiciel pendant les années 90. Cette tendance a grandement influencé le domaine de la migration de systèmes patrimoniaux. Plusieurs chercheurs se sont donc intéressés à la ré-implémentation des composants de logique d'affaires de systèmes patrimoniaux dans le paradigme orienté objet [18, 62, 64, 82, 84, 85].

Malgré tous les avantages indéniables de l'orienté objet, la conversion des compo-

5. L'auteur a développé un ensemble d'outils, appelés Cobwrap, afin d'automatiser le processus d'emballage des programmes Cobol.

sants de logique d'affaires de systèmes patrimoniaux en objets ne semble pas prometteuse [67]. Premièrement, la conversion des programmes procéduraux en leurs équivalents en orienté objet représente un défi important qui implique l'emploi de différentes techniques d'analyse et de décomposition de code. Deuxièmement, l'automatisation complète de cette opération est pratiquement impossible, car tout code patrimonial, candidat à former un objet, a besoin d'être raffiné par un expert du domaine pour s'assurer de la mise en place des principes du paradigme orienté objet, tels que l'encapsulation, l'héritage, le polymorphisme, etc. Finalement, de manière générale, les connaissances approfondies du domaine d'affaires du système - nécessaires pour effectuer les tâches manuelles de telles conversions - ne sont pas présentes chez les développeurs ou les analystes qui en sont responsables.

Dans le cadre d'un projet M&S SW (*Methods and Tools for the Production of the Software, Formation and Applications*), des chercheurs du parc scientifique et technologique (*Scientific and Technological Park*) de Salerno en Italie proposent une stratégie de migration incrémentale pour des systèmes patrimoniaux. Ils s'intéressent particulièrement à la migration des systèmes d'information patrimoniaux semi-décomposables. L'objectif de ce projet de recherche est d'établir de nouvelles solutions technologiques pour des PME (pour *Petites et Moyennes Entreprises*) possédant des systèmes d'information patrimoniaux [8].

Dans ce projet, la migration des IUs, laquelle constitue une partie de la migration complète d'un système patrimonial, repose sur la refonte des IUs textuelles dans une technologie de présentation Web. Le système patrimonial faisant l'objet du projet pilote de migration est un système semi-décomposable, développé en Cobol. Le processus de migration se divise en trois étapes :

1. utilisation de l'analyse statique de code, afin d'extraire toutes les informations nécessaires aux différentes étapes du processus de migration. Cette tâche concerne surtout l'identification des composants clés implémentant les principales fonctionnalités du système. Ces composants sont ensuite employés pour former des objets qui seront encapsulés dans des programmes Cobol. Cette transformation des programmes et des sous-programmes Cobol en objets est en effet une restructuration fondamentale de ces derniers. Notons que l'utilisation du même langage de programmation pour la transformation des composants en objets permet d'accélérer le processus de migration. Une fois la migration réussie, on peut reprendre chacun des objets, le développer dans un langage-objet et le réintégrer à l'application Web de migration ; cela constitue l'aspect incrémental de cette solution de migration ;

2. identification et extraction de tout le code implémentant les interactions entre les

utilisateurs et le système patrimonial à l'aide des techniques de *slicing* de code [78]. Le code identifié sert de base dans le développement des IUs Web ;

3. conception d'une application Java en s'inspirant du *patron de conception modèle-vue-contrôleur* ; on le fait communiquer avec les objets du système patrimonial hébergés sur leur environnement originel.

La figure 3.7 illustre l'architecture de l'application Web proposée par les chercheurs, et ses différents composants. Le « système restructuré » présente les composants, transformés en objets, du système patrimonial. L'appel à ces objets par l'application Java se fait par l'entremise de la « passerelle du système patrimonial ». La *passerelle* (de l'anglais *gateway*) est un compilateur de Cobol vers du code intermédiaire (de l'anglais *bytecode*) Java. À l'exécution, elle prend le code Cobol du système patrimonial et le compile en bytecode Java. Ainsi un programme Cobol se transforme en une classe Java qui peut être appelée et utilisée par les composants de *modèle* de l'application Java, située au centre de la figure.

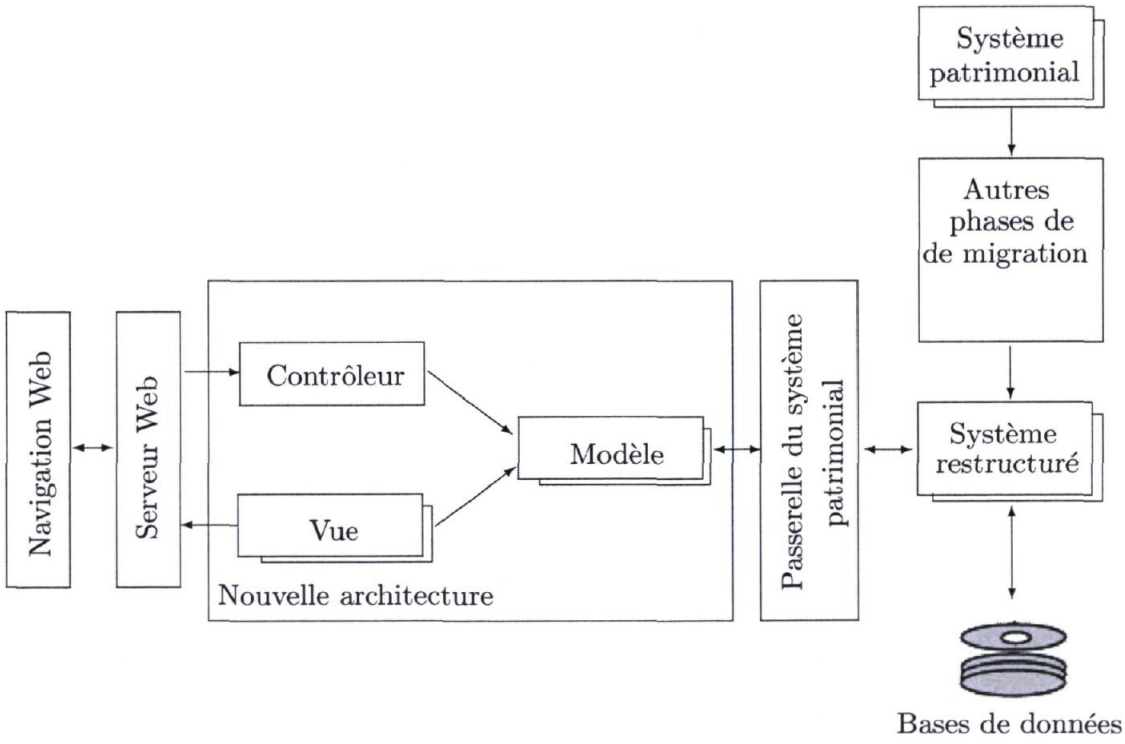

FIGURE 3.7 – Architecture Web basée MVC [8].

L'application Java comprend trois composants, modèle, vue et contrôleur :

- le *modèle* implémente la logique d'affaires de l'application ; il contient des objets enveloppants (*wrappers*) pour communiquer avec le système patrimonial via la

passerelle ainsi que les classes Java Bean des pages JSP de la *vue* ;
- le *contrôleur* est composé de servlets Java [59] responsables d'intercepter des demandes provenant des clients et de les envoyer au *modèle*. Après avoir reçu la réponse du *modèle*, il la communique à la *vue* pour qu'elle génère la page HTML demandée ;
- la *vue* est composée de pages JSP (pour *JavaServer Pages Technology*) [34]. À la demande du *contrôleur*, une page JSP se complète par la classe Java Bean [26] correspondante se trouvant sur le *modèle*. La page JSP génère une page HTML, laquelle est ensuite envoyée au client.

Les outils d'automatisation que les chercheurs ont développés, peuvent être employés pour appliquer cette stratégie dans le processus de migration d'un système patrimonial. Ces outils ainsi que l'architecture Web de migration sont présentés dans [8].

Bien que la migration vers le paradigme objet n'est pas répandue, les résultats de la phase d'ingénierie-inverse d'une telle migration peuvent être utilisés dans le processus de ré-implémentation du système patrimonial (voir la section 1.3.3). Lors du design - dans le processus de ré-implémentation d'un système patrimonial - il est intéressant d'avoir la documentation orientée objet du système patrimonial qui fait l'objet de la ré-implémentation [70].

3.2.3 Emballer des composants réutilisables

L'emballage des composants (de l'anglais *component wrapping*) est une extension de l'emballage des objets. À l'opposé de l'emballage des objets, dans l'emballage des composants, on définit un environnement dans lequel, les composants doivent interagir. Il est donc crucial que chaque composant (voir la page 33 pour la définition de composant dans ce contexte) soit emballé de sorte qu'il respecte les standards de l'environnement où il sera intégré.

Zou et al. [85] proposent une architecture Web distribuée pour la migration de systèmes patrimoniaux. Leurs travaux reposent sur la réutilisation des composants de logique d'affaires d'un système patrimonial dans une architecture distribuée Web. L'intégration de ces composants dans un environnement basé services fait en sorte que les fonctionnalités offertes par ces composants soient accessibles en tant que services. La solution de migration présentée par ces chercheurs comprend quatre étapes :

- l'identification de potentiels objets dans le système patrimonial, à l'aide de l'analyse de l'arbre syntaxique du système ;

- la migration des composants identifiés vers un langage orienté objet. La migration vers un langage orienté objet implique la création de classes et la définition des membres de chaque classe - les propriétés et les méthodes - en considérant les notions d'héritage et de polymorphisme ;
- la création des spécifications d'interface des objets obtenus, afin qu'ils puissent être utilisés dans un environnement distribué - les spécifications d'interface des objets sont conçues en utilisant les standards définis par OMG's CORBA (pour *Common Object Request Broker Architecture*) [31] [6] ;
- l'intégration de chaque composant CORBA dans une infrastructure Web basée service en utilisant le protocole SOAP.

La figure 3.8 illustre les technologies utilisées dans l'architecture de migration proposée dans [85]. Dans leur article, les chercheurs mettent l'accent sur les techniques et les méthodes employées pour identifier les objets potentiels dans le système patrimonial, pour les envelopper dans un environnement orienté objet et, finalement, pour les intégrer dans une architecture Web en utilisant le protocole SOAP. Quant à la partie client de l'application Web - les composants d'IUs -, elle ne fait toutefois pas l'objet de leurs discussions.

Notons pour terminer que, dans [85], toutes les étapes de migration ainsi que les méthodes et les technologies utilisées pendant chaque étape, sont expliquées en détail.

3.3 Conclusion

Maintenant que nous avons passé en revue les différentes approches de migration d'IUs textuelles de systèmes patrimoniaux vers le Web, il nous est possible de faire la récapitulation des avantages et des inconvénients de chaque classe d'approche.

Les techniques faisant partie de la classe d'approches frontales bénéficient des avantages suivants :

- l'implémentation des techniques frontales dans une migration d'IUs ne nécessite pas d'avoir des connaissances approfondies du domaine d'affaires du système patrimonial faisant l'objet de migration. Cela fait en sorte de diminuer des frais reliés à cette opération ;

6. CORBA définit un standard pour accéder aux données ainsi qu'aux objets dans des environnements distribués.

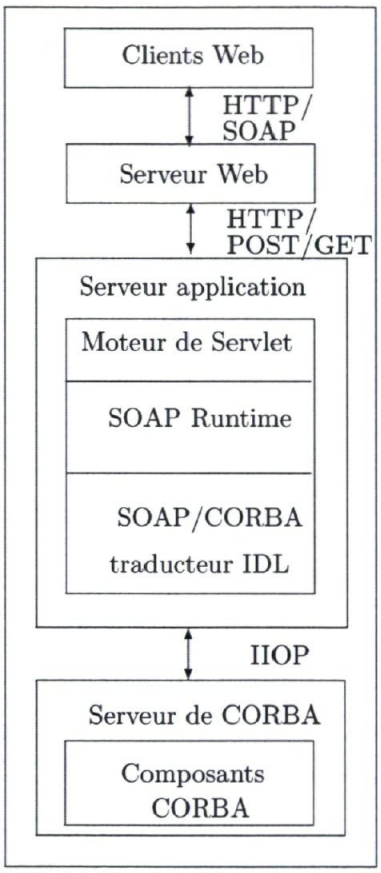

FIGURE 3.8 – Exemple d'architecture distribuée Web pour la migration de systèmes patrimoniaux [85].

- il n'y a normalement aucune modification au code existant du système patrimonial dans le processus de migration d'IUs avec des techniques frontales. Les risques de perturber le bon fonctionnement du système patrimonial ne sont donc pas importants.

Malgré leur simplicité, les techniques frontales sont souvent critiquées pour les raisons suivantes :

- l'utilisation d'une technique frontale en migration d'IUs mène à la conception d'applications Web inflexibles et non réutilisables - dans ce type de migration, l'intégrité du système patrimonial est emballée par une couche logiciel, dite *enveloppeur* (de l'anglais *wrapper*). De cette manière, le système patrimonial peut continuer à s'exécuter sur son environnement originel, et à communiquer avec des autres composants de l'application Web qui l'entoure, par l'entremise de l'*enveloppeur*. Ainsi, le système patrimonial constitue un des principaux composants de la nouvelle application générée. Malgré la migration du système vers le Web et les coûts reliés à cette opération, aucun composant de l'application Web n'est réutilisable [7] ;

- les techniques frontales n'apportent généralement pas d'amélioration fonctionnelle aux IUs textuelles. Elles représentent donc des solutions de migration à court terme avec lesquelles on tente uniquement de préparer un accès externe aux IUs à un système patrimonial ;

- l'emballage des IUs textuelles par des IUs Web fait en sorte de dupliquer le code des IUs. Cela a comme conséquence de nécessiter la duplication des activités de maintenance sur les IUs. Autrement dit, toute modification d'IUs doit nécessairement se faire à la fois sur les anciennes et sur les nouvelles IUs.

Quant aux techniques de greffage, elles sont généralement appréciées pour les raisons suivantes :

- séparer les IUs du reste du système et leur refonte dans une technologie de présentation permet d'obtenir un système modulaire dans lequel les IUs peuvent être maintenues plus facilement ;

- dans la migration d'IUs textuelles vers le Web, les IUs textuelles sont remplacées par des IUs Web. Le remplacement des IUs textuelles permet de se débarrasser de la vieille technologie des IUs ;

7. Aujourd'hui, la réutilisabilité représente l'un des principaux soucis dans le développement de logiciel.

- dans la migration d'IUs à l'aide des techniques de greffage, il est possible d'améliorer l'aspect et la convivialité des IUs textuelles pour mieux répondre aux exigences du marché ;

- une grande partie du processus d'une telle migration peut être automatisée.

Toutefois, ces techniques souffrent généralement des lacunes suivantes :

- des coûts relativement élevés - le processus de migration d'IUs à l'aide d'une technique de greffage est composé de plusieurs étapes. Chacune d'entre elles nécessite l'utilisation de différentes techniques. De plus, pour la réalisation des interventions manuelles d'une telle migration, il faut avoir recours à des experts du domaine d'affaires ;

- l'altération du code existant - modifier les règles d'affaires d'un système patrimonial effectuant des tâches critiques entraîne des risques élevés.

Comme nous l'avons déjà mentionné, avant que la refonte d'IUs soit automatisée au niveau universitaire, la refonte d'IUs dans l'industrie se faisait manuellement. Malgré sa complexité, l'automatisation de refonte d'IUs rend le processus moins long et plus robuste. Le prochain chapitre est concentré sur la refonte automatisée d'IUs et de deux étapes que comprend cette refonte.

Chapitre 4

Refonte automatisée d'IUs textuelles

> Pour atteindre la vérité, il faut une fois dans la vie se défaire de toutes les opinions qu'on a reçues, et reconstruire de nouveau tout le système de ses connaissances.
>
> René Descartes

Comme nous l'avons mentionné à la section 1.3.2.1, la migration d'IUs textuelles de systèmes patrimoniaux vers le Web impose souvent la refonte d'IUs textuelles dans une technologie de présentation Web [1] (voir la page 24). Dans le chapitre 3, nous avons également vu qu'étant donné la taille gigantesque d'une grande majorité des systèmes d'information patrimoniaux, la réalisation manuelle de cette refonte était pratiquement impossible. C'est la raison pour laquelle, dans la refonte d'IUs textuelles, on favorise l'utilisation des techniques et des outils qui aident à automatiser les traitements. En plus de réduire le temps et les ressources nécessaires, l'automatisation permet d'éviter l'introduction d'erreurs, et ce, parce que les résultats d'une opération automatisée ne reposent pas sur le jugement humain [50].

Le processus de refonte automatisée d'IUs textuelles se décline en deux étapes : l'ingénierie-inverse et la conception descendante. Il s'agit d'étapes successives, c'est-à-dire que les résultats obtenus à la première étape sont utilisés pour constituer les entrées

1. Bien qu'elles aient été étudiées dans ce document pour fournir une vision globale des techniques existantes sur le marché, les techniques frontales qui se limitent à donner un accès Web aux IUs du système patrimonial, objet de migration, sans nécessiter la refonte des IUs, ne figurent pas parmi les techniques de migration d'IUs.

de la seconde étape. Dans l'étape d'ingénierie-inverse, on modélise les IUs textuelles faisant l'objet de migration pour ensuite, dans l'étape de conception descendante, se baser sur les modèles abstraits afin de générer des IUs Web.

Dans ce chapitre, nous présentons les 2 étapes de la refonte d'IUs dans le processus de migration d'IUs textuelles de systèmes patrimoniaux vers le Web. Nous énumérons, pour chaque étape, quelques exemples en nous basant sur les travaux étudiés dans le cadre de ce mémoire.

4.1 Ingénierie-inverse

Dans le processus de refonte des IUs textuelles d'un système patrimonial, l'étape d'*ingénierie-inverse* a pour objectif de modéliser les IUs. Pour ce faire, on procède d'abord à une étape d'*inférence de spécifications d'IU*, à savoir l'identification de la structure, les fonctionnalités et les comportements de chaque IU. Ensuite, on génère les modèles conceptuels des informations obtenues [13]. Les modèles conceptuels sont indépendants du langage source des IUs et représentent deux types de spécifications des IUs textuelles qui font l'objet de la refonte : les spécifications statiques et les spécifications dynamiques. Dans l'étape de conception descendante, les spécifications statiques servent au design de la mise en page (de l'anglais *layout*) des nouvelles IUs. Quant aux spécifications dynamiques, elles permettent de reproduire les anciennes relations entre les IUs sur l'architecture Web de migration. Nous présenterons trois exemples concrets de modélisation d'IUs au chapitre 5.

Les *spécifications statiques* d'une IU correspondent à la présentation des éléments utilisés dans la composition de l'IU ainsi que les relations qui existent entre eux. Dit de façon simplifiée, les spécifications statiques d'une IU représentent la structure visuelle de cette dernière. La position et la dimension des différents champs et la séquence de tabulation dans une IU, c'est-à-dire l'ordre dans lequel les champs deviennent actifs, sont parmi les informations que l'on trouve dans les spécifications statiques d'une IU. Quant aux *spécifications dynamiques*, elles exposent les comportements d'une IU sous forme d'une séquence des états possibles que l'IU peut assumer de même que les transitions possibles entre les différents états. En d'autres termes, les spécifications dynamiques représentent la carte de navigation (de l'anglais *road-map*) des IUs du système à l'étude. À l'affichage, une IU est dans son *état* initial où un ensemble d'*actions* est disponible à l'utilisateur. L'IU garantit certaines réponses (comportements), si l'utilisateur effectue une ou une série de ces actions. À la suite de chaque action, l'IU change d'état : l'ouverture ou la fermeture d'une fenêtre ou d'une boîte de dialogue sont des exemples

d'actions qui peuvent changer l'état d'une IU. Dans son nouvel état, les actions offertes par l'IU ne sont pas nécessairement les mêmes que dans son ancien état. Il est toutefois possible qu'elle offre les mêmes actions, mais avec un sens différent. À titre d'exemple, l'action « aide » dans une IU produit des résultats différents dans les différents états d'un système. Dans la suite de ce chapitre, nous présentons deux méthodes d'inférence des spécifications des IUs dans un système patrimonial.

Tel qu'expliqué, l'objectif de l'étape d'ingénierie-inverse consiste à générer des modèles abstraits des spécifications des IUs d'un système. Dans la littérature, on propose l'utilisation de certaines méthodes formelles pour présenter les spécifications dynamiques [37, 60, 61]. Des grammaires formelles, des machines à états finis (de l'anglais *finite state machine*), des réseaux de Petri (de l'anglais *Petri nets*) et des systèmes de transition concurrents (de l'anglais *concurrent transition system*) sont des exemples de telles méthodes formelles employées pour représenter des spécifications dynamiques [28, 41]. Notons que l'utilisation des méthodes formelles permet de garantir que les modèles conçus reflètent de façon précise et exacte les spécifications des IUs textuelles.

Les modèles abstraits conçus à l'étape d'ingénierie-inverse d'une refonte d'IUs sont utilisés pour générer des IUs graphiques dans l'étape de conception descendante. Notons que les nouvelles IUs générées à partir de modèles précis, qui reflètent les vraies fonctionnalités des IUs d'un système, se comportent sensiblement de la même manière que les IUs textuelles originelles. Puisque la reproduction des anciennes fonctionnalités constitue un des objectifs importants pour une migration d'IUs (voir la page 24), il existe un lien direct entre le niveau de précision des modèles d'IUs textuelles et la réussite d'une telle migration - d'où vient l'importance accordée à l'étape d'ingénierie-inverse dans tout processus de migration d'IUs. De plus, l'automatisation de l'étape de conception descendante des modèles précis est relativement moins compliquée que l'automatisation de la conception descendante des modèles vagues et non précis.

Puisque les technologies de présentation Web imposent généralement que les spécifications d'IUs, sur lesquelles la génération des nouvelles IUs repose, soient basées objets, des modèles conceptuels d'IUs textuelles nécessitent pour la plupart une restructuration, avant qu'ils ne soient utilisés dans le processus de conception descendante.

Par ailleurs, un système d'information patrimonial est souvent accompagné d'une pile de documentations qui expliquent le design et les fonctionnalités des différents composants. Bien qu'au moment de la livraison d'un tel système, ces documentations expriment précisément le fonctionnement du système lors de la migration, elles ne sont plus une source fiable d'information, car dans la majorité des cas elles ne sont pas

gardées à jour. Considérant cela, pour obtenir de l'information à jour concernant les fonctionnalités et les comportements des IUs, il devient nécessaire d'utiliser d'autres sources d'information que les documentations du système.

Pour obtenir les informations sur les spécifications des IUs d'un système - inférence de spécifications -, deux méthodes sont proposées dans la littérature :

- ingénierie-inverse d'interactions (de l'anglais *interaction reverse-engineering*) ;
- ingénierie-inverse de code (de l'anglais *code reverse-engineering*).

Dans la suite de cette section, nous présentons ces deux types d'ingénierie-inverse et nous les illustrons à travers quelques exemples d'utilisation de ces techniques dans la littérature.

4.1.1 Ingénierie-inverse d'interactions

Comme nous l'avons mentionné à la section 3.1, dans la migration d'IUs à l'aide d'une technique frontale, l'objectif est de conserver l'intégrité du système patrimonial qui fait l'objet d'une telle migration (voir la page 44). Dans ce contexte, puisque la décomposition du système n'est pas envisagée, la modélisation des IUs textuelles ne nécessite pas une compréhension approfondie du code source du système. En d'autres termes, il est possible d'analyser les entrées et les sorties du système patrimonial, pendant son utilisation par des utilisateurs, afin de comprendre les comportements et la structure de ses interfaces [9, 23]. Dans la littérature, une telle technique est nommée *technique d'analyse dynamique de code* ou encore *technique de boîte noire* [16].

Rappelons qu'un système patrimonial est souvent l'objet de nombreuses activités de maintenance, faites par des développeurs autres que ceux qui ont participé au développement initial du système. Puisque la documentation des systèmes patrimoniaux n'est généralement pas à jour et qu'elle ne reflète pas les vraies fonctionnalités, ces développeurs s'appuient seulement sur leur compréhension du code source pour effectuer des changements. Compte tenu de la grande taille et de la complexité de compréhension du code de la majorité des systèmes patrimoniaux, les développeurs tentent habituellement de ne pas courir le risque de perturber le bon fonctionnement du système, en modifiant ou en abolissant les parties de code qui peuvent être éventuellement utilisées par le code existant. Ils préfèrent donc être prudents et se contentent d'ajouter de nouvelles fonctionnalités en définissant des méthodes séparées, lesquelles s'intègrent au code existant. Par conséquent, le code source des systèmes patrimoniaux souffre souvent d'une présence importante de code inutilisé, qui n'a pas été enlevé - code mort - ou encore de

code déjà existant, qui n'a pas été identifié - code dupliqué - lors des modifications.

Les techniques d'*analyse statique* de code sont basées sur l'étude du code source d'un système ; l'analyse statique d'un code, qui comprend des parties inutilisées et redondantes, mène à une compréhension inexacte des fonctionnalités. En revanche, puisque, les techniques d'analyse dynamique de code reposent sur les informations que l'on obtient pendant l'exécution d'un système, les résultats de telles analyses reflètent les vraies fonctionnalités. C'est la raison pour laquelle, les défenseurs des techniques de boîte noire estiment que, lorsque l'on n'est pas dans l'obligation d'altérer le code source d'un système patrimonial, ces techniques sont privilégiées aux techniques qui nécessitent la compréhension du code source.

Cependant, il devient parfois nécessaire de décomposer et de faire migrer un système patrimonial de son environnement originel vers de nouvelles technologies. De plus, il n'est pas toujours possible de récupérer toutes les informations nécessaires à la modélisation d'IUs à l'aide des techniques d'analyse dynamique de code. Dans ces contextes, l'utilisation des techniques d'analyse statique de code est indispensable.

Ci-dessous, nous énumérons deux exemples d'utilisation de techniques de boîte noire dans le processus de refonte d'IUs textuelles :

- Bovenzi et al. [9] utilisent des techniques de boîte noire pour obtenir de l'information sur les aspects statiques et dynamiques d'IUs de systèmes patrimoniaux. Ils proposent l'usage des diagrammes UML (pour *Unified Modeling Language*) : plus précisément, ils proposent les diagrammes de cas d'utilisation (de l'anglais *use cases diagrams*), les diagrammes d'activités et les diagrammes de séquences, comme techniques pour la modélisation des IUs. Ces diagrammes UML serviront ensuite à concevoir des diagrammes états-transitions (pour *State Transition Diagram* ou STD) modélisant les comportements des IUs. Dans un diagramme états-transitions, les noeuds représentent les différents états des IUs, alors que les arcs représentent les événements causant des changements d'état. Notons que, dans ce travail, la conception des diagrammes UML ainsi que des diagrammes STD se font manuellement.

- Stroulia et al. [74] utilisent les traces des interactions entre l'utilisateur et le système pour générer les modèles conceptuels du fonctionnement des interfaces. Les travaux de ces chercheurs, qui représentent un autre exemple d'utilisation des techniques de boîte noire, sont présentés à la section 5.3.

4.1.2 Ingénierie-inverse de code

Tel que mentionné à la section 3.2, dans la migration des IUs d'un système patrimonial vers le Web à l'aide d'une des techniques de greffage, la première étape consiste à séparer le code des IUs textuelles de celui des autres composants du système (voir la page 51). Pour ce faire, il convient d'analyser le code source du système patrimonial afin d'y identifier et d'en extraire le code des composants d'IU : décomposition.

Une fois que la séparation des IUs est faite, la modélisation des IUs repose sur la compréhension du code des composants d'IU textuelles séparées du système. Le phénomène de la compréhension de code, à l'aide des techniques d'analyse statique que l'on connait sous le nom de *compréhension de programmes* (de l'anglais *program understanding*), a fait l'objet de plusieurs travaux de recherche et a permis d'énoncer des solutions et de proposer des outils automatisés d'analyse de code [2].

La représentation du code, à ce stade, est habituellement dépendante du langage du code analysé. Dans l'étape de conception descendante, si la génération des nouvelles IUs se base sur des représentations dépendantes du langage du code source, chaque langage de programmation nécessite son propre outil pour effectuer cette génération. Alors qu'en pratique, on tente de développer des outils de génération d'IUs les plus génériques possible. Ainsi, on prétend que de tels outils sont en mesure de générer des IUs graphiques, peu importe le langage utilisé pour le développement des IUs originelles. Considérant cela, la prochaine étape d'ingénierie-inverse consiste à transformer les représentations dépendantes des IUs textuelles en des modèles abstraits. Tel qu'expliqué à la page 69, les modèles abstraits peuvent être présentés à l'aide d'un formalisme de représentation de comportement.

Ci-dessous, nous présentons quelques exemples d'ingénierie-inverse de code, proposés dans la littérature :

- Merlo et al. [41] proposent l'utilisation de l'analyse de flot de données, l'analyse de flot de contrôle, les techniques de *slicing*, la propagation des constantes (pour *Multi-Valued Constant Propagation* ou MVCP) [42] et l'analyse de dépendance des programmes (de l'anglais *program dependency analysis*) [27], afin d'effectuer l'inférence de spécifications d'IUs (de l'anglais *IU specifications inference*) [28]. Les chercheurs ont développé des outils pour automatiser le processus d'inférence de

2. Dans certains langages de programmation, l'analyse complète de code exige l'utilisation d'une combinaison de l'analyse statique et dynamique. Par exemple, dans un programme écrit en C, c'est uniquement à l'exécution que l'on connait les chemins pris par la valeur d'un pointeur. Dans un tel contexte, l'utilisation conjointe des techniques d'analyse statique et dynamique est indispensable.

spécifications. Ils établissent un langage de représentation intermédiaire, nommé AUIDL (pour *Abstract User Interface Description Language*), pour décrire la structure et les comportements d'IUs. Dans AUIDL, la structure d'une IU est représentée à l'aide d'une approche orientée objet. Quant aux comportements, ils sont représentés par l'algèbre de processus CCS [45]. Les chercheurs utilisent un outil pour automatiser la génération des IUs graphiques à partir des spécifications AUIDL [25]. Le langage AUIDL et les détails des travaux de ces chercheurs sont présentés à la section 5.1.

- Moore et al. [47] font aussi appel aux techniques d'analyse statique de code afin d'inférer les spécifications des IUs. Ils se servent ensuite d'un langage de représentation de connaissances, appelé CLASSIC [55], pour présenter les spécifications des IUs de façon abstraite. Nous étudions ces travaux dans la section 5.2.

4.2 Conception descendante

L'étape de *conception descendante* (de l'anglais *forward-engineering*), dans le processus de refonte des IUs textuelles d'un système patrimonial, consiste à générer des IUs dans une technologie de présentation Web. Les nouvelles IUs sont conçues à partir des modèles abstraits obtenus de l'étape d'ingénierie-inverse. Dans la majorité des projets pilotes de migration Web, présentés dans la littérature, l'étape de conception descendante est négligée : les chercheurs se concentrent plutôt sur l'étape d'ingénierie-inverse, étant l'étape la plus technique du processus de refonte. Ils supposent que les modèles précis et complets aboutissent facilement à la conception des IUs Web qui correspondent bien aux fonctionnalités des IUs textuelles originelles.

Comme mentionné précédemment, étant donné la popularité et la simplicité d'utilisation du Web, dans ce mémoire, nous nous concentrons sur la migration des IUs patrimoniales vers des technologies de présentation Web. L'étape de conception descendante dans le contexte du Web a pour objectif de concevoir des IUs graphiques pouvant être comprises et exécutées par des navigateurs Web. Le choix de technologie Web pour la génération des IUs d'un système patrimonial est fortement influencé par les besoins de l'organisme propriétaire du système, car les organismes ont tendance de vouloir faire migrer leurs systèmes patrimoniaux vers les mêmes technologies qu'ils utilisent déjà dans le contexte de leurs systèmes existants.

Avec la prolifération des appareils mobiles tels que les assistants numériques personnels (de l'anglais *Personal Digital Assistant* ou *PDA*) et les téléphones cellulaires qui

offrent un accès Internet, il est fort intéressant de pouvoir fournir un accès aux services des systèmes patrimoniaux pour les utilisateurs de ce type d'IUs. Comparativement aux IUs Web, les appareils mobiles possèdent généralement de plus petits écrans et une connectivité limitée de réseau. Néanmoins, certaines technologies de présentation Web, comme XML (pour *Extensible Markup Language*), sont multiplate-formes. Une *technologie multiplate-forme* est une technologie dont l'utilisation est possible sur différentes plate-formes. La génération des IUs dans une telle technologie de présentation est donc privilégiée. Ainsi, les IUs générées peuvent être facilement réutilisables sur d'autres clients que les clients Web (navigateurs Web).

La conception descendante représente l'étape de migration qui peut être facilement automatisée. L'automatisation de la génération des IUs, pour l'environnement cible, permet d'avoir une rétroaction rapide des modèles abstraits de l'étape d'ingénierie-inverse. Ainsi, si les nouvelles IUs générées ne sont pas satisfaisantes et qu'elles ne reflètent pas toutes les fonctionnalités de l'ancien système, il est possible de raffiner les modèles et de régénérer les IUs autant que nécessaire.

Voici une énumération des technologies de présentation utilisées par des chercheurs dans le cadre de leurs travaux sur la migration d'IUs patrimoniales vers le Web :

- XML (pour *Extensible Markup Language*) [80] et des fichiers XSL ont été utilisés par Bovenzi et al. [9] dans la conception des IUs multiplate-formes ;

- dans le cadre d'un projet pilote, des chercheurs de l'Université de Sannio en Italie [4] ont utilisé des pages ASP (pour *Microsoft Active Server Pages*) et le langage de scripte VBScript pour le développement des IUs Web du système migré ;

- Bodhuin et al., dans [8], proposent l'utilisation de pages HTML générées dynamiquement à partir de fichiers JSP (pour *JavaServer Pages Technology*) [34]. Les fichiers JSP, qui sont compilés par un serveur Web, constituent uniquement la présentation des IUs et n'implémentent aucune règle d'affaires. Dans leur approche, chaque page JSP correspond à une classe Java Bean [26] qui complète sa page JSP correspondante.

4.3 Conclusion

Ce chapitre a donc présenté deux étapes de la refonte des IUs textuelles dans la migration d'IUs vers le Web. Parmi les étapes exposées, l'étape d'ingénierie-inverse

représente une étape technique et, dans certains cas, complexe.

Dans l'étape d'ingénierie-inverse, lorsque l'on opte pour des techniques de compréhension de programmes, il importe que l'architecture du système à l'étude soit modulaire. Malgré l'utilisation des techniques élaborées de compréhension de programmes, la compréhension et la décomposition d'une architecture non décomposable s'avèrent difficiles et risquées. D'une part, l'identification du code des composants d'IUs dans une structure monolithique est difficile et d'autre part, la séparation du code de ces composants risque de perturber le bon fonctionnement du système, lequel est souvent critique et important. Les travaux de recherche étudiés dans le cadre de ce mémoire viennent à l'appui de cette dernière hypothèse, car dans ces travaux, l'ingénierie-inverse à l'aide des techniques de compréhension de programmes est souvent favorisée dans la migration des IUs des systèmes patrimoniaux qui possèdent une architecture semi-décomposable ou décomposable. Pour la modélisation des IUs des systèmes non décomposables, on propose généralement l'emploi des techniques de boîte noire qui ne nécessitent aucune modification du code du système qui fait l'objet de migration.

Nous avons vu que la génération des IUs Web à partir des modèles abstraits se fait à l'étape de conception descendante. L'automatisation de cette étape permet d'avoir une rétroaction rapide sur l'exactitude des modèles obtenus à l'étape d'ingénierie-inverse. Le choix de la technologie de présentation pour le développement des nouvelles IUs dépend de l'architecture du système patrimonial ainsi que des besoins de l'organisme qui en fait la demande. Cependant, avec l'émergence des appareils mobiles, il est devenu de plus en plus intéressant de pouvoir proposer des solutions de migration d'IUs qui offrent la possibilité de générer des IUs multiplate-formes.

Dans le chapitre suivant, nous étudions trois plate-formes de migration d'IUs. Dans notre présentation de ces trois plate-formes, nous nous attardons notamment sur les étapes de refonte d'IUs et nous énumérons les techniques et les méthodes qui y sont utilisées. Le contexte d'essai pour chaque plate-forme est également présenté.

Chapitre 5

Plates-formes de migration d'IUs

Sans exemple, on ne peut rien enseigner correctement.

Columelle

Dans ce chapitre, nous présentons trois plate-formes [1] de migration d'IUs dans lesquelles l'objectif visé est d'automatiser le processus de refonte d'IUs. Parmi les trois plate-formes présentées, les deux premières font appel à des techniques d'analyse statique de code, alors que la dernière utilise des techniques de boîte noire afin d'analyser et de modéliser les IUs textuelles. Les techniques proposées dans ces trois plate-formes sont à la base de plusieurs travaux de recherche dans le domaine de la migration d'IUs.

5.1 AUIDL

Un des premiers travaux de recherche sur la migration d'IUs textuelles est réalisé par Melro *et al.* dans le cadre d'un projet Macroscope géré par la firme DMR Inc [44, 28, 42, 43]. L'objectif de ce travail, mené par le centre de recherche informatique de Montréal (*Computer Research Institute of Montreal*), est d'établir une méthodologie pour automatiser la refonte d'IUs textuelles dans des technologies de présentation graphiques. L'approche de migration d'IUs employée dans ce projet se classifie dans la classe d'approches de greffage (voir la section 3.2).

1. Par plate-forme, nous entendons un ensemble de méthodes et de techniques réunies généralement dans un environnement de développement pour permettre la réalisation complète d'un processus donné - dans notre cas, il s'agit de migration d'IUs.

La figure 5.1 illustre le modèle de refonte d'IUs proposé par les chercheurs. Ce modèle se compose des étapes suivantes :

- analyse du code source du système et production de l'arbre syntaxique abstrait (de l'anglais *Abstract Syntax Tree* ou AST) [2] du code analysé ;
- extraction des fragments d'IUs de l'arbre syntaxique du système et conception de l'arbre syntaxique des IUs ;
- modélisation des spécifications des IUs textuelles avec le langage AUIDL ;
- transformation des spécifications textuelles en spécifications graphiques en utilisant le même langage ;
- génération des IUs graphiques à partir des spécifications graphiques des IUs.

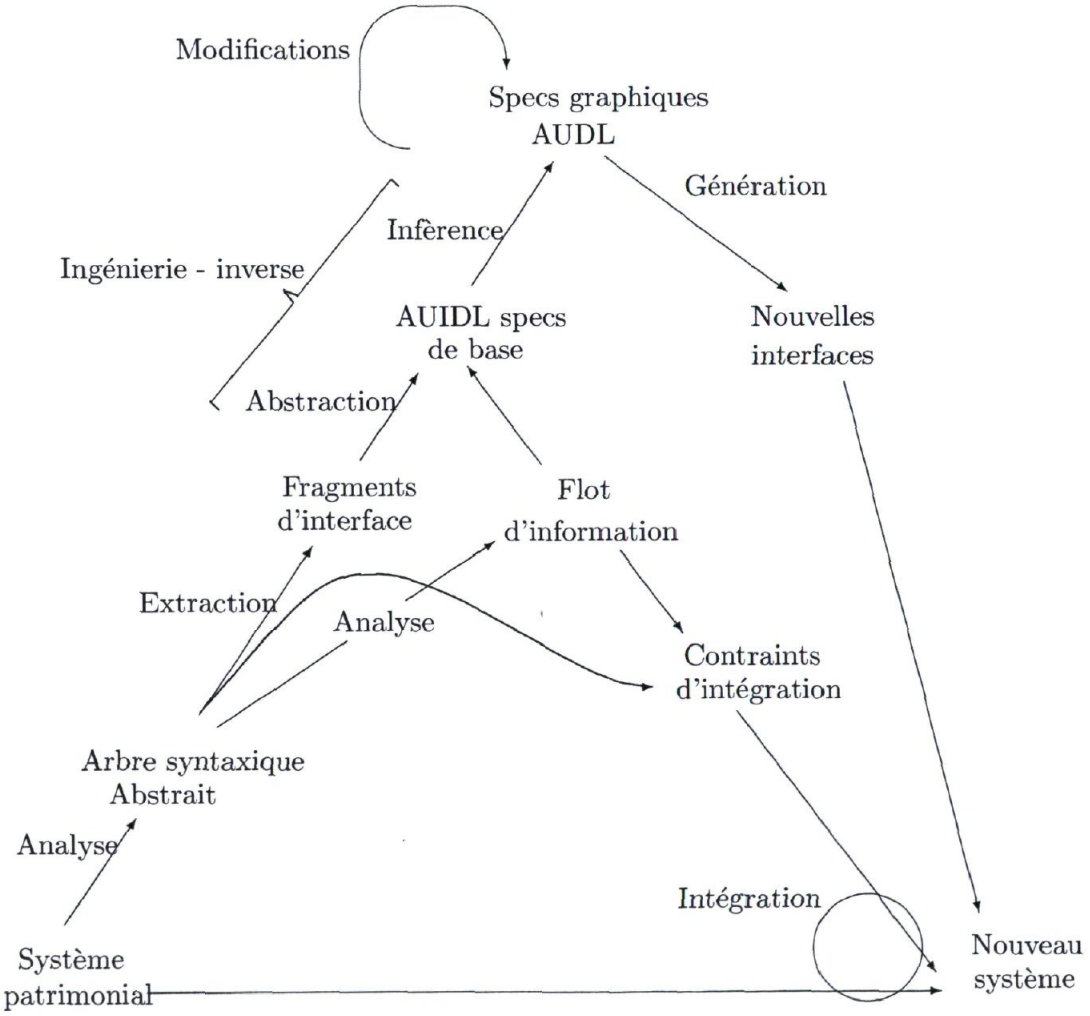

FIGURE 5.1 – Modèle de ré-ingénierie [44].

Les techniques et les méthodes utilisées dans l'étape d'ingénierie-inverse de cette solution ont servi de base dans plusieurs travaux de recherche sur la migration d'IUs. La suite de cette section décrit sommairement le langage AUIDL. Nous y présentons

également le contexte d'essai et les différentes techniques employées par les chercheurs pour automatiser la refonte d'IUs.

5.1.1 Langage AUIDL

L'approche AUIDL doit son succès à la capacité du langage AUIDL (pour *Abstract User Interface Description Language*) à offrir un langage de description abstrait pour représenter à la fois les spécifications statiques et dynamiques d'IUs. Ce langage joue en même temps le rôle du langage cible dans la phase d'ingénierie-inverse, le langage de travail pour le design d'IUs graphiques, mais aussi le rôle de langage de spécifications pendant la phase de conception descendante de GUIs [41].

5.1.1.1 Expression de spécifications statiques

Le langage AUIDL permet de présenter la structure visuelle d'une IU en utilisant des notions orientées objets. Dans l'AUIDL, des *classes* représentent les différents éléments qui peuvent être utilisés dans la conception d'une IU. Pour chaque classe, un ensemble d'*attributs* décrit ses propriétés. À titre d'exemple, la syntaxe d'une classe est illustrée à la figure 5.2. La classe « Window » représente donc le cadre dans lequel les éléments d'une IU se réunissent. Comme nous pouvons le constater, la taille, la position, le fond d'écran et la police de fonte sont des exemples d'attributs qu'une classe « Window » peut posséder.

```
class WINDOW
attributes
        SIZE_X : INTEGER is 100;
        SIZE_Y : INTEGER is 100;
        POS_X  : INTEGER is 0;
        FONT   : SYMBOL is Courier;
        COLOR  : INTEGER is 0;
end class
```

FIGURE 5.2 – Exemple de définition d'une classe en AUIDL.

Alors qu'une classe définit une notion générale, une *instance* - *objet* - est une définition concrète d'un élément d'une IU. Par défaut, une instance détient tous les attributs de la classe dont elle hérite.

La figure 5.3 illustre une instance de la classe WINDOW, appelée W1. Notons que les valeurs définies pour les deux attributs « SIZE_X » et « SIZE_Y » remplacent les valeurs préalablement définies pour les mêmes attributs dans la définition de la classe WINDOW. Quant aux autres attributs de l'objet W1, ils conservent les valeurs définies pour les attributs de la classe mère : WINDOW.

```
instance W1 : WINDOW
attributes
        SIZE_X : INTEGER is 50;
        SIZE_Y : INTEGER is 100;
end;
```

FIGURE 5.3 – Exemple de définition d'une instance en AUIDL.

Les éléments visuels ou les composants d'une IU ont des relations compositionnelles entre eux. Par exemple, un élément peut contenir un autre élément ou encore être contenu dans un autre élément. Dans l'AUIDL, ces relations compositionnelles entre les objets - éléments - sont exprimées par deux mots-clés « *contains* » et « *contained by* ». La figure 5.5 illustre l'utilisation de ces mots-clés dans la syntaxe de l'AUIDL.

```
instance window W1 : WINDOWS
contains field B;
...
end;

instance field B : FIELD
contained by window W1;
...
end;
```

FIGURE 5.4 – Exemple de l'utilisation des mots-clés *contains* et *contained by*.

La dernière notion qu'il reste à définir pour compléter la présentation de l'expression des spécifications statiques d'IUs à l'aide du langage AUIDL, est la notion de partage des valeurs des attributs entre les objets. Les mots-clés « *import* » et « *export* » permettent aux deux objets de partager une valeur entre eux. Par exemple, tel que décrit à la figure 5.5, l'objet A exporte la valeur de son attribut « POS_X » pour la partager avec l'objet B, lequel importe cette valeur sans la nécessité de la définir.

```
instance field A : FIELD
attribute
        POS_X : INTEGER is 100;
export
        POS_X to B;
end;

instance field B : FIELD
import
        POS_X from A
end;
```

FIGURE 5.5 – Exemple de l'utilisation des mots-clés *import* et *export*.

5.1.1.2 Expression de spécifications dynamiques

Pour exprimer les spécifications dynamiques d'une IU, le langage AUIDL utilise l'algèbre de processus CCS (pour *Calculus of Communicating Systems*) de Robin Milner [45]. Comparativement aux autres formalismes de représentation de comportements (voir la page 69), CCS possède l'avantage de pouvoir décrire les comportements d'un objet en équations mathématiques. De plus, cette représentation est intuitive, c'est-à-dire qu'un être humain saisirait et observerait les comportements d'une IU de la même façon qu'ils sont présentés avec l'algèbre de processus CCS.

Dans la définition d'un objet, les spécifications dynamiques se trouvent sous le mot-clé « *behaviour* ». Une liste d'équations exprime ces spécifications. La partie gauche de chaque équation représente un objet dans un certain état. La partie à droite, par contre, expose des interactions possibles à partir de cet état et les résultats obtenus, si l'interaction se réalise. La figure 5.6 illustre un exemple des comportements d'une IU simple. Selon les spécifications, l'objet W1 est une instance de la classe « WINDOW » qui contient deux objets A et B. Dans l'état initial de W1, l'objet A peut accepter un paramètre x et changer l'état de W1 en W2. Dans le nouvel état, l'objet B affiche le même paramètre x et W1 revient à son état initial.

Les détails techniques, la syntaxe et un exemple d'usage du langage AUIDL sont présentés dans [41].

```
instance W1 : WINDOW
contains A;B;
parameter x

behaviour
        W1 = A:GetString(x).W2
        W2 = B:PutString(x).W1
end;
...
```

FIGURE 5.6 – Exemple de présentation des spécifications dynamiques d'une IU.

5.1.2 Contexte d'essai

Les expérimentations de la présente solution sont effectuées sur les IUs textuelles d'un système d'information patrimonial écrit en Cobol/CICS dont l'architecture semi-décomposable facilite grandement l'étape d'ingénierie-inverse. Pour l'étape de conception descendante, on fait usage d'un environnement commercial de développement d'IUs graphiques, à savoir Easel, lequel a comme fonction d'automatiser le processus de la génération des nouvelles IUs. Pour les besoins du projet, il est convenu d'accommoder des utilisateurs actuels du système patrimonial et de continuer à leur offrir les services du système via des terminaux. C'est la raison pour laquelle dans le contexte d'essai, les nouvelles IUs couvrent, mais ne remplacent pas les IUs textuelles.

5.1.3 Phase d'ingénierie-inverse

Dans l'étape d'ingénierie-inverse, les chercheurs optent pour l'ingénierie-inverse de code. Afin d'identifier le code des IUs textuelles, ils procèdent d'abord à une analyse statique du code source du système en utilisant un analyseur syntaxique. Cela permet de construire l'arbre syntaxique abstrait (AST) du système. À l'aide des techniques de *slicing* [78], ils en extraient ensuite des fragments d'IUs pour obtenir un sous-arbre d'IUs.

Une analyse de flot d'information doit être également effectuée sur le code source du système pour identifier les comportements des IUs ainsi que pour pouvoir déterminer les points de coupure desquels on déduit les endroits - dans le code originel - où les GUIs générées doivent être intégrées.

La prochaine étape consiste à convertir l'AST des fragments d'IUs textuelles en spécifications graphiques. Cette phase comprend un processus d'abstraction et un processus d'inférence. Dans le processus d'abstraction, on se sert de l'AST des IUs et les résultats du flot d'information, obtenus à l'étape précédente, pour construire des *spécifications textuelles*, aussi nommées *spécifications de base*. Celles-ci désignent les premiers concepts de design des IUs textuelles. Jusqu'à maintenant, on possède une représentation abstraite de la structure et des comportements des IUs textuelles. Cependant, avant de procéder à la conception des IUs graphiques, il nous faut des *spécifications graphiques* exprimées en terme d'objet. C'est la raison pour laquelle, à la suite de l'abstraction, on procède à une étape d'inférence où on prend les spécifications textuelles et on tente de les transformer en spécifications graphiques. Ces dernières serviront ensuite à générer de nouvelles IUs graphiques pendant l'étape de conception descendante.

5.1.4 Phase de conception descendante

Dans le cadre de leur projet pilote, les créateurs du AUIDL ont choisi Easel [2], un langage de développement d'interfaces graphiques, pour la génération des IUs graphiques. Les GUIs générées en Easel peuvent s'exécuter sur des postes de travail (PC) qui fonctionnent sur des systèmes d'exploitation Windows, OS/2 ou DOS. Un émulateur de terminal (voir la page 46) est intégré à l'environnement Easel et permet aux IUs graphiques de communiquer avec le système patrimonial hébergé sur un ordinateur central.

Comme nous l'avons mentionné, les spécifications objets des IUs textuelles sont écrites à l'aide du langage AUIDL. Dans la phase de conception descendante, la génération des GUIs commence par la transformation des spécifications objet AUIDL des IUs textuelles en spécifications EASEL des nouvelles GUIs. Un ensemble de règles de génération préside à cette transformation. Ces règles qui forment un analyseur, sont définies en utilisant un outil de l'environnement Refine [53], appelé *Refine and Dialect*. L'application de ces règles sur les spécifications AUIDL permet de les transformer en spécifications Easel, de façon automatique. La génération des IUs graphiques à partir des spécifications Easel se fait ensuite très facilement [25].

2. Easel est langage propriétaire développé par Easel Corporation. La compagnie Easel a été achetée par VMARK, en 1995.

5.1.5 Synthèse de la solution

Voici une énumération des avantages et des inconvénients de l'approche AUIDL [3].

+ le langage AUIDL peut exprimer de façon orientée objet et intuitive les spécifications d'IUs de systèmes patrimoniaux ;

- la refonte automatisée d'IUs, proposée dans AUIDL, est focalisée sur les systèmes patrimoniaux semi-décomposables. Les chercheurs s'intéressent plus particulièrement aux systèmes patrimoniaux Cobol/CICS dans lesquels la séparation du code des IUs des autres composants du système n'implique pas une analyse complexe de code. Conséquemment, l'emploi des méthodes proposées dans AUIDL pour la migration d'un système non décomposable risque de nécessiter des interventions manuelles, car l'analyse d'un code monolithique n'a pas été prévue dans la solution ;

- les étapes de conception descendante et d'intégration des IUs dans le nouvel environnement n'ont pas été abordées dans les discussions des chercheurs.

5.2 MORPH

Melody Moore et ses collègues à l'Institut de technologie de Georgia ont un regard différent sur le processus de migration d'IUs et adoptent une approche de gestion des connaissances (de l'anglais *knowledge-based*) pour la refonte d'IUs [46, 48, 49]. Les chercheurs proposent un ensemble de techniques et de méthodes pour automatiser le processus de refonte. Ces méthodes sont utilisées dans le développement d'une plate-forme de refonte d'IUs, appelée MORPH.

MORPH (pour *Model Oriented Reengineering Process for Human-Computer Interface*) est une plate-forme semi-automatisée de migration d'IUs patrimoniales vers des IUs graphiques. Étant donné qu'au moment de la réalisation de ces travaux de recherche l'accessibilité de services via le Web ou les réseaux interanet ne présentait pas encore un défi sérieux dans le domaine d'affaires, la migration d'IUs vers le Web ne fit pas l'objet des discussions des chercheurs. Il est toutefois possible de tirer profit des méthodes proposées dans MORPH afin d'automatiser la phase d'ingénierie-inverse dans une refonte d'IUs textuelles.

3. Nous avons pris la liberté d'exprimer les avantages avec (+) et les inconvénients avec (-).

MORPH se classifie dans la classe d'approches de greffage. Cependant, elle ne concerne pas une architecture en particulier, à savoir : non décomposable, semi-décomposable ou décomposable. L'approche est surtout concentrée sur le mappage d'éléments d'IUs patrimoniales et leurs équivalents dans des IUs graphiques.

La figure 5.7 illustre le modèle de refonte d'IUs dans MORPH. Dans ce modèle, les ovales représentent les étapes où une transformation de données se produit, c'est-à-dire que les données entrent sous une forme et sortent sous une autre forme. Les lignes en continu représentent des référentiels (de l'anglais *repositories*) externes de données - fichiers, bases de données ou bases de connaissances. Quant aux arcs, ils indiquent la direction des flots de données et finalement, le carré représente des interventions humaines faites généralement par des experts du domaine.

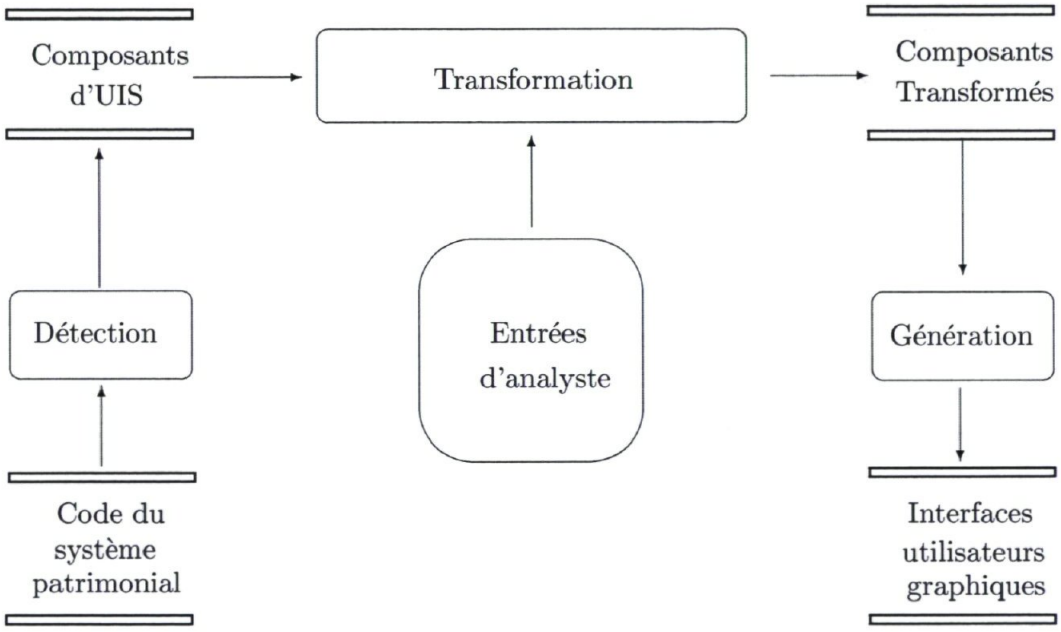

FIGURE 5.7 – Le processus de MORPH [50].

Le processus de refonte d'IUs dans MORPH se divise en trois étapes [46] :

- détection - identifie l'abstraction des éléments d'IUs dans le code d'un système patrimonial et modélise les abstractions identifiées. Les modèles abstraits sont répertoriés dans une base de connaissances ;
- transformation - transforme des modèles abstraits des éléments d'IUs patrimoniales en les modèles graphiques correspondant ;
- génération - génère des IUs graphiques pour l'environnement de migration et les intègre dans le code du système patrimonial.

5.2.1 Phase d'ingénierie-inverse

Melody Moore et ses collègues ont manuellement analysé le code source de 22 systèmes d'information patrimoniaux développés en différents langages : Ada, C, Cobol et Pascal. L'objectif de ces analyses consista à identifier les fonctionnalités que de tels systèmes offraient aux utilisateurs, c'est-à-dire à identifier les interactions possibles entre un système patrimonial et ses utilisateurs, que l'on nomme *tâches d'interaction* (de l'anglais *interaction tasks*). Cette expérience a montré que les systèmes patrimoniaux sont pour la plupart orientés-données, c'est-à-dire que leurs fonctions se limitent à la modification, la suppression et l'ajout de données. De ce fait, il est possible de concevoir un modèle général des tâches d'interaction et de le généraliser pour tout système patrimonial. Dans un tel modèle, chaque tâche d'interaction est présentée de façon abstraite. Par exemple, la *sélection d'une valeur* est l'abstraction d'une des tâches qu'un utilisateur peut effectuer. Ainsi on peut regrouper les tâches d'interaction de systèmes patrimoniaux en quatre catégories :

- sélection - faire un choix parmi une ensemble de choix ;
- quantification - introduire une valeur numérique dans un champ de saisie ;
- position - indiquer une position sur une IU ;
- saisie d'un texte - introduire un texte dans un champs de saisie.

Dans une IU, la réalisation d'une tâche d'interaction se fait à l'aide d'un de ses éléments, appelé *objet d'interaction* (de l'anglais *interaction objet*). Par exemple, un menu ou une liste déroulante sont des objets d'interaction à l'aide desquels la sélection d'une valeur peut se réaliser.

Bien que les abstractions des tâches d'interaction dans les différents langages d'IUs se ressemblent, l'implémentation de ces abstractions au niveau du code est différente d'un langage à l'autre. C'est la raison pour laquelle, trouver les patrons de code qui implémentent les abstractions des tâches d'interaction et des objets d'interaction dans un langage de programmation, implique l'analyse de plusieurs exemples de programmes qui sont écrits dans le langage en question.

Comme nous l'avons mentionné, dans MORPH, les patrons de code qui implémentent des tâches d'interaction - obtenues de l'analyse de plusieurs programmes - sont présentés dans un langage de représentation et répertoriés dans une base de connaissances. Cette base de connaissances (de l'anglais *knowledge base*) est donc composée de patrons de code (de l'anglais *pattern code*) et de règles heuristiques. Chaque patron de code décrit les fonctionnalités d'un élément d'IU, de façon abstraite et sous forme de phrase dans un langage de représentation ; dans MORPH, ce langage se nomme

CLASSIC [55]. Dans [47, 48], les chercheurs présentent en détail, la syntaxe et quelques exemples d'utilisation du langage CLASSIC.

L'étape d'ingénierie-inverse de MORPH se déroule ainsi :

- à l'aide de techniques d'analyse statique de code, on identifie toute tranche (de l'anglais *slice*) de code touchée par l'entrée-sortie d'utilisateur. L'analyse de flot de contrôle, l'analyse de flot de données et la recherche de patrons (de l'anglais *pattern matching*) sont des techniques d'analyse statique de code utilisés. L'ensemble des tranches de code constitue un sous-ensemble d'IUs (de l'anglais *user interface subset*) ;

- de manière automatique et à l'aide des patrons de code répertoriés dans la base de connaissances, on détecte les abstractions qui se trouvent dans le code des IUs textuelles. Ces abstractions sont répertoriées dans la base de connaissances.

5.2.2 Phase de conception descendante

La phase de conception descendante correspond à l'étape de transformation et de génération d'IUs. La première consiste à remplacer le patron de code de chaque élément d'IUs patrimoniales par son équivalent dans l'environnement graphique de migration. Pour ce faire, il est nécessaire d'avoir analysé et collecté les patrons de code des éléments d'IUs graphiques dans la base de connaissances. De cette manière, on est en mesure d'inférer les patrons de code des éléments graphiques à partir des abstractions textuelles à l'aide de règles heuristiques. Une fois que tous les éléments d'IUs graphiques sont définis, la seconde étape consiste à substituer les patrons de code des éléments graphiques par leur code respectif dans l'environnement graphique. Notons que dans MORPH, la dernière étape se fait manuellement.

5.2.3 Synthèse de la solution

Voici une énumération des avantages et des inconvénients de l'approche MORPH :

+ les méthodes et les techniques utilisés par les chercheurs permettent de traiter des systèmes patrimoniaux avec une complexité élevée au niveau de la séparation du code des IUs ;

+ les techniques et les méthodes de la séparation du code d'IUs dans MORPH permettent d'identifier et de collecter une partie des règles d'affaires qui concernent les données d'entrée. Cela a comme avantage de pouvoir concevoir des IUs graphiques qui exécutent des règles d'affaires ;

- dans une migration d'IUs, les abstractions du langage du code patrimonial et celles du code cible de migration doivent être déjà définies dans la base de connaissances ;

- la conception de la base de connaissances nécessite l'analyse manuelle de plusieurs programmes ;

- dans l'étape de transformation de MORPH, les règles heuristiques de la base de connaissances proposent généralement un ensemble d'éléments pour remplacer un élément d'une IU textuelle. Cela nécessite des interventions manuelles afin de sélectionner l'élément le mieux adapté.

5.3 CelLEST

Le projet CELLEST (pour *CEL Legacy Enhancement Software Technologies*) a été élaboré à l'Université d'Alberta sous la direction de la professeure Eleni Stroulia en collaboration avec la compagnie Celcorp. L'objectif du projet est de développer une méthode intelligente et semi-automatisée pour la migration d'IUs textuelles de systèmes patrimoniaux vers le Web [36, 72, 73, 74].

L'approche de migration d'IUs employée par CELLEST se classifie dans la classe d'approches frontales. Le choix de l'approche s'explique par le fait que le code source de systèmes patrimoniaux contient souvent du code mort et inutilisé ; la présence importante de codes morts empêche que l'analyse de code source soit précise et qu'elle reflète les vraies fonctionnalités de tels systèmes. De plus, dans certains cas, apporter des modifications au code source d'un système engendre des risques trop importants ou n'est tout simplement pas souhaitable. Ce sont les raisons pour lesquelles dans le processus de refonte d'IUs du projet CELLEST, les chercheurs optent pour l'ingénierie-inverse d'interactions basée sur les interactions utilisateurs-systèmes.

L'utilisation des techniques telles que, l'analyse de document, l'extraction des caractéristiques (de l'anglais *feature extraction*), l'analyse de regroupement (de l'anglais *cluster analyse*), la modélisation d'interactions (de l'anglais *user action modeling*), la visualisation, l'exploitation de données (de l'anglais *data mining*), l'inférence de modèles de tâches (de l'anglais *task model inference*), l'emballage avec XML (de l'anglais *XML wrapping*) et la génération automatique d'IUs permet à CELLEST d'automatiser une grande partie de la refonte d'IUs textuelles [24].

Notons que l'approche CELLEST dans le processus de refonte d'IUs textuelles est une approche basée tâches. Par conséquent, la génération d'IUs graphiques dans l'approche CELLEST est basée sur les tâches quotidiennes des utilisateurs. Cela mène à la conception des IUs graphiques qui représentent chacune plusieurs IUs textuelles. Comparativement aux techniques de refonte d'IUs traditionnelles qui génèrent une IU graphique pour une IU textuelle, CELLEST offre des IUs plus conviviales et plus faciles à utiliser.

L'environnement CELLEST est composé de trois prototypes : LENDI (pour *LEgacy Navigation Domain Identifier*) [21, 22, 20, 36, 73, 74], Mathaino [75, 35] et QANDA (pour Questions AND Answers). Les outils développés dans chaque prototype emploient différentes méthodes et algorithmes de l'intelligence artificielle pour automatiser le processus de refonte d'IUs. Les zones encadrées à la figure 5.8 représentent ces prototypes et les rôles que chacun joue dans l'ensemble du processus.

Ci-dessous, voici le déroulement du processus de refonte des IUs d'un système d'information patrimonial, dans CELLEST [24] :

- un intergiciel enregistre les traces des interactions entre le système et les utilisateurs. Il s'agit d'un émulateur de terminal qui peut être configuré pour enregistrer les interactions utilisateurs-systèmes. Notons que le protocole de communication émulé dans le cadre du projet pilote est un protocole basé blocs (voir la page 9). Dans ce contexte, les traces d'interactions comprennent les captures d'écran (de l'anglais *snapshots*) des IUs textuelles accédées par les utilisateurs, les actions des utilisateurs sur ces IUs - par exemple, les touches appuyées - pour naviguer d'une interface à une autre ; ainsi que les données d'entrée pendant l'enregistrement des interactions ;

- LENDI a comme fonction d'automatiser la modélisation de comportements des IUs textuelles. Cette modélisation repose sur l'analyse de traces d'interactions utilisateurs-systèmes. Les résultats de cette analyse sont des modèles états-transitions (de l'anglais *state-transition models*) où les états représentent des IUs textuelles

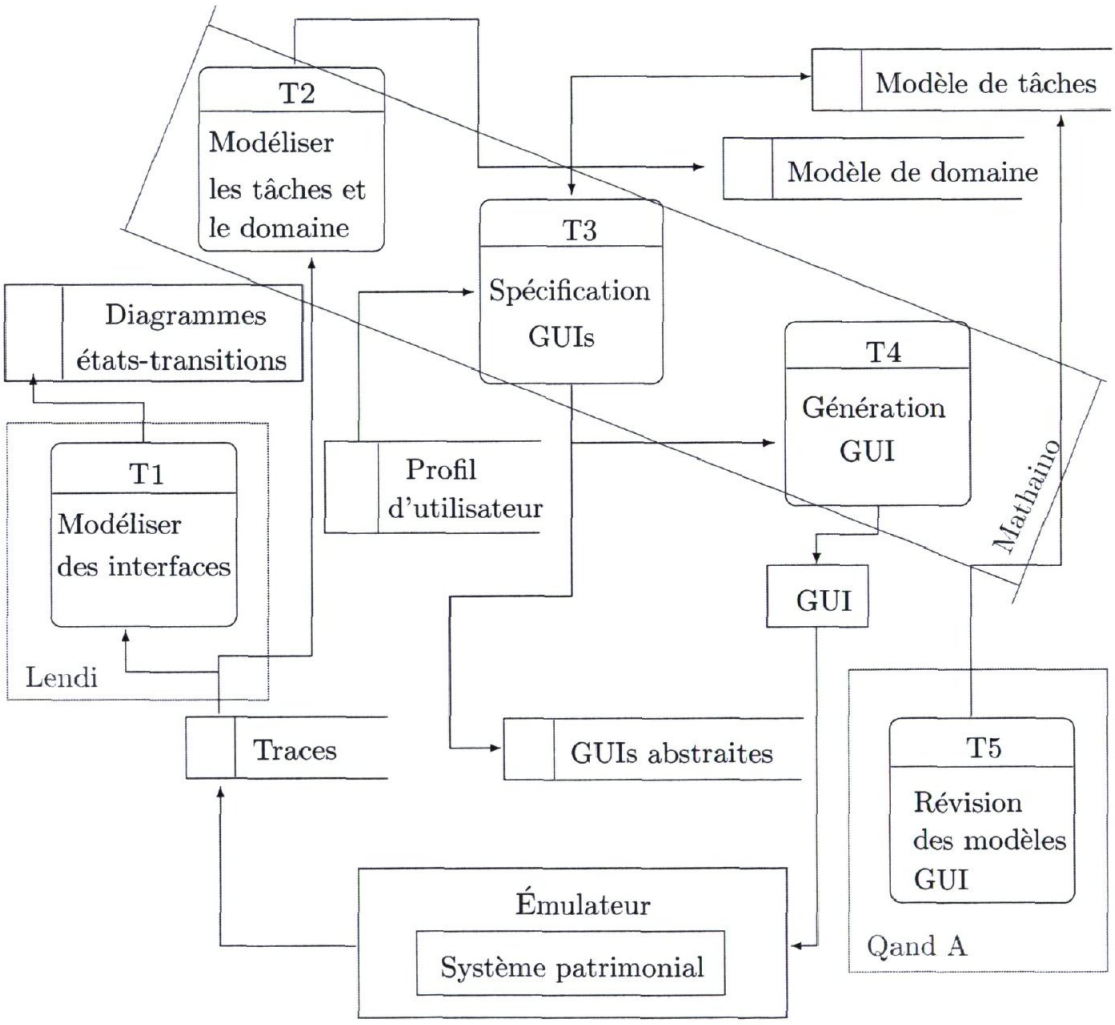

FIGURE 5.8 – Le processus de CelLEST [74].

du système et les transitions correspondent aux actions faisant naviguer le système d'une interface à une autre ;

- Mathaino obtient les modèles abstraits des IUs ainsi qu'un ensemble d'enregistrements spécifiques aux tâches, c'est-à-dire les interactions de l'utilisateur avec le système, pendant qu'il effectue une tâche spécifique. Chaque modèle de tâches est produit pour chacune des tâches. Un modèle de tâches spécifie le chemin, dans les modèles états-transitions, qu'un utilisateur prend pour effectuer la tâche en question. Les modèles de tâches forment la base pour la conception des spécifications abstraites des IUs graphiques en XML. Ces fichiers XML sont susceptibles de se transformer en format XHTML pour être utilisés sur les navigateurs Web, ou en format WML (pour *Wireless Markup Language*) pour être utilisés sur les appareils qui fonctionnent sur de protocoles sans fil (de l'anglais *Wireless Application Protocol* ou WAP) ;

- finalement, l'objectif du prototype QANDA est de permettre de visualiser les modèles générés par LENDI et Mathaino afin qu'un expert du domaine puisse les réviser et les valider.

5.3.1 Contexte d'essai

La série d'outils CELLEST est utilisée dans le cadre d'un projet de migration d'IUs pour une compagnie d'assurances qui désire donner un accès externe à certains services de son système à ses avocats qui ne travaillent pas dans les bureaux de la compagnie en question. Dans ce contexte, puisque la nature propriétaire du système empêche tout accès externe à ses services, la compagnie doit faire migrer son système vers une application client-serveur et préférablement vers une application Web.

Le système patrimonial qui fait l'objet de migration possède deux sous-systèmes : le système des données des clients et le système des services des réclamations. Au sein de la compagnie d'assurances, le système des données des clients permet la gestion des informations de la clientèle, alors que le système des services des réclamations sert à la génération de rapports de réclamations. Afin d'utiliser les services de génération de rapports, un utilisateur a besoin de connaitre le numéro de l'accident pour lequel il veut générer un rapport. Cependant, puisque la seule information disponible lors de l'utilisation de ce système est le nom du réclamant, l'utilisateur a l'obligation d'utiliser d'abord le système des données des clients afin de trouver le numéro de l'accident, pour ensuite être en mesure de générer le rapport de réclamation.

Comme mentionné, l'objectif du projet pilote est de donner un accès externe aux services d'un système patrimonial à un nouveau type d'utilisateurs. Les résultats de l'analyse préliminaire des besoins indiquent que :

- contrairement aux utilisateurs habituels du système, les nouveaux utilisateurs ne sont pas familiers avec les IUs textuelles du système ;
- l'utilisation du système par les nouveaux utilisateurs est restreinte et ne comprend qu'un nombre limité de services. Cependant, ils ont besoin d'utiliser un sous-ensemble des services des deux sous-systèmes.

Une solution classique de migration propose d'analyser le code source du système, afin d'isoler des modules de « recherche de numéro de réclamation » et « extraction des données de réclamation », nécessaires à la réalisation des tâches des nouveaux utilisateurs, d'en faire une application serveur et d'y donner accès via une application client. Toutefois, CELLEST propose une solution alternative : enregistrer les interactions utilisateurs-systèmes des tâches des nouveaux utilisateurs, modéliser les tâches des utilisateurs et utiliser une technique du mappage d'écran pour la migration des IUs textuelles vers une architecture Web.

Les sections suivantes décrivent les détails des deux étapes de refonte d'IUs dans l'approche de migration adoptée par CELLEST.

5.3.2 Phase d'ingénierie-inverse

Dans CELLEST, la phase d'ingénierie-inverse est basée sur l'analyse des traces d'interactions utilisateurs-systèmes. Comme nous l'avons mentionné, ces traces comprennent, entre autres, toutes les captures d'écran des IUs accédées par les utilisateurs pendant l'enregistrement des interactions.

Le premier objectif de la phase d'ingénierie-inverse est de concevoir les modèles états-transitions des comportements des IUs du système à l'étude. Pour ce faire, on a d'abord recours à des techniques d'analyse de document afin d'extraire les caractéristiques de chaque capture d'écran. Les résultats obtenus consistent en un vecteur de caractéristiques par capture d'écran : chaque vecteur comprend, entre autres, le contenu, le format et l'emplacement de l'en-tête, des caractères spéciaux figurant sur la capture d'écran ainsi que d'autres aspects visuels de cette dernière. Les vecteurs obtenus font ensuite l'objet d'une analyse à l'aide d'un algorithme de regroupements (de l'anglais *clustering algorithm*). Cette analyse fait en sorte de regrouper les captures d'écran selon leurs vecteurs de caractéristiques. La prochaine étape de modélisation des IUs consiste à

étudier les groupes formés des captures d'écran, afin d'induire un identificateur - prédicat - pour chaque groupe. L'induction de prédicat permet de concevoir les noeuds des modèles états-transitions. Ainsi, chaque noeud correspond à une IU textuelle pour laquelle on identifie plusieurs instances dans les traces d'interactions. Pour compléter la conception des modèles, il reste donc à définir les transitions entre les états. Appuyer sur une touche de fonction (de l'anglais *function key*), taper une commande, sélectionner une option d'un menu - dans des systèmes pilotés par menu (de l'anglais *menu-based systems*) - sont les exemples d'actions qu'un utilisateur peut effectuer afin de naviguer dans un système patrimonial. L'analyse des données des traces d'interactions permet d'identifier et de modéliser les actions des utilisateurs sur chaque IU pour naviguer entre les IUs.

Déterminer les données d'entrée, les données de sortie et la séquence des IUs textuelles impliquées dans la réalisation des tâches des utilisateurs constituent la prochaine étape de l'ingénierie-inverse dans l'approche CELLEST. Il s'agit donc d'identifier et d'extraire des sous modèles états-transitions représentant des tâches quotidiennes d'un utilisateur type. Pour ce faire, on fait appel à des techniques d'exploitation de patrons (de l'anglais *pattern mining*) pour analyser les traces d'interactions et extraire des modèles correspondants. Ces modèles de tâches servent de base dans la phase de conception descendante pour la génération des modèles de spécifications des IUs graphiques en format XML.

5.3.3 Phase de conception descendante

Dans l'environnement CELLEST, l'étape de conception descendante prend en entrée les modèles de tâches obtenus à l'étape d'ingénierie-inverse et produit des spécifications graphiques des nouvelles IUs. Une spécification est une forme abstraite qui représente un sous-ensemble des IUs textuelles à l'aide desquelles la réalisation d'une tâche est possible. Chaque forme est composé des champs d'entrée et de sortie des nouvelles IUs graphiques. La génération des spécifications des IUs graphiques est automatisée et se fait en utilisant des heuristiques de design d'IUs à base de modèles (de l'anglais *model-based UI design heuristics*). Considérant un modèle de tâches, les heuristiques permettent de proposer un élément graphique abstrait pour chaque élément du modèle de tâches. Par exemple, un champ dont le contenu est « imprévisible » peut être remplacé par un champ texte, alors qu'un champ avec une gamme énumérée de valeur peut être remplacé par une case à cocher ou une case d'options.

Comme nous l'avons mentionné, un des avantages de l'approche CELLEST est qu'elle propose une solution multiplate-forme pour la génération d'IUs graphiques. Sur la plate-

forme CELLEST, deux interpréteurs - pour générer des IUs en XHTML et en WMP - sont développés. Ils prennent les spécifications abstraites en XML et génèrent des IUs graphiques correspondant aux spécifications.

5.3.4 Synthèse de la solution

Voici une énumération des avantages et des inconvénients de l'approche CELLEST :

+ l'utilisation d'une approche orientée tâches dans CELLEST permet de bonifier l'aspect et la convivialité des nouvelles IUs ;

+ les deux étapes de refonte d'IUs ont été complètement couvertes par les chercheurs. Ces derniers proposent, entre autres, des techniques pour la génération d'IU ;

- la modélisation des IUs textuelles d'un système nécessite que tous les cas d'utilisation du système soient reproduits.

5.4 Retour expérience

Dans ce mémoire, nous avions comme objectif de présenter une synthèse complète de l'état de l'art dans le domaine de la migration d'IUs patrimoniales laquelle couvre différentes solutions proposées dans ce domaine. Bien que nous ne sommes pas entrés dans les détails techniques de toutes les approches et les solutions présentées, ces détails sont facilement accessibles à l'aide de la bibliographie que nous croyons complète.

Nous avons principalement focalisé notre recherche sur les travaux de recherche universitaires. L'étude de la littérature nous a montré que les recherches sur le sujet abordé tout au long de ce document, c'est-à-dire la migration d'IUs patrimoniales, sont pratiquement abandonnées depuis environ 10 ans et que les articles présentés dernièrement à des conférences portant sur la maintenance et la réingénierie de logiciel - comme ICSM (pour *International Conference on Software Maintenance*) et CSMR (pour *European Conference on Software Maintenance and Reengineering*) - sont dans la majorité des cas des rapports sur des expériences pratiques au niveau de l'industrie. Ces rapports

résument pour la plupart des activités de migration effectuées au niveau d'un système patrimonial ; la migration d'IUs ne faisant pas l'objet des discussions.

Étant donné la réalité de la présence importante des systèmes patrimoniaux sur le marché et que ce domaine représente toujours un domaine actif au niveau de l'industrie, dans cette section, nous récapitulons nos constats pour ce qui est de l'application des méthodes et des solutions proposées de migration d'IUs par des chercheurs universitaires dans l'industrie.

En pratique, chaque système patrimonial a ses propres particularités. De ce fait, dans la migration d'IUs patrimoniales, des techniques génériques qui, en théorie, donnent généralement des résultats satisfaisants, doivent être adaptées aux spécificités de chaque système. Par conséquent, pour réussir une migration de système patrimonial, il est nécessaire de personnaliser ces techniques en fonction des particularités du système faisant l'objet de migration et d'effectuer certaines étapes de migration manuellement. De plus, pour pouvoir réaliser une migration d'IUs en utilisant des techniques étudiées dans ce mémoire, il est primordial que la firme à laquelle une migration est confiée dispose des outils d'automatisation du processus de migration ainsi que du personnel avec des connaissances approfondies dans chacun des domaines mentionnés pour chaque technologie. Seules les firmes spécialisées dans le domaine de la migration de systèmes patrimoniaux possèdent une telle expertise.

Des firmes de consultants, quant à elles, ont généralement de la difficulté à justifier le budget nécessaire pour le développement des outils ou encore à former et à garder en disponibilité une équipe spécialiste en migration de systèmes patrimoniaux. Premièrement, pour être compétitive sur le marché de solutions de migration de systèmes, toute firme répondant à des appels d'offres - lancés par des clients potentiels - doit présenter son approche et ses outils d'automatisation du processus de migration aux clients et être en mesure de prouver que son approche est appropriée et qu'elle est assez robuste pour faire le travail de migration. C'est-à-dire que le temps mis pour le développement des outils en question n'est rentable que si un client s'intéresse à la solution proposée par la firme et que la firme gagne le contrat de migration. Pour une firme de consultants pour laquelle la rentabilité instantanée d'un projet présente un des facteurs critiques pour l'approbation de l'allocation de budget au projet, il est difficile de justifier cette dépense. Deuxièmement, compte tenu des particularités de chaque système patrimonial, la firme ayant gagné un contrat de migration de systèmes doit préférablement disposer d'une équipe spécialisée dans la migration de systèmes patrimoniaux pour effectuer des tâches manuelles de migration et pour adapter, au besoin, la solution de migration. Avoir en disponibilité une équipe spécialisée qui ne travaille pas sur des projets instantanément facturables est également difficile à justifier.

Considérant les éléments mentionnés ci-dessus, dans le domaine de migration de systèmes patrimoniaux, l'industrie a un grand intérêt pour des approches et des méthodes simples à mettre en place et compréhensibles par la majorité des développeurs. De ce fait, des technologies et des méthodes proposées par des chercheurs universitaires, qui demandent pour la plupart des investigations importantes, sont habituellement négligées. De plus, comme nous l'avons mentionné au chapitre 3, les techniques de migration d'IUs patrimoniales présentées dans ce mémoire représentent pour la plupart des travaux de recherche universitaires dont l'efficacité et le bon fonctionnement ont été testés dans le cadre des projets pilotes de petites envergures. Le fait que ces méthodes n'aient pas été testées et approuvées dans le cadre de grands projets fait en sorte qu'il existe toujours des risques qu'elles ne soient pas assez robustes et efficaces.

Maintenant que nous avons discuté des enjeux importants du côté de l'industrie, il est aussi intéressant de connaitre les préoccupations des propriétaires des systèmes patrimoniaux lorsqu'ils décident de migrer leur système. En effet, le propriétaire d'un système patrimonial a aussi des contraintes budgétaires à respecter. Pour que la migration de son système patrimonial soit justifiable, les coûts associés à cette migration doivent nécessairement être beaucoup moins élevés que le remplacement de son système. Si une migration - incluant la formation des utilisateurs du système et l'achat du nouveau matériel - coûte le même prix qu'un remplacement, de toute évidence, le propriétaire du système patrimonial sera plus intéressé à remplacer son vieux système par des technologies de pointe, car les résultats de ce remplacement sont normalement plus prometteurs comparativement à ceux d'une migration de système.

Lorsqu'un client décide de migrer son système patrimonial, certains éléments influencent ses choix de technologies cibles de migration. Plus souvent, il préfère migrer son système patrimonial dans les mêmes technologies qu'il utilise pour le développement de ses nouveaux systèmes. Le développement de nouveaux systèmes au sein d'un organisme se fait très souvent en utilisant des nouvelles technologies du marché. Ce qui n'empêche pas l'organisme de continuer d'utiliser et de maintenir son système patrimonial. En migrant son système patrimonial en utilisant les technologies connues par ses employés, la maintenance du système migré ainsi que l'adaptation des différents systèmes entre eux se feront plus facilement.

L'autre élément clé qui peut entrer en ligne de compte, surtout dans la migration des IUs d'un système patrimonial, est le choix des utilisateurs finaux par rapport à la technologie cible de migration d'IUs. Si dans un organisme, les utilisateurs finaux connaissent très bien le système patrimonial et maîtrisent son utilisation, malgré le fait que les IUs ne sont généralement pas conviviales, il y a de fortes chances que des responsables de la migration préfèrent que l'aspect et la convivialité (de l'anglais *look and feel*)

des IUs migrées dans la nouvelle technologie soient exactement les mêmes que ceux du vieux système. Premièrement, former les utilisateurs pour utiliser le nouveau système n'est pas nécessaire ; deuxièmement, la résistance aux changements sera moins importante chez les utilisateurs. Dans ce contexte, l'analyse complète du système patrimonial et l'utilisation des techniques présentées dans ce mémoire, afin d'améliorer l'aspect et la convivialité des nouvelles IUs, ne semblent pas être nécessaires.

Dans le cas où il est désiré d'apporter des changements radicaux à l'utilisation des IUs d'un système patrimonial, une des deux options suivantes est normalement sélectionnée :

1. utilisation des techniques mentionnées dans ce document - lorsque le propriétaire du système patrimonial fait affaire avec une firme spécialisée dans le domaine de migration de systèmes patrimoniaux, il peut compter sur leur expertise pour utiliser le code du système patrimonial, afin d'améliorer l'aspect et la convivialité des nouvelles IUs ;
2. génération des nouvelles IUs en commençant l'analyse et le développement à zéro (de l'anglais *from scratch*) - lorsque le système patrimonial qui fait l'objet de migration possède une architecture semi-décomposable ou décomposable, la séparation du code des IUs du code du reste du système permet de générer des nouvelles IUs en se basant sur l'aspect visuel (de l'anglais *look*) des IUs patrimoniales. Dans l'industrie cela est fait très souvent manuellement par des développeurs. Cette opération est normalement chronophage.

Conclusion

Les activités de migration font partie du cycle de vie de tout logiciel applicatif, y compris des systèmes d'information. La migration d'IUs, en soi, s'inscrit rarement dans les activités de migration comme étant le seul objectif de cette opération. La migration d'IUs est habituellement accompagnée d'un changement d'architecture ou fait partie de la migration complète d'un système patrimonial. Néanmoins, la migration d'IUs vers de nouvelles technologies de présentation, en général, et la migration d'IUs vers des technologies de présentation Web, en particulier, peuvent être étudiées en tant qu'un processus distinct des activités de migration. La raison s'explique ainsi :

- un grand nombre de systèmes d'information patrimoniaux qui font l'objet de migration utilisent de services de moniteurs de télétraitement. Il s'agit donc de systèmes patrimoniaux semi-décomposables dont les composants d'IUs sont, par définition, déjà séparés des autres composants et peuvent être traités indépendamment ;

- les technologies de présentation, utilisées pour le développement des composants d'IUs, sont différentes des technologies utilisées pour le développement des autres composants d'un logiciel ;

- de nos jours et dans le développement de logiciels, l'accent est de plus en plus mis sur la séparation des différents composants, et ce, afin de faciliter le développement et la maintenance des logiciels conçus. Cela permet de pouvoir travailler sur les IUs comme des unités distinctes, sans avoir à se soucier du développement des autres composants de logiciel.

Dans ce mémoire, nous avons concentré notre étude sur la migration d'IUs textuelles vers des technologies de présentation Web. Puisque cette migration présente un cas spécial de migration d'IUs textuelles vers des IUs graphiques, dans les deux cas, une grande partie du processus de migration se ressemble. La différence se trouve seulement au niveau de la technologie de présentation de migration et l'architecture à laquelle les

IUs graphiques s'intègrent.

Nous avons divisé les solutions de migration d'IUs en deux grandes classes d'approches :

- l'approche frontale - lorsqu'on fait face à la migration d'un système patrimonial non décomposable, l'approche frontale est utilisée. Cette classe d'approches offre une solution rapide et peu coûteuse, mais à court terme, à la migration d'IUs textuelles ;

- l'approche de greffage - pour la migration des systèmes semi-décomposables ou décomposables, on a une préférence pour l'utilisation de l'approche de greffage, dans la migration d'IUs. Cette classe d'approches permet de séparer les IUs d'un système patrimonial du reste du système. Les IUs sont ensuite analysées, afin de concevoir de nouvelles IUs graphiques. Ainsi, les IUs graphiques de migration ont l'avantage de pouvoir être maintenues séparément. De plus, il est possible d'améliorer et de moderniser l'aspect et la convivialité des IUs après la migration, et ce, sans avoir à toucher au reste du système.

Dans le processus de migration d'IUs vers le Web, l'objectif est d'intégrer le système patrimonial dans une architecture Web et d'y accéder via des IUs développées à l'aide des technologies de présentation Web. Encore une fois, le choix d'architecture varie d'un projet de migration à un autre et dépend des préférences du client et de l'architecture conceptuelle du système patrimonial qui fait l'objet de la migration. Cependant, la majorité des architectures logicielles Web de migration partage la caractéristique de posséder une application Web client léger (de l'anglais *thin client*) où les IUs n'exécutent pas de règle d'affaires. Le choix de l'architecture Web de migration définit aussi la(les) technologie(s) de présentation Web dans lesquelles des nouvelles IUs seront développées. Ensuite, vient le temps de modéliser les IUs textuelles - l'ingénierie-inverse - et de générer des IUs Web - la conception descendante - en se basant sur les informations obtenues. Nous avons utilisé le terme « refonte » pour décrire ce processus.

Comme nous l'avons constaté, la réalisation manuelle de la refonte d'IUs est une tâche chronophage (de l'anglais *time consuming*) et pratiquement impossible. De ce fait, l'automatisation complète de ce processus était, depuis toujours, un objectif important à atteindre. D'une part, afin d'économiser le temps et, d'autre part, pour pouvoir justifier un projet de migration. Dans la littérature, plusieurs chercheurs ont proposé des techniques et des méthodes afin d'automatiser le processus de refonte d'IUs.

Dans ce mémoire, nous avons tenté de rassembler les techniques d'automatisation du processus de refonte d'IUs et de les présenter brièvement. Notons qu'une grande partie des techniques exposées dans ce document représentent un domaine de recherche à part entière. C'est la raison pour laquelle nous nous sommes limités à donner une description sommaire pour chaque technique et à fournir des ressources supplémentaires à notre lecteur. Cela permet aux intéressés d'en savoir plus sur le sujet.

À titre d'exemple, nous avons présenté les résultats des travaux de trois équipes universitaires dans le domaine de la migration d'IUs, lesquels ont focalisé leur attention sur l'automatisation de refonte d'UIs patrimoniales vers de nouvelles technologies. Bien que les travaux de deux de ces trois équipes datent les années 90, on peut toujours réutiliser une bonne partie de leur expérience, surtout dans les étapes de décomposition et d'ingénierie-inverse. Toutefois, l'arrivée de nouvelles technologies de présentation Web nous encourage à développer des IUs Web de migration pour ces nouvelles technologies. Nous avons donc brièvement fait référence à ces plate-formes de migration d'IUs en nous attardant sur les parties pouvant être réutilisées dans une migration d'IUs textuelles vers le Web.

Le résultat de notre étude sur la migration des IUs textuelles de systèmes d'information patrimoniaux vers le Web est une synthèse de l'état de l'art dans ce domaine de recherche. En réunissant une grande partie des travaux de migration d'IUs, nous espérons que ce document pourra servir de point de départ pour toute recherche sur la migration d'IUs.

Lexique

Analyse de regroupement L'*analyse de regroupement* (de l'anglais *cluster analysis*) est une technique statistique qui permet la formation de groupes dont les unités présentent une corrélation plus forte entre elles qu'avec les unités des autres groupes[4]. 89

Analyse statique En informatique, la notion d'*analyse statique* de programmes couvre une variété de méthodes utilisées pour obtenir des informations, sans l'exécuter, sur le comportement dynamique d'un programme. C'est cette restriction qui distingue l'analyse statique des analyses dynamiques qui nécessite l'exécution d'un programme pour en obtenir des informations. 59, 71

Applet Java Un *applet Java* est une petite application écrite en langage Java qui, insérée dans un document Web, exécute ses objets multimédias en présence d'un navigateur compatible, et ce, directement sur l'ordinateur de l'internaute, peu importe le système d'exploitation utilisé. Cette caractéristique, appelée *portabilité*, s'explique par le fait que les applets Java sont écrits en code intermédiaire (de l'anglais *bytecode*) et non pas en code binaire. Par conséquent, ils peuvent s'exécuter sur des plate-formes matérielles différentes. 47

Application Web client léger Dans le cadre d'une application Web, lorsque la partie de la gestion de traitements est complètement réalisée par le serveur, on parle de *client léger* (de l'anglais *thin client*). Des pages Web dynamiques ainsi que des applications d'Ajax sont les exemples de clients légers. 43, 100

Arbre syntaxique abstrait - AST Un *arbre syntaxique abstrait* (de l'anglais *Abstract Syntax Tree* ou AST) est un arbre dont les noeuds internes sont marqués par des opérateurs et dont les feuilles - ou noeuds externes - représentent les opérandes de ces opérateurs. 78, 82

ASP *Active Server Pages* est un ensemble de logiciels développés par Microsoft et utilisés dans la programmation Web. C'est une suite de logiciels destinée à créer des

4. La principale source d'information pour ce lexique : Wikipédia et l'Office Québécois de la langue fraçaise. Notons que pour chaque entrée du lexique figurent des pointeurs vers les pages du mémoire où les différentes notions sont utilisées.

sites web dynamiques. Elle nécessite pour fonctionner une plate-forme Windows avec IIS installé, ou encore une plate-forme Linux ou Unix avec une version modifiée d'Apache. ASP est une structure composée d'objets accessibles par deux langages principaux : le VBScript et le JScript. Il est possible d'utiliser d'autres langages comme PerlScript, REXX, ou encore Python en ajoutant le moteur d'interprétation du langage adéquat à IIS. 33, 43, 74

Assistant numérique personnel Un *assistant numérique personnel* est un appareil numérique portable, souvent appelé par son sigle anglais PDA pour Personal Digital Assistant. 73

Base de connaissances Une *base de connaissances* est un élément d'un système expert contenant des informations représentant les connaissances acquises dans un domaine particulier[5]. 85, 86

Base de données relationnelle En informatique, une *base de données relationnelle* est un stock de données décomposées et organisées dans des matrices appelées relations ou tables conformément au modèle de données relationnel. Le contenu de la base de données peut ainsi être manipulé par des opérations d'algèbre relationnelle telles que l'intersection, la jointure et le produit cartésien. Une base de données est un ensemble de données, connexes de manière directe ou indirecte, enregistrées dans un dispositif informatique. Dans une base de données relationnelle, les informations sont stockées sous forme de groupes de valeurs : les enregistrements. Un ensemble d'enregistrements relatif à un sujet forme une relation et est stocké dans une table. La base de données comporte une ou plusieurs tables et les sujets sont connexes. 56

Environnement d'exécution Java L'environnement d'exécution Java (Jave Runtime Environment) désigne un ensemble d'outils permettant l'exécution de programmes Java sur toutes les plates-formes supportées. JRE est constitué d'une JVM (*Java Virtual Machine* - Machine Virtuelle Java), le programme qui interprète le code Java compilé (bytecode) et le convertit en code natif. Mais le JRE est surtout constitué d'une bibliothèque standard à partir de laquelle doivent être développés tous les programmes Java. C'est la garantie de portabilité qui a fait la réussite de Java dans les architectures client-serveur en facilitant la migration entre serveurs, très difficile à réaliser sur les gros systèmes. 47

Exploitation de données L'*exploitation de données* (de l'anglais *data mining*) est la technique de recherche et d'analyse de données, qui permet de dénicher des tendances ou des corrélations cachées parmi des masses de données, ou encore de détecter des informations stratégiques ou de découvrir de nouvelles connaissances en s'appuyant sur des méthodes de traitement statistique. 89

5. La traduction du terme « knowledge base » selon l'office québécois de la langue française (OQLF).

Extensibilité Selon l'OQLF, l'extensibilité d'un système est son aptitude à conserver ses propriétés fonctionnelles malgré un changement de taille ou de certains paramètres - par exemple l'augmentation du nombre d'utilisateurs pour un serveur. 38, 45

Extraction des caractéristiques L'*extraction des caractéristiques* (de l'anglais *feature extraction*) est l'étape du traitement de l'image où seules les caractéristiques pertinentes de l'objet contenant le maximum d'informations sont retenues en vue de modéliser cet objet pour sa classification. 89

Fiabilité Selon l'Office québécois de la langue française (OQLF), la *fiabilité* est la propriété d'un système informatique qui est capable d'assurer ses fonctions sans défaillance, dans des conditions préalablement définies et sur une période déterminée. 45

Fichier indexé Un *fichier indexé* est un fichier de données dont le contenu est trié selon la valeur d'une clé unique. Cette dernière permet de trouver un enregistrement parmi tous les enregistrements stockés dans le fichier. Un fichier indexé joue le rôle d'une bases de données dans les systèmes modernes. 8, 55

Hôte Internet Ordinateur relié directement à Internet par le protocole TCP/IP (TCP pour *Transmission Control Protocol* et IP pour *Internet Protocol*) et possédant sa propre adresse IP. 30

Indexed Sequential Access Method - ISAM *ISAM* est une méthode d'indexation de données pour permettre une recherche rapide. Dans un système ISAM, les données sont organisées dans des enregistrements de taille fixe. Les index constituent un ensemble de tables de hachage qui contient des pointeurs sur les données stockées. Puisque les index ne sont pas volumineux, la recherche de données se fait rapidement. 7

Instruction GOTO L'*instruction GOTO* (de l'anglais *go to*, en français aller à) est utilisée pour réaliser des sauts inconditionnels dans un programme. L'exécution est renvoyée vers une étiquette (label), qui est soit un numéro de ligne donné, soit une étiquette déclarée, selon le langage. L'instruction GOTO, directement héritée des instructions de saut des langages machines, était nécessaire dans les langages primitifs (Fortran pré-90, Cobol, Basic) comme instruction de base permettant de réaliser des boucles et autres structures de contrôles. Depuis la révolution de la programmation structurée des années 1970, l'instruction GOTO n'est plus guère appréciée des programmeurs modernes, car elle rend souvent les programmes plus difficiles à comprendre et à maintenir (on parle dans ce cas de programmation spaghetti). 22

JavaServer Page Technology - JSP Le *JavaServer Pages* ou JSP est une technique basée sur Java qui permet aux développeurs de créer dynamiquement du code

HTML, XML ou tout autre type de page Web. Cette technique permet au code Java et à certaines actions prédéfinies d'être ajoutés dans un contenu statique. Depuis la version 2.0 des spécifications, la syntaxe JSP est complètement conforme au standard XML. 33, 61, 74

Langage de script-serveur Un *langage de script-serveur* est un langage de script qui peut être exécuté par un serveur. Les langages de script sont interprétés et faciles à apprendre. Les langages de script les plus reconnus sont Perl, Tcl, Python, JavaScript. 32

Langage impératif Un *langage impératif* est un langage de programmation qui utilise une séquence d'instructions pour atteindre un objectif. Ainsi, chaque instruction a un impact sur l'état général du programme. Java est un exemple de langage impératif. 35

Migration Incrémentale Lorsque l'approche et les méthodes utilisées dans la migration d'un système patrimonial nous permettent de retravailler sur un module migré et déjà intégré, de l'améliorer et de le réintégrer dans l'architecture, la migration se nomme *incrémentale*. Le principal avantage d'opter pour une solution incrémentale est que dans cette approche, le système patrimonial est migré en petits morceaux. Ainsi, l'intégration des composants à l'architecture de migration se fait au fur et en mesure qu'ils sont migrés et si un composant ne répond pas aux besoins, il est possible de réviser la méthode de migration et reconstruire le composant en question. 59

Modèle-vue-contrôleur - MVC Le patron de conception *modèle-vue-contrôleur* (MVC) décrit une manière d'organiser le code d'une application en séparant les données d'une application, l'interface d'utilisateur, et la logique de contrôle dans trois composants différents : le *modèle* implémente les règles d'affaires. Il est composé d'objets ayant comme rôle de gérer la logique d'affaire et d'interagir avec les données. Le *contrôleur* reçoit la demande de l'utilisateur par l'entremise de la vue et la communique avec le modèle. Il interagit ensuite avec la vue afin de lui envoyer les réponses obtenues. Finalement, les composants de la *vue* sont responsables de la génération des interfaces utilisateurs dans une technologie de présentation. 60

Noeud de réseau Un *noeud de réseau* est un ordinateur personnel ou autre unité connecté au réseau par l'intermédiaire d'une carte de réseau ou d'un pilote de réseau local. Un noeud de réseau peut réunir un serveur, un poste de travail, un routeur, une imprimante ou un télécopieur. 28

Ordinateur Personnel Les ordinateurs personnels ont commencé à apparaître à la fin des années 70. Un des premiers modèles était Apple II. Il a été mis sur le marché par la compagnie Apple en 1977. IBM a produit ses premiers ordinateurs personnels en 1981. L'idée était d'avoir un ordinateur moins cher et moins

performant que les ordinateurs centraux, mais capable d'exécuter ses propres applications. Contrairement aux terminaux muets servant uniquement à saisir les données dans le système et à afficher les résultats, un ordinateur personnel est un système complet possédant un système d'exploitation et étant en mesure d'exécuter des morceaux de la logique d'application en plus de la saisie et l'affichage de données. 46

Petites et moyennes entreprises - PME Entreprise qui est considérée comme étant de petite ou de moyenne importance en raison du nombre de ses salariés, de son chiffre d'affaires ou du total de son actif. 59

PHP (Hypertext Preprocessor) L'*Hypertext Preprocessor*, plus connu sous son sigle PHP, est un langage de script libre, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale, en exécutant les programmes en ligne de commande. PHP est un langage impératif disposant, depuis la version 5, de fonctionnalités complètes du modèle objet. En raison de la richesse de sa bibliothèque, on désigne parfois PHP comme une plate-forme plus qu'un simple langage. 33

Progiciel prêt à utiliser (COTS systems) Des *progiciels prêts à utiliser* (de l'anglais *off-the-shelf commercial systems* ou COTS systems) sont des systèmes désignés et fabriqués en grande série, et qui ne visent pas un besoin particulier. L'utilisation d'un tel système dans un projet réduit des coûts de conception et de maintenance. Cependant le système doit être personnalisé afin de répondre aux besoins propres au projet. 26

Propriétaire Un produit *propriétaire* est un produit informatique qui est spécifique à un constructeur ou un développeur donné, ce qui veut dire qu'il n'est pas nécessairement conforme à une norme ou un standard, qu'il n'est pas toujours compatible avec d'autres produits, qu'il est protégé par le droit d'auteur et qu'il faut l'acheter ou acquérir une licence pour pouvoir l'utiliser. 1, 23, 35

Prototype Un *prototype* est un modèle ou mise en oeuvre préliminaire permettant l'évaluation de la conception d'un système, de sa réalisation et de son potentiel d'exploitation, ou encore une meilleure identification et compréhension des besoins. 89

Servlet Java Un *servlet* est une classe Java qui permet de créer dynamiquement des données au sein d'un serveur HTTP. Ces données sont généralement présentées avec le format HTML, mais elles peuvent également l'être au format XML ou tout autre format destiné aux navigateurs web. Un servlet s'exécute dynamiquement sur le serveur web et permet son extension : accès à des bases de données, transactions d'e-commerce, etc. Un servlet peut être chargé automatiquement lors

du démarrage du serveur web ou lors de la première requête du client. Une fois chargées, les servlets restent actifs dans l'attente d'autres requêtes du client. 61

Slicing Le *programme slicing* a été introduit par Mark Weiser. L'idée de base était de faciliter le processus de débogage d'un système complexe. Cependant cette technique a été utilisée pour la compréhension de programmes, la maintenance des logiciels, la mise à l'essai (de l'anglais *testing*), l'optimisation, l'intégration de programmes, etc. Le programme slicing peut être également utilisé pour la décomposition d'un programme en utilisant l'analyse de flot de contrôle et l'analyse de flot de donnée. 53, 60, 82

SNA *SNA* (pour *System Network Architecture*) est une architecture de communication de données définie par IBM en 1974. Ce protocole a comme objectif d'établir un ensemble de conventions de communication pour le matériel et les logiciels des produits IBM. 47

SOAP *SOAP* est un protocole de transmission de messages. Il définit un ensemble de règles pour structurer des messages qui peuvent être utilisés dans de simples transmissions unidirectionnelles. Il est particulièrement utile pour exécuter des échanges requête-réponse RPC (Remote Procedure Call). SOAP est indépendant de tout langage et système d'exploitation. Ainsi, les clients et serveurs de ces dialogues peuvent tourner sur n'importe quelle plate-forme et être écrits dans n'importe quel langage du moment qu'ils puissent formuler et comprendre des messages SOAP. Il s'agit donc d'un important composant de base pour développer des applications distribuées qui exploitent des fonctionnalités publiées comme services Web par des intranets ou Internet. 62

Standards ouverts Il n'existe pas une définition universelle pour le terme « standards ouverts ». Dans ce document, nous nous référons à la définition donnée par le gouvernement français, au travers de la LCEN[6]. Il définit un standard ouvert comme suit : « On entend par standard ouvert tout protocole de communication, d'interconnexion et d'échange et tout format de données interopérable et dont les spécifications techniques sont publiques et sans restriction d'accès ni de mise en oeuvre ». 27

Système d'information Un *système d'information d'entreprise* ou tout simplement *système d'information*, est un ensemble de programmes servant à automatiser certaines tâches dans le domaine d'affaire. Fréquemment, il est conçu pour traiter des quantités importantes de données. Ainsi, la création, la suppression et la modification des éléments de données sont les fonctionnalités les plus fréquentes d'un tel système. 1, 5

Système de fichiers Un *système de fichiers* (de l'anglais *file system*) ou système de gestion de fichiers (SGF) est une façon de stocker les données et de les organiser

6. Loi pour la confiance dans l'économie numérique, Loi 2004-575 du 21 juin 2004.

dans des fichiers sur ce que l'on appelle des mémoires secondaires (disque dur, clé USB, etc.). Une telle gestion des fichiers permet de traiter, de conserver des quantités importantes de données ainsi que de les partager entre plusieurs programmes informatiques. Il offre à l'utilisateur une vue abstraite sur ses données et permet de les localiser à partir d'un chemin d'accès. Il existe d'autres façons d'organiser les données, par exemple les bases de données et les fichiers indexés. 7, 30

Uniform Resource Locator - URL *URL* (littéralement « localisateur uniforme de ressource ») se dit d'une chaîne de caractères normalisés servant à identifier et à localiser des ressources consultables sur Internet et à y accéder à l'aide d'un navigateur. 30

Utilisabilité L'*utilisabilité* est une caractéristique de qualité liée à la facilité à se servir de quelque chose. Plus précisément, il s'agit de la rapidité avec laquelle des personnes peuvent apprendre à l'utiliser, de son efficacité, de sa rémanence, de son taux d'erreur et de sa facilité d'utilisation. 38

Virtual Storage Access Method - VSAM *VSAM* est une méthode de stockage informatique de données utilisée sur les systèmes z/OS. Cette méthode d'accès comprend l'organisation des données, les techniques d'accès à ces données et des outils de maintenance. Physiquement, un fichier VSAM (data set) est stocké sur disque sous la forme de blocs physiques regroupés en extents. Logiquement, un fichier VSAM est un espace-octet adressable. 7

W3C Le *World Wide Web Consortium*, abrégé par le sigle W3C, est un organisme de normalisation à but non-lucratif, fondé en octobre 1994 comme un consortium chargé de promouvoir la compatibilité des technologies du World Wide Web telles que HTML, XHTML, XML, RDF, SPARQL, CSS, PNG, SVG et SOAP. 27

Bibliographie

[1] *Ajax : A new approach to web applications*, `http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications`, Accédé : 2012/12/10.

[2] Alfred V. Aho, Monica S. Lam, R. S. et J. D. Ullman, *Compilers : Principles, techniques, and tools*, Addison-Wesley, 1986.

[3] *Sun applet resources*, `http://java.sun.com/applets/`, Accédé : 2012/12/10.

[4] Aversano, L., G. Canfora, A. Cimitile et A. De Lucia, « Migrating legacy systems to the web : an experience report », *Processing of the Fifth IEEE Conference on Software Maintenance and Reengineering (CSMR)*, p. 148–157, 2001.

[5] Bennett, K., « Legacy systems : Coping with success », *IEEE Software*, p. 19–23, 1995.

[6] Bisbal, J., D. Lawless, B. Wu et J. Grimson, « Legacy information system migration : A brief review of problems, solutions and research issues », *IEEE Software*, 16, p. 103–111, 1999a.

[7] Bisbal, J., D. Lawless, B. Wu et J. Grimson, « Legacy information systems : issues and directions », *Software, IEEE*, 16(5), p. 103 –111, sep/oct 1999b.

[8] Bodhuin, T., E. Guardabascio et M. Tortorella, « Migrating cobol systems to the web by using the mvc design pattern », *Proceedings. ninth working conference on reverse engineering, 2002.*, p. 329 – 338, 2002.

[9] Bovenzi, D., G. Canfora et A. R. Fasolino, « Enabling legacy system accessibility by web heterogeneous clients », *Processing of the Fifth IEEE Conference on Software Maintenance and Reengineering (CSMR)*, p. 73–81, 2003.

[10] Brodie, M. L. et M. Stonebraker, *Migrating legacy systems gateways, interfaces & the incremental approach*, Morgan Kaufmann, 1995.

[11] Canfora, G., A. Cimitile, A. De Lucia et G. A. Di Lucca b, « Decomposing legacy programs : a first step towards migrating to client-server platforms », *The Journal of Systems and Software*, 54, p. 99–110, 2000.

[12] *The www common gateway interface version 1.1*, `http://tools.ietf.org/html/draft-robinson-www-interface-00`, Accédé : 2012/12/10.

[13] Chikofsky, E. J. et J. H. C. II, « Reverse engineer and design recovery : A taxonomy », *IEEE Software*, 7, p. 13–17, 1990.

[14] *Cics transaction gateway.*, `http://www-01.ibm.com/software/htp/cics/ctg/`, Accédé : 2012/12/10.

[15] *Cics/esa application programming reference release 3*, `ftp://ftp.software.ibm.com/software/cics/pdf/dfhp400.pdf`, Accédé : 2012/12/10.

[16] Comella-Dorda, S., K. Wallnau, R. C. Seacord et J. Robert, « A survey of legacy system modernization approaches », *CMU/SEI-2000-TN-003*, 2000.

[17] Deursen, v., P. Klint et C. Verhoef, « Research issues in software renovation », *Proceedings of the Fundamental Approaches to Software Engineering, FASE99.*, p. 1–23, 1999.

[18] Di Lucca, G. A., A. R. Fasolino, P. Guerra et S. Petruzzelli, « Migrating legacy systems towards object-oriented platforms », *ICSM '97 processing of the International Conference on Software Maintenance*, p. 122–129, 1997.

[19] *Display processing system (dps 2200) form design programming guide*, `http://public.support.unisys.com/2200/docs/cp11.2/pdf/78312279-004.pdf`, Accédé : 2012/12/10.

[20] El-Ramly, M., P. Iglinski, E. Stroulia, P. Sorenson et B. Matichuk, « Modeling the system-user dialog using interaction traces », *in Proceedings Of the 8th Working Conference on Reverse Engineering (WCRE)*, 2001.

[21] El-Ramly, M., E. Stroulia et P. Sorenson, « Interaction-pattern mining : Extraction usage scenarios from run-time behavior traces », *In Proceedings of the 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2002a.

[22] El-Ramly, M., E. Stroulia et P. Sorenson, « Recovering software requirements from system-user interaction traces », *Proceedings of the 14th international conference on software engineering and knowledge engineering*, SEKE '02, p. 447–454, New York, NY, USA, ACM, 2002b.

[23] El-Ramly, M. et E. Stroulia, « Web-enabling legacy systems via presentation access : From webulation to automation », *ICENCO'2004, 1st Int. Computer Engineering Conference New Technologies for the Information Society*, 2004.

[24] El-Ramly, M., *Reverse engineering legacy user interfaces using interaction traces*, thèse de doctorat, Université d'Alberta, 2003.

[25] Elwahidi, A. et E. Merlo, « Generating user interfaces from specifications produced by a reverse engineering process », *Proceedings., seventh international workshop on computer-aided software engineering, 1995.*, p. 296 –302, juillet 1995.

[26] Englander, R., *Developing java beans*, O'Reilly Media, 1997.

[27] Ferrante, J., K. J. Ottenstein et J. D. Warren, « The program dependence graph and its use in optimization », *ACM Trans. Program. Lang. Syst.*, 9(3), p. 319–349, juillet 1987.

[28] Gagné, P.-Y., *Reverse engineering of user interfaces : Inference of structural and behevioural specifications*, mémoire de maîtrise, McGill University, 1994.

[29] Garrett, J., *Ajax : A new approach to web applications*, 2005.

[30] Honkala, M., *Web user interaction - a declarative approach based on xforms*, thèse de doctorat, Université technologique de Helsinki, 2007.

[31] Hoque, R., *Corba 3*, IDG Books Worldwide, Inc., 1998.

[32] *Aoc/mvs ims automation programmer's reference and installation guide*, IBM, Kingston, New York, USA, 1994.

[33] Jatich, A., *Cics command level programming*, Wiley, 1985.

[34] *Java server pages technology*, `http://www.oracle.com/technetwork/java/javaee/jsp/index.html`, Accédé : 2012/12/10.

[35] Kapoor, R. V., *Device-retargetable user interface reengineering using xml.*, mémoire de maîtrise, University of Alberta. Dept. of Computing Science., 2001.

[36] Kong, L., E. Stroulia et B. Matichuk, « Legacy interface migration : A task-centered approach », *Proc. of 8 th int. conf. on human-computer interaction hci international ?99 (munich*, p. 22–27, 1999.

[37] Larson, J. A., *Tools for building interactive user interfaces*, Prentice Hall, 1992.

[38] Lientz, B. P., *Software maintenance management : A study of the maintenance of computer application software in 487 data processing organizations*, Addison Wesley.

[39] luc Hainaut, J., M. Chandelon, M. Ch, C. Tonneau, M. Joris, J. l. Hainaut, M. Ch, C. Tonneau et M. Joris, « Contribution to a theory of database reverse engineering », *in proc. of the ieee working conf. on reverse engineering*, p. 161–170, IEEE Computer Society Press, 1993.

[40] Martin, R. J. et W. M. Osborne, *Guidance on software maintenance*, National Bureau of Standards : NBS special publication 500-106.

[41] Merlo, E., J. Girard, K. . Kontogiannis, P. Panangaden et R. De Mori, « Reverse engineering of user interfaces », *Proceedings of the Working Conference. on Reverse Engineering, Baltimore*, 1993a.

[42] Merlo, E., J. Girard, L. Hendren et R. De Mori, « Multi-valued constant propagation for the reengineering of user interfaces », *Proceedings., conference on software maintenance ,1993. csm-93*, p. 120 –129, sep 1993b.

[43] Merlo, E., P.-Y. Gagne et A. Thiboutot, « Inference of graphical auidl specifications for the reverse engineering of user interfaces », *Proceedings., international conference on software maintenance, 1994.*, p. 80 –88, sep 1994.

[44] Merlo, E., P.-Y. Gagné, J.-F. Girard, K. Kontogiannis, L. Hendren, P. Panangaden et R. De Mori, « Reengineering user interfaces », *IEEE software 12, 1 (January 1995)*, p. 64–73, 1995.

[45] Milner, R., *Communication and concurrency*, Prentice-Hall, New York, 1989.

[46] Moore, M. M. et L. Moshkina, « Migrating legacy user interfaces to the internet : Shifting dialogue initiative », *Proceedings of the seventh working conference on reverse engineering (wcre'00)*, WCRE '00, p. 52–, Washington, DC, USA, IEEE Computer Society, 2000.

[47] Moore, M. et S. Rugaber, « Using knowledge representation to understand interactive systems », *Proceedings of the 5th international workshop on program comprehension (wpc '97)*, WPC '97, p. 60–, Washington, DC, USA, IEEE Computer Society, 1997a.

[48] Moore, M. M. et S. Rugaber, « Domain analysis for transformational reuse », *Proceedings of the fourth working conference on reverse engineering, 1997.*, p. 156 –163, oct 1997b.

[49] Moore, M. M., « Rule-based detection for reverse engineering user interfaces », *Proceedings of the third working conference on reverse engineering, 1996.*, p. 42 –48, nov 1996.

[50] Moore, M. M., *User interface reengineering*, thèse de doctorat, College of Computing Georgia Institute of Technologiy, 1998.

[51] *Advancing open standards for the information society*, `https://www.oasis-open.org/`, Accédé : 2012/12/10.

[52] Phanouriou, C. et M. Abrams, « Transforming command-line driven systems to web applications », *Comput. Netw. ISDN Syst.*, 29(8-13), p. 1497–1505, septembre 1997.

[53] *Refine dialect*, `http://tams-www.informatik.uni-hamburg.de/vhdl/tools/vom/REFINE-INTRO`, Accédé : 2012/12/10.

[54] *Renaissance project*, `http://www.comp.lancs.ac.uk/projects/renaissance/RenaissanceWeb/`, 1999, Accédé : 2012/12/10.

[55] Resnick Lori Alperin, a., *Classic description and reference manual for the common lisp implementation version 2.1*, AT & T Bell Labs, Murray Hill, 1994.

[56] Sebesta, R. W., *Programming the world wide web*, Addison-Wesley, 2011.

[57] Sellink, A., C. Verhoef et H. Sneed, « Restructuring of cobol/cics legacy systems », *Proceedings of the third european conference on software maintenance and reengineering*, CSMR '99, p. 72–, Washington, DC, USA, IEEE Computer Society, 1999.

[58] Seng, J.-l. et W. Tsai, « A structure transformation approach for legacy information systems- a cash receipts/reimbursement example », *Proceedings on the 32nd Hawaii International Conference on System Sciences*, 1999.

[59] *The java servlet technology*, `http://java.sun.com/servlet/`, Accédé : 2012/12/10.

[60] Shneiderman, B., « Multiparty grammars and related features for defining interactive systems », *IEEE Systems, Man, and Cybernetics*, 1982.

[61] Shneiderman, B., *Designing the user interface : Strategies for effective human-computer interaction*, Addison-Wesley, 1992.

[62] Sneed, H., « Migration of procedurally oriented cobol programs in an object-oriented architecture », *Software maintenance, 1992. proceerdings., conference on*, p. 105 –116, nov 1992.

[63] Sneed, H. M., « Encapsulating legacy software for use in client/server systems », *Proceedings of the 3rd working conference on reverse engineering (wcre '96)*, WCRE '96, p. 104–, Washington, DC, USA, IEEE Computer Society, 1996a.

[64] Sneed, H. M., « Object-oriented cobol recycling », *Processing of WCRE '96*, 1996b.

[65] Sneed, H., « Risks involved in reengineering projects », *Proceedings. sixth working conference on reverse engineering, 1999.*, p. 204 –211, oct 1999.

[66] Sneed, H. M., « Accessing legacy mainframe applications via the internet », *Processing of 2nd International Workshop on Web Site Evolution*, p. 34–46, 2000.

[67] Sneed, H. M., « Wrapping legacy cobol programs behind an xml interface », *Proc. of 8th WCRE-2001, IEEE Computer Siciety Press*, 2001.

[68] Sneed, H. M., « Migrating to web services a research framework », *1st International Workshop on Service-Oriented Architecture Maintenance (SOAM'07)*, 2007.

[69] Sneed, H., « 20 years of software-reengineering : A résumé », *10th Workshop Software Reengineering*, p. 115–124, 2008.

[70] Sneed, H., « Migrating from cobol to java », *Software maintenance (icsm), 2010 ieee international conference on*, p. 1 –7, sept. 2010.

[71] *Soare*, `https://lilab.isys.ucl.ac.be/groups/lilab/wiki/c3829/Soare.html`, Accédé : 2012/12/10.

[72] Stroulia, E., M. El-Ramly, P. Sorenson et R. Penner, « Legacy systems migration in cellest », *Proceedings of the 2000 international conference on software engineering, 2000.*, p. 790, 2000.

[73] Stroulia, E., M. El-Ramly et P. Sorenson, « From legacy to web through interaction modeling », *Proceedings. international conference on software maintenance, 2002.*, p. 320 – 329, 2002.

[74] Stroulia, E., M. El-Ramly, P. Iglinski et P. Sorenson, « User interface reverse engineering in support of interface migration to the web », *Automated Software Engg.*, 10(3), p. 271–301, juillet 2003.

[75] Stroulia, E. et R. Kapoor, « Reverse engineering interaction plans for legacy interface migration. », *Proceedings of the fourth international conference on computer-aided design of user interfaces*, p. 15–17, Valenciennes, France, IEEE Computer Society, 2002.

[76] Tan, Y. S., D. B. Lindquist, T. O. Rowe et J. R. Hind, « Ibm enetwork host on-demand : The beginning of a new era for accessing host information in a web environment », *IBM Systems Journal*, 37(1), p. 133 –151, 1998.

[77] *World wide web consortium*, `http://www.w3.org/`, Accédé : 2012/12/10.

[78] Weiser, M., « Program slicing », *Proceedings of the 5th international conference on software engineering*, ICSE '81, p. 439–449, Piscataway, NJ, USA, IEEE Press, 1981.

[79] Wilde, E., *Wilde's www : Technical foundation of the world wilde web*, Springer, 1999.

[80] *Extensible markup language*, `http://www.w3.org/XML/`, Accédé : 2012/12/10.

[81] *Extensible stylesheet language family*, `http://www.w3.org/Style/XSL/`, Accédé : 2012/12/10.

[82] Zou, Y. et K. Kontogiannis, « Enabling technologies for web-based legacy system integration », *In Proc. of the International Workshop on Web Site Evolution*, 1999.

[83] Zou, Y. et K. Kontogiannis, « Web-based legacy system migration and integration », *the 4th International Conference in Systematic, Cybernetics and Informatics (SCI)*, 2000.

[84] Zou, Y. et K. Kontogiannis, « Migration to object oriented platforms : a state transformation approach », *Proceedings. international conference on software maintenance, 2002.*, p. 530 – 539, 2002.

[85] Zou, Y. et K. Kostas, « Reengineering legacy systems towards web environments », *Dagstuhl Seminar Proceeding 04101*, 2005.