Vers une Evaluation Automatique et Transparente de l'Adéquation Homme-Machine

Gaëlle Calvary[12] *Joëlle Coutaz*[2] *Jean-Luc Voirin*[1]

THOMSON-CSF Radars et Contre-Mesures[1]
10, avenue de la 1ère DFL
29283 BREST cedex, France
{calvary, voirin}@rcm.thomson.fr

CLIPS-IMAG[2]
BP 53
38041 GRENOBLE cedex 9, France
Joelle.Coutaz@imag.fr

RÉSUMÉ

En l'absence d'outils dédiés à une évaluation expérimentale de l'adéquation homme-machine, nous proposons CatchIt, un environnement de développement basé sur une modélisation des connaissances domaine : sa finalité consiste à détecter et expliquer tout problème potentiel d'utilisabilité, observé sur un expert du domaine placé en conditions opérationnelles. L'expérience nous montre en effet que l'évaluation logicielle reste trop souvent fonctionnelle : CatchIt se propose de l'enrichir d'un point de vue opérationnel par la capitalisation et l'activation de connaissances métier pragmatiques.

L'originalité de CatchIt réside dans le caractère exécutable de ces connaissances : regroupées au sein d'un modèle normatif implémenté, elles sont connectées de façon déclarative à l'application. Les évaluations prédictive et expérimentale sont alors menées de façon automatique et transparente. L'instrumentation du code est notamment assumée par CatchIt, conformément aux liens tissés entre l'application et ce modèle. Ainsi, support à l'évaluation de l'utilisabilité, garant d'une meilleure traçabilité, cadre de réflexion à la conception de nouvelles applications, candidat à la réutilisation, le modèle normatif augmente l'application de sa sémantique opérationnelle.

MOTS CLÉS : Adéquation Homme-Machine, évaluation, automatisation, transparence, critique, explication, correction, utilisabilité, réutilisabilité, traçabilité, approches déclaratives.

INTRODUCTION

Bien que l'utilisabilité soit aujourd'hui reconnue comme facteur à part entière en matière de qualité logicielle [26], elle reste souvent sacrifiée. Outre le manque de méthodes et d'outils, c'est aussi le contexte de développement qui explique ce paradoxe. Typiquement sous-estimés, non seulement par manque d'habitude mais aussi pour des raisons de concurrence, les tests d'utilisabilité se déroulent souvent en fin d'affaire, dans des conditions de coûts et de délais tendus.

Avec les avancées de la technologie de l'information, les ordinateurs envahissent aujourd'hui le secteur de la défense [16]. Pour ces applications stratégiques, notamment celles développées à THOMSON-CSF Radars et Contre-Mesures, l'adéquation Homme-Machine, loin d'être un luxe, est une réelle nécessité [15]. Susceptible de mettre en péril des vies humaines, l'interaction se doit d'être la plus naturelle, la plus cognitivement transparente possible.

Dans une démarche exploratoire, c'est aujourd'hui le nombre d'itérations nécessaires à la finalisation du produit qui éponge l'artisanat en la matière. La complexité croissante des systèmes pilotés [3], la concurrence de plus en plus pressante ne peuvent plus se contenter désormais d'approches aussi pragmatiques et peu formalisées.

Après une description de nos motivations, objectifs applicatifs et scientifiques, un état de l'art dans le domaine révèle l'absence d'outil idoine. Nous proposons alors CatchIt, a Critic-based Automatic and Transparent tool for Computer Human Interaction Testing. Après en avoir décrit la couverture fonctionnelle, nous zoomons sur sa facette évaluation. Nous détaillons ses principes, architecture et implémentation puis cernons ses limites. Nous précisons alors les perspectives dans le domaine et mentionnons sa richesse applicative. Notons que par les termes « application » et « outil », nous désignons respectivement l'application sujette à évaluation et l'outil utilisé à ces fins.

MOTIVATIONS

L'évaluation de l'adéquation homme-machine est une tâche intrinsèquement délicate puisque partiellement subjective. Selon le contexte mais aussi la culture, les habitudes, les préférences, les exigences des différents opérateurs, une même application peut être appréciée ou sanctionnée.

L'expérience montre que le pivot d'une utilisabilité maîtrisée gravite autour des connaissances sur le domaine. Influentes en phases de spécification et conception système, elles s'avèrent indispensables à une compréhension fine du comportement opérateur. La faille aujourd'hui identifiée dans nos procesus de conception s'exprime en termes de traçabilité. Du manque de formalisation en la matière, découle une évaluation de l'interaction bien plus fonctionnelle qu'opérationnelle : l'argumentaire déployé en phases amont est souvent mal connu des équipes de développement et il leur est alors difficile

d'appréhender, de façon significative et efficace, l'interaction proposée.

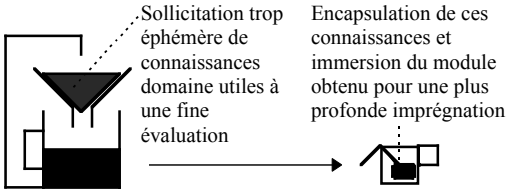

Figure 1 : De la cafetière à l'infusion pour une exploitation durable de connaissances, souvent invoquées lors de la conception, mais qui, par manque de formalisation, échappent notamment à l'évaluation.

Aussi, dans une recherche de compétitivité, spéculons nous sur une modélisation informatique, réutilisable de ces connaissances métier. La tendance industrielle à la spécialisation par domaines, la forte durée de vie de nos applications et la complexité croissante de nos systèmes nous confortent dans cette démarche. Notons que cette formalisation pourrait en outre servir de cadre de réflexion à la conception de nouvelles applications (Figure 2).

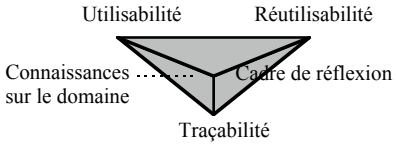

Figure 2 : Quatre arguments pour la modélisation des connaissances métier.

Focalisons notre étude sur la facette évaluation et exprimons, en ces termes, nos objectifs applicatifs.

OBJECTIFS APPLICATIFS

Le postulat à la base de notre démarche suppose que l'utilisation en contexte opérationnel d'un système à travers son IHM, par un opérateur expert du domaine, permet de détecter les insuffisances ou inadéquations de l'interaction proposée. Autrement dit, nous misons sur la modélisation du comportement d'un expert du domaine pour détecter toute irrégularité de comportement par rapport à ce schéma nominal normatif, prescriptif.

L'expérience montre que, sans outil, l'approche est vaine. Le dépouillement des enregistrements s'avère long et fastidieux [1] et se heurte à l'impérative nécessité d'une fine connaissance du domaine. Vouées à l'échec, ces approches manuelles ne sont pas viables pour des applications complexes. Aussi envisageons-nous une automatisation tant pour l'évaluation proprement dite que pour la mise en œuvre de l'outil. Désignons par le terme de transparence cette commodité de mise en œuvre : elle sanctionne toute tâche manuelle d'instrumentation et autres venant polluer et, à terme condamner, la mise en œuvre de l'outil (Figure 3).

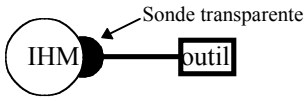

Figure 3 : Concept de transparence pour une médicalisation de l'évaluation : tel un électro-cardiogramme, l'outil sonde le système sans aucune modification manuelle de son code.

Nous rejoignons ainsi [18] qui dénonce la trop fréquente difficulté rédhibitoire de mise en œuvre des outils proposés. Aussi, inspectons nous les cinq dimensions de l'espace problème [7] avec ce souci constant de minimiser les exigences portant sur l'exploitation de l'outil.

Dimension résultats

Par analogie avec la procédure humaine mise en œuvre dans une démarche d'évaluation, six degrés ou lieux d'automatisation apparaissent (Figure 4).

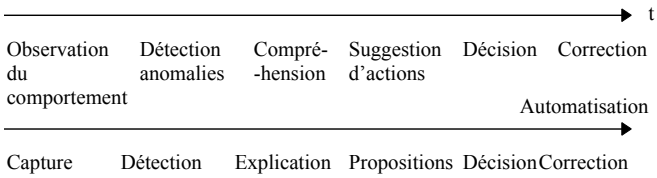

Figure 4 : Six niveaux d'automatisation conformément au processus humain de résolution.

L'expérience montre qu'une simple observation de l'interaction homme-machine, même munie de points de mesure, ne permet pas aisément de rendre compte des incohérences ou inadaptations de l'interaction et encore moins des actions correctrices à mener [5] [2] [1]. L'explication est donc le niveau minimum d'automatisation à atteindre. Ceci signifie que le processus d'abstraction des données, du niveau physique au niveau domaine, est délégué à l'outil.

Dimension moyens humains

L'outil est étalonné par rapport au comportement d'un expert du domaine. Cet expert devra par ailleurs être familier de l'interface pour que soit évaluée, non pas son exploration, mais une utilisation réaliste [35].

Les difficultés de mise en œuvre liées à la disponibilité et au coût de prestation des personnes sollicitées incitent à une prévalidation prédictive . Une détection préliminaire automatisée de problèmes potentiels d'utilisabilité, par l 'application d'heuristiques extraites de disciplines aussi complémentaires que l'ergonomie, les sciences cognitives, la sociologie, etc., doit permettre, dans un premier temps, de résoudre des problèmes d'utilisabilité génériques, décorrélés du contexte opérationnel.

Dimension facteurs situationnels

[32] relate l'influence du contexte sur le comportement opérateur. La combinaison de configurations singulières, liées par exemple à des conditions extrêmes de stress, fatigue, pannes

d'équipements, etc., peut perturber l'opérateur dans son accomplissement de tâche. Une simulation réaliste du contexte opérationnel s'avère nécessaire pour une évaluation objective des distances sémantiques et articulatoires [30].

Dimension connaissances

Les connaissances se déclinent en «Compétences » et «Descriptions ». Les compétences ont trait au coût cognitif d'accès à la méthode d'évaluation. Il est primordial de les limiter. Les descriptions regroupent, quant à elles, les données d'entrée nécessaires au bon fonctionnement de la méthode : modèles de l'utilisateur, de la tâche, spécifications externes de l'interface, scénarios, etc.. Un large spectre d'informations en entrée augmente certes le coût de mise en œuvre de la méthode, mais aussi la robustesse et la concision des résultats [7]. La criticité de nos applications étant à l'origine de notre démarche, nous préférons compenser l'éventuel surcoût par une meilleure réutilisation de ces modèles.

Dimension ressources matérielles

Pour une facilité de mise en œuvre, nous excluons toute dépendance en ressources matérielles.

Synthèse

En résumé, rappelons que l'outil recherché s'adresse à des applications relevant d'un domaine identifié, pour lequel existent des connaissances expertes : conduite automobile, diagnostic médical ou guerre électronique par exemple. Il s'agit ici de structurer ces informations en une base de connaissances informatisée, réutilisable.

Deux types d'évaluation sont alors attendus : une approche prédictive pour la localisation d'anomalies potentielles ; une version expérimentale pour valider le système sur un expert du domaine placé en situation opérationnelle. Ce sont les écarts par rapport au schéma nominal, prescrit par les connaissances métier, que nous recherchons. Pour une meilleure utilisabilité de l'outil, sa mise en œuvre doit être automatique et transparente et toute critique doit pouvoir être justifiée par des explications.

OBJECTIFS SCIENTIFIQUES

Dans l'hypothèse où l'état de l'art ne révéle ni outils ni approches compatibles de nos objectifs applicatifs, il nous faut valider les concepts à la base de notre approche : la faisabilité technique d'une modélisation des connaissances métier, leur exploitation par des mécanismes d'analyse et de confrontation, la génération d'explications et enfin, une mise en œuvre automatique et transparente de ces mécanismes. Il conviendra alors d'étudier la structure d'accueil permettant la structuration, l'édition, la réutilisation des modèles, voire la capitalisation des écarts pour une explication centrée facteurs humains.

ETAT DE L'ART

En l'absence d'outils commercialisés dédiés à l'évaluation de l'interaction homme-machine, examinons les propositions qui émanent de la recherche. Deux grands courants apparaissent selon la finalité de l'approche : construire ou évaluer l'interface.

En matière de construction, des outils tels Mastermind [39] valident la puissance d'expression des modèles. Apparu voici dix ans, ce paradigme du « model-based user interface design » se réfère à une description explicite, largement déclarative, capturant la sémantique de l'application et toute connaissance nécessaire à la spécification tant de l'apparence que du comportement du système interactif [36]. Le but de ces approches consiste à identifier des composants réutilisables, rassembler le plus de connaissances possible au sein de modèles, pour réduire la quantité de code procédural nécessaire à une nouvelle application [36]. UIDE [13], HUMANOID [38], ADEPT [19], MECANO [31], FUSE [24], AME [27], etc., témoignent, comme Mastermind, de la faisabilité technique d'une telle modélisation.

En matière d'évaluation, tandis que de nombreuses méthodes existent, peu d'outils émergent. Nous passons ici sous silence toutes démarches manuelles. En matière d'outils, deux courants s'affrontent selon la philosophie de l'approche. Tandis que certains systèmes évaluent la conception de l'interface, d'autres jugent son utilisation. Ces derniers tels OPADE [8], FRAMER [22], CRACK [11], JANUS-CRACK [12], etc., ne cadrent pas avec nos objectifs. Parmi les outils dédiés à l'évaluation de la conception, certains reposent sur des techniques prédictives, d'autres sur des approches expérimentales. Parmi ces premières, mentionnons GLEAN [20] une version informatisée de GOMS [6], KRI [25], SYNOP [21], CritiGUI [29], AIDE [34], USAGE [4], ERGOVAL [23] [10] et CHIMES (référencé dans [9]). En version expérimentale, une nuance apparaît selon le degré d'automatisation atteint. Playback [28] et le système d'Hammontree [17] ciblent le niveau capture ; MRP [35] et EMA [2] l'analyse ; VDDE [37] l'explication appliquée à la téléphonie.

Cette revue appelle deux remarques : d'une part, la couverture fonctionnelle très ciblée des outils proposés (conception, évaluation prédictive ou expérimentale) ; d'autre part l'absence d'analyseurs génériques permettant une critique à haut niveau d'abstraction de l'interaction proposée. CatchIt, notre contribution au domaine, vise à combler ces lacunes.

CONTRIBUTION : CatchIt

Comme support à la description de CatchIt, considérons un exemple typique de notre métier. Extrait du domaine de la guerre électronique mais largement simplifié pour des raisons de clarté et de confidentialité, c'est le poste de l'opérateur tactique que nous envisageons. A la lumière de cet exemple, nous précisons la couverture fonctionnelle de CatchIt puis zoomons sur l'évaluation. Nous mentionnons enfin ses limites puis les perspectives dans le domaine.

Exemple

L'opérateur tactique est chargé de surveiller la situation environnante. Sur un écran, lui sont positionnés graphiquement les différents bâtiments (bâteaux, etc.) détectés par les équipements embarqués. Une double sélection de ces icônes lui permet d'afficher, dans un espace dédié, des informations complémentaires sur le bâtiment considéré. Ces informations synthétiques le guident dans son choix de réaction : différentes tactiques lui sont permises selon les équipements embarqués. L'opérateur peut consulter un système expert pour l'aider dans sa décision. Quelle que soit la tactique retenue, il devra configurer l'équipement distant chargé de la mettre en œuvre, ordonner, et éventuellement confirmer, le déploiement de cette tactique. Ainsi, la sûreté de l'équipage dépend-elle, non seulement, de la perception de la situation tactique, mais aussi des moyens de réaction disponibles.

Couverture fonctionnelle

Basé sur une modélisation des métier, CatchIt travaille sur des modèles exécutables, dits « modèles normatifs » (Figure 5).

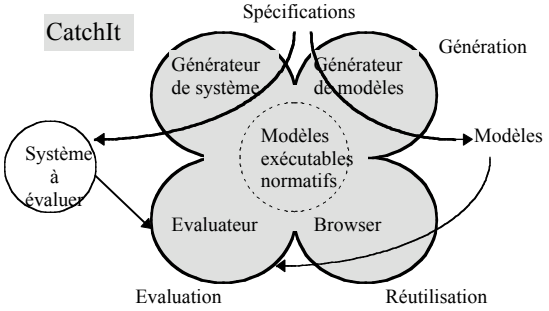

Figure 5 : Les quatre facettes fonctionnelles de CatchIt : génération, évaluation, réutilisation.

- La génération de modèles est une production automatique des modèles normatifs à partir de spécifications textuelles. Ces modèles sont une description informatisée, d'une part, des connaissances sur le domaine et, d'autre part, de la mission opérateur. Parmi les huit types de connaissances prévues (objets, tâches, sessions, rôles, opérateurs, équipements, ressources, environnement), seuls objets et tâches sont aujourd'hui implémentés.
- La génération de systèmes correspond, tel ADEPT, à une génération automatique du système à partir de ses spécifications.
- Le browser s'inscrit dans le cadre de la réutilisation : il permet un parcours et une édition des différents modèles.
- L'évaluateur permet, quant à lui, deux types de validation selon que l'on s'intéresse aux intentions du développeur ou de l'opérateur. C'est sur ce dernier composant, le seul implémenté, que nous nous focalisons ici.

Principe de l'évaluation

La clé de voûte de l'évaluation est un adaptateur qui fait collaborer application et modèles normatifs (Figure 6). Support aux évaluations prédictive et expérimentale, l'adaptateur mémorise, de façon déclarative, les liens sémantiques entre l'application et les modèles. Ces liens sont définis par le développeur et c'est la seule tâche qui lui incombe.

Tout événement impactant un de ces liens est répercuté, de façon transparente, sur les modèles normatifs. Ceux-ci sont alors automatiquement mis à jour, concrétisant ainsi contexte et trace d'interaction. Le contexte mémorise l'évolution des objets au cours du temps : leur patrimoine est enrichi d'un vécu (déploiement de telle tactique sur tel bâtiment). La trace d'interaction, partiellement implémentée aujourd'hui, conserve l'enchaînement des actions opérateurs.

L'intérêt de ces contexte et trace d'interaction réside dans la confrontation désormais possible entre l'intention de l'opérateur et sa mise en œuvre. En effet, tout événement, d'origine externe ou opérateur, est traité par l'application. Il se répercute par d'éventuels actions ou changements d'état. Il s'agit de vérifier leur compatibilité avec les actions ou états prescrits par les modèles normatifs : propagé vers ces modèles, l'événement provoque, via l'adaptateur, la mise à jour des contexte et trace d'interaction. L'intention probable de l'opérateur étant ainsi déterminée, ils identifient les états ou actions souhaitables. Ce sont ces états ou actions qu'il convient de confronter aux états ou actions mis en œuvre côté application (Figure 6).

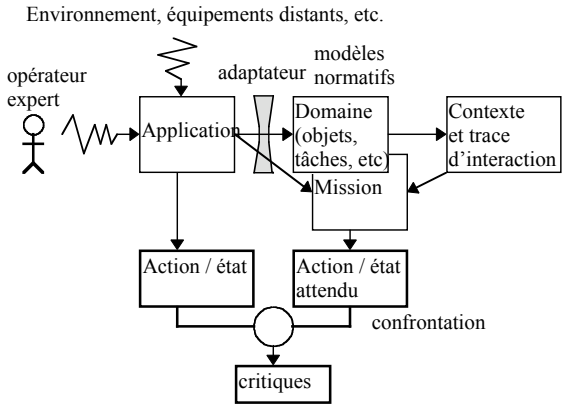

Figure 6 : Principe de l'évaluation expérimentale.

Détaillons les modèles normatifs et l'adaptateur puis décrivons les mécanismes de transparence, détection, explication et correction mis en œuvre.

Modèles normatifs

Au lieu d'enfouir dans l'application les connaissances domaine, au risque alors de les y disséminer, nous les maintenons au sein des modèles normatifs. Au nombre de deux, ces modèles, le modèle du domaine et de la mission, sont exécutables. Ils explicitent respectivement des connaissances générales au domaine traité et une description fine de la mission confiée à l'opérateur. A l'éventuelle redondance qui peut légitimement être invoquée, nous rétorquons réutilisation et surtout l'unité logique et sémantique ainsi recouvrée.

Pour illustrer ces modèles, envisageons une approche orientée objet et adoptons la notation suivante : *Classe* (*Var1*, *Varn*) désigne la classe *Classe* de variables d'instance *Var1*, *Varn*. *Classe* (*Val1*, *Valn*) désigne une instanciation de la classe *Classe* avec les valeurs *Val1*, *Valn*. Introduisons la classe *Tâche* définie conformément au formalisme MAD [33] mais, pour les besoins de l'exposé, réduite à l'expression suivante : *Tache* (*précondition*). Selon ces notations, voici des exemples de modélisation d'objets et de tâches :

- *ObjetTactique* (*position, cap, distance, vitesseEnNds*)
- évaluerCriticité (infosDetailléesDisponibles).

C'est la connexion de ces modèles à l'application qui enrichit cette dernière sémantiquement. Elle est gérée par un composant dédié, « l'adaptateur », qui garantit pureté et transparence.

Adaptateur

La connexion entre l'application et les modèles normatifs est assurée par des liens sémantiques . Mémorisés au sein d'un composant dédié « l'adaptateur », ces liens ne polluent ni l'application, ni ces modèles.

Aujourd'hui implémenté en Smalltalk [14], ce concept de lien présuppose l'évaluation d'une application orientée objet. Les traits sémantiques sont alors exprimés sous forme de classes, variables d'instance, de classe, de métaclasse, etc. Le développeur précise, à la granularité souhaitée, les règles de correspondance entre application et modèles normatifs.

Conformément à l'éternelle dualité tâche/objet, deux types de liens existent selon la nature des entités liées : à un objet est associé un autre objet ; à une action est associée une tâche. La modélisation du comportement opérateur par des règles du type « si *condition* alors *action* » incite à introduire un troisième type de liens : prédicat-méthode. Ces prédicats sont typiquement référencés dans les pré et post conditions des tâches des modèles normatifs (cf *évaluerCriticité)*. La méthode qui leur est associée permet de les évaluer.

- Liens objet-objet : ces liens établissent des relations entre variables pertinentes provenant, d'une part, de l'application à évaluer, et, d'autre part, des modèles normatifs. Une règle de correspondance est prévue pour des associations non symétriques. A ces fins, les classes Variable et Lien sont introduites : *Variable* (*classe*, *variable) ; Lien* (*variable1*, *variable2*, *règle)*, où la règle est un bloc d'instructions. Supposons par exemple que la classe *Bâtiment* de notre application corresponde à la classe *ObjetTactique* du domaine et que leurs variables d'instance respectives *vitesseEnM/S* et *vitesseEnNds* se correspondent moyennant ce changement d'unité. Le développeur définira alors dans l'adaptateur le lien suivant : (*Bâtiment*, *vitesseEnM/S*), (*ObjetTactique*, *vitesseEnNds*), [:rcvr| *vitesseEnNds* := (rcvr *vitesseEnM/S* *3.6)]. Appelons le, par la suite, OO1.
- Liens action-tâche : pour l'instant, seules sont traitées les tâches élémentaires, c'est-à-dire des tâches décomposables en actions physiques [2]. Les liens expriment la correspondance entre les niveaux physique et logique. Trois types d'action peuvent être distingués selon qu'elles s'effectuent sur widgets, objets graphiques ou dispositifs physiques d'interaction : mentionnons respectivement les associations entre le bouton *Tir* et la tâche *Déployer tactique* ; l'icône *Bâtiment* et la tactique opératoire *sélectionner Bâtiment* ; le bouton droit de la souris et la tâche *fermer fenêtre courante*.
- Liens prédicat-méthode : outre l'expression des pré et post conditions des tâches, ces liens répondent aussi au besoin d'expression des propriétés d'utilisabilité. Tandis que les prédicats déclinent les critères, les méthodes concrétisent leur vérification au sein de l'application ou des modèles normatifs. Considérons par exemple dans le modèle de la mission le motif comportemental modélisé par la règle R : « si *nouvelObjetTactique* alors *évaluerCriticité* ». Rappelons que la tâche *évaluerCriticité* possède la précondition *infosDétailléesDisponibles*. Deux liens prédicat-méthode PM1, PM2 sont introduits : ils associent respectivement aux prédicats *nouvelObjetTactique* et *infosDétailléesDisponibles* les méthodes *new* de *ObjetTactique* et *infosDétailléesAffichées* de *Appli*, cette dernière étant définie dans l'application. A défaut de méthodes dédiées, des blocs d'instruction peuvent être associés aux prédicats. Notons que ces liens permettent de concilier les approches état/événement et notamment de s'affranchir des traditionnelles difficultés liées aux approches exclusivement basées tâches : informations déjà affichées, informations affichées mais masquées, raccourcis clavier ayant refermé la fenêtre, etc..

Rappelons que ce tissage déclaratif de liens est la seule tâche à la charge du développeur. CatchIt assume alors l'instrumentation du code, et ceci de façon transparente.

Transparence

Tout objet ou méthode référencé dans l'adaptateur est traité pour une propagation automatique et transparente des événements vers les modèles normatifs. Concrètement, toute méthode liée à un prédicat référencé en partie gauche de règle ou accédant en écriture à l'un ou l'autre des traits sémantiques des objets référencés est automatiquement instrumentée et recompilée. L'instrumentation consiste à immiscer dans le code, juste avant le return, une instruction espion qui redirige l'événement vers l'adaptateur. Notons que cette instrumentation n'est jamais perceptible au développeur, même lorsque ce dernier browse les méthodes

recompilées. La détection d'anomalies peut alors prendre place.

Détection

Deux types de vérification sont effectués :

- une première passe statique, l'évaluation prédictive, s'assure des complétude et correction des liens. Elle vérifie notamment que toute donnée pertinente des modèles normatifs possède un correspondant dans l'application.
- la seconde, l'évaluation expérimentale, repose sur la mise à jour dynamique des contexte et trace d'interaction. Pour cela, chaque objet de l'application, au moins référencé dans un lien, possède un proxy géré par l'adaptateur : toute méthode accédant en écriture à l'objet (mise à jour, destuction) est reproduite sur le proxy. La fidélité des contexte et trace d'interaction est ainsi garantie. A chaque événement, le processus de détection peut alors, dans le contexte courant, soit corroborer et affiner l'intention opérateur déjà déduite, soit, au contraire, révéler un risque potentiel d'inadéquation. La finesse du contexte courant permet d'expliquer l'anomalie détectée.

Pour une perception globale des mécanismes mis en œuvre, illustrons CatchIt sur notre exemple.

Illustration

Considérons le motif comportemental modélisé par la règle R déjà introduite. Rappelons que trois liens, OO1, PM1 et PM2, sont tissés. CatchIt instrumente alors automatiquement les méthodes impliquées dans ces liens, à savoir : le *new* des classes *Bâtiment* et *ObjetTactique* conformément à OO1 et PM1.

Supposons l'arrivée d'un nouveau bâtiment. Une instance de la classe *Bâtiment* est alors créée dans l'application via la méthode *new*. Cette méthode étant instrumentée, l'adaptateur répercute l'événement sur le modèle du domaine en exécutant la méthode *new* de la classe *ObjetTactique*. Cette méthode étant elle-même instrumentée pour l'évaluation du prédicat *nouvelObjetTactique*, la règle R est déclenchée. Il convient alors d'évaluer le prédicat *infosDétailléesDisponibles* par exécution de la méthode associée *infosDétailléesAffichées*. Celle-ci retourne un booléen.

Agenda de recherche

Explication et correction ne sont aujourd'hui pas implémentés. La faisabilité technique d'un adaptateur automatique et transparent est prouvée mais son exploitation est loin d'être explorée. Restent à préciser les propriétés visées (atteignabilité, non préemption, multi-fils, présentations multiples, migration de tâches, observabilité, insistance, honnêteté, curabilité, prévisibilité), à leur définir des métriques, à affiner les mécanismes d'analyse et de confrontation et enfin à expérimenter les modes de publication des critiques (à la volée ou en différé).

Concernant l'explication, il reste à valider que nos modèles normatifs répondent et suffisent à la nécessaire structuration des connaissances [40]. Nous comptons les valuer d'un critère de variabilité comme support à la correction. L'expérience montre en effet qu'au fil de l'énumération suivante la probabilité de remise en cause des informations, ou variabilité, augmente : objets et tâches, règles ergonomiques et cognitives, contraintes logiques et opératoires, tactiques de mise en œuvre, objets d'interaction. Nous souhaiterions enfin maintenir un modèle des écarts pour, à terme, proposer des explications centrées facteurs humains.

Concernant les facettes « génération », il convient de trouver un formalisme d'expression des spécifications.

Limites

Deux points faibles peuvent être perçus dans l'approche : premièrement, CatchIt ne s'adresse qu'à des applications orientées objet ; deuxièmement, il est nécessaire de disposer de leur code source. Concernant le premier grief, rappelons que la tendance actuelle consiste à s'orienter vers ce genre de techniques. Quant au second, la programmation par composants, dont émanent Opendoc et OLE, laisse espérer dans un futur proche l'apparition de protocoles permettant à CatchIt et les applications d'interopérer.

Perspectives

Les applications potentielles de cette approche sont nombreuses. Mentionnons la détection des conditions opérationnelles optimales d'utilisation, la maîtrise des exigences et débits d'interaction, l'adaptation voire l'adaptativité, l'extraction d'expertise, l'assistance à l'opérateur (aide contextuelle voire délégation), l'explication tant pour la pédagogie que l'entraînement.

CONCLUSION

En conclusion, rappelons les originalités principales de CatchIt : la capitalisation de connaissances métier au sein de modèles normatifs exécutables ; la connexion déclarative entre l'application à évaluer et ces modèles; l'instrumentation automatique et transparente du code et l'évaluation argumentée de l'utilisabilité. Précisons que CatchIt s'adresse à des applications réalistes, existantes ou en cours de développement. Il nous reste à l'appliquer sur des systèmes existants et à évaluer notamment son apport en matière de rétro-conception. Nous souhaiterions enfin travailler récursivement en soumettant à CatchIt sa propre évaluation.

REMERCIEMENTS

Nous remercions THOMSON-CSF Radars et Contre-Mesures pour le soutien apporté à l'étude.

BIBLIOGRAPHIE

1. Aublet-Cuvelier, L., Carraux, E., Coutaz, J., Nigay, L., Portolan, N., Salber, D., Zanello, M.L. NEIMO, un laboratoire d'utilisabilité numérique : Leçons de l'expérience. *ERGO IA 96*, pp 149-160.

2. Balbo, S. *Evaluation Ergonomique des Interfaces Utilisateur : Un Pas Vers l'Automatisation*. Thèse de l'Université Joseph Fourier de Grenoble, Septembre 1994, 287 p.

3. Bares, M., Pastor, D. Principe d'un moteur d'interaction multimodale pour systèmes embarqués. *Génie Logiciel*, N°40, Juin 1996, 31-38.

4. Byrne, M.D., Wood, S.D., Sukaviriya, P.N., Foley, J.D., Kieras, D.E. Automating Interface Evaluation. *CHI'94*, Boston, Massachusetts, April 24-28, 1994, pp 232-237.

5. Calvar, J.O., Goubier, T. *Auto-Apprentissage et Interfaces Homme-Machine*. Stage ENSTB réalisé à THOMSON-CSF Radars et Contre-Mesures, Brest, 1992.

6. Card, S., Moran, T., Newell, A. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, 1983.

7. Coutaz, J., Balbo, S. Evaluation des interfaces utilisateur : Taxonomie et recommandations. *IHM'94*, 8-9 décembre, 1994, Lille, pp 211-218.

8. De Rosis, F., Cozza, M.T., De Carolis, B., Errore, S., Pizzutilo, S., De Zegher, I. Adaptative Interaction With Knowledge-Based Systems. *AVI'94*, Bari, Italy, June 1-4, 1994.

9. Farenc, C. ERGOVAL : une méthode de structuration des règles ergonomiques permettant l'évaluation automatique d'interfaces graphiques, Thèse de l'Université Toulouse I, 1997, pp 204

10. Farenc, C., Liberati, V., Barthet, M.F. Automatic Ergonomic Evaluation : What are the limits ? *CADUI'96*, J. Vanderdonckt (eds), 1996, pp 159-170

11. Fischer, G., Morch, A. CRACK : A critiquing approach to cooperative kitchen design. *Proceedings of the ACM International Conference on Intelligent Tutoring Systems*, pp 176-185, May 1988.

12. Fischer, G., Lemke, C., Mastaglio, T., Morch, A.I. Using Critics to Empower Users. *CHI'90*, April, 1990, pp 337-347.

13. Foley, J., Kim, W.C., Kovacevic, S., Murray, K. *The User Interface Design Environment.* GWU-IIST-88-04, Department of Electrical Engineering and Computer Science, Scool of Engineering and Applied Science, The George Washington University, Washington, D.C. 20052, January 88.

14. Goldberg, A. *Smalltalk-80, The Interactive Programming Environment*, Addison-Wesley, 1984.

15. Gould, J.D. How to Design Usable Systems, *Handbook of Human-Computer Interaction*. M. Helander ed., Elsevier Science B.V., 1988, pp 757-789.

16. Goodman, S.E. War, Information Technologies, and International Asymmetries. *Communications of the ACM*, December 1996, Volume 39, Number 12, pp 11- 15.

17. Hammontree, M.L., Hendrickson, J.J., Hensley, B.W. Integrated data capture and analysis tools for research and testing on graphical user interfaces. *CHI'92*, Monterey, 3-7 May 1992, pp 431-432.

18. Holyer, A. Methods For Evaluating User Interfaces, *Cognitive Science Research Paper No. 301*, November 10, 1993.

19. Johnson, P., Wilson, S., Markopoulos, P., Pycock, J. ADEPT - Advanced Design Environment for Prototyping with Task Models. *InterCHI'93*, 24-29 April, 1993, pp 56-57.

20. Kieras, D.E., Wood, S.D., Abotel, K., Hornof, A. GLEAN : A Computer-Based Tool for Rapid GOMS Model Usability Evaluation of User Interface Designs. *UIST'95*, Pittsburgh, Pennsylvania, November 14-17, 1995, pp 91-100.

21. Kolski, C. *Contribution à l'ergonomie de conception des interfaces graphiques homme-machine dans les procédés industriels : application au système expert SYNOP*. Thèse de l'Université de Valenciennes et du Hainaut-Cambrésis, 1989.

22. Lemke, A., Fischer, G. A cooperative problem solving system for user interface design. *Proceedings of the Eighth National Conference on Artificial Intelligence*, 1990, pp 479-484.

23. Liberati, V., Farenc, C. Modélisation des connaissances d'ERGOVAL, un outil d'évaluation ergonomique des interfaces. *IHM'95*, pp 193-200

24. Lonczewski, F., Schreiber, S. The FUSE-System : an Integrated User Interface Design Environment, *CADUI'96*, J. Vanderdonckt (eds), 1996, pp 37-56.

25. Löwgren, J. A Knowledge-Based Tool for User Interface Evaluation and its Integration in a UIMS. *Interact'90*, 1990 , pp 395-400.

26. McCall, J. *Factors in Software Quality*. General Electric Ed., 1977.

27. Märtin, C. Software Life Cycle Automation for Interactive Applications : The AME Design Environment. *CADUI'96*, J. Vanderdonckt (eds), 1996, pp 57-73.

28. Neal, A.S., Simons, R.M. Playback : a method for evaluating the usability of software and its documentation. *CHI'83*, December 1983, pp 78-82.

29. Nitsche-Ruhland, D. Zimmermann, G. CritiGUI-Knowledge-based Support fort the Interface Design Process in Smalltalk. *EWHCI'95*, Moscow, Russia, July 1995.

30. Norman, D.A., Draper, S.W. *User Centered System Design*. Lawrence Erlbaum Associates, Publishers, 1986.

31. Puerta, A. The MECANO Project : Comprehensive and Integrated Support for Model-Based Interface Development. *CADUI'96*, J. Vanderdonckt (eds), 1996, pp 19-35.

32. Reason, J. L'erreur humaine. *Le travail humain*, Collection dirigée par Jean-Michel Hoc, Presses universitaires de France, 1993.

33. Scapin, D.L., Pierret-Golbreich, C. MAD : Une méthode analytique de description des tâches. *IHM'89* , Sophia-Antipolis, Mai 1989, pp 131-148.

34. Sears, A. AIDE : A step toward metric-based interface development tools. *UIST'95*, November 14-17, 1995, pp 101-110.

35. Siochi, A.C., Hix, D. A study of computer-supported user interface evaluation using maximal repeating pattern analysis. *CHI'91*, New Orleans, May, 1991, ACM New-York, pp 301-305.

36. Sukaviriya, N., Kovacevic, S., Foley, J.D., Meyers, B.A., Olsen, D.R., Schneider-Hufschmidt, M. Model-Based Interfaces, What are They and Why Should We Care ?. *UIST'94*, November 2-4, 1994, pp 133-135.

37. Summer, T., Bonnardel, N., Kallak, B.H. The Cognitive Ergonomics of Knowledge-Based Design Support Systems. *CHI'97*, 22-27 march 1997, Atlanta, Georgia, pp 83-90.

38. Szekely, P., Luo, P., Neches, R. Facilitating the Exploration of Interface Design Alternatives : The HUMANOID Model of Interface Design. *CHI'92*, May 3-7, 1992, Monterey, California, pp 507-515.

39. Szekely, P., Sukaviriya, P., Castells , P., Muthukumarasamy, J., Salcher, E. Declarative interface models for user interface construction tools: the MASTERMIND approach. *EHCI'95*.

40. Waern, Y., Hägglund, S., Ramberg, R., Rankin, I., Harrius, J. Computational advice and explanations - behavioural and computational aspects. *Interact'95*, pp 203-206.