Contribution à la formalisation des propriétés graphiques des systèmes interactifs pour la validation automatique

Pascal Béger, Valentin Becquet, Sébastien Leriche, Daniel Prun

Contribution à la formalisation des propriétés graphiques des systèmes interactifs pour la validation automatique

Pascal Béger[1], Valentin Becquet[1], Sébastien Leriche[1], et Daniel Prun[1]

[1]ENAC, Université de Toulouse, France

Résumé

Dans la plupart des secteurs, les systèmes d'aujourd'hui sont interactifs et disposent d'interfaces graphiques sophistiquées. A notre connaissance, il existe peu d'études sur la vérification des propriétés spécifiques de la scène graphique des interfaces homme-machine et en particulier sur celles liées à la visibilité des composants graphiques par l'utilisateur humain. Dans ce papier, nous introduisons notre formalisme permettant de spécifier certaines propriétés des composants graphiques, dans l'objectif de réaliser à terme un outillage de validation automatique de ces propriétés. Nous illustrons l'usage de ce formalisme pour des propriétés extraites d'une norme décrivant un instrument critique présent dans les cockpits d'avions commerciaux. Nous avons développé une version complètement fonctionnelle de cet instrument au moyen de Smala, un langage interactif de haut niveau produit par notre équipe. Cela nous permet de montrer la manière dont nous envisageons les opérations de validation formelle automatique de propriétés graphiques de tels systèmes.

1 Introduction

Les systèmes interactifs sont des systèmes informatiques réactifs qui traitent des informations (clics de souris, entrées de données, etc.) provenant de leur environnement (autres systèmes ou humains) et produisent une représentation (notifications sonores, représentations visuelles, etc.) de leur état interne. Ils sont devenus largement répandus dans différents secteurs tels que l'aéronautique, le spatial, le médical ou les applications mobiles. Ces systèmes prennent de plus en plus en compte l'utilisateur humain par le biais de nouvelles interactions et proposent de nouvelles interfaces riches en composants graphiques et en interactions sophistiquées.

Pour valider ou certifier ces systèmes, les outils issus des méthodes formelles ne sont pas toujours adaptés pour prendre en compte les nouvelles interactions et considérations de l'humain.

En analysant les travaux sur les méthodes formelles appliquées aux systèmes interactifs tels que [13, 19, 5], nous avons constaté que les propriétés liées à la scène graphique et en particulier la visibilité des composants graphiques par l'utilisateur humain étaient peu étudiées. Cela peut s'expliquer par le fait qu'historiquement les programmes informatiques ne possédaient pas d'interface graphique (ou que celle-ci restait peu développée) tandis qu'aujourd'hui nous trouvons des systèmes avec des interfaces où la scène graphique est riche et dynamique. De plus, la conception de la scène graphique suit souvent une norme ou des règles de conception imposées [26, 27]. De ce fait, l'étude formelle de ces propriétés reste considérée comme peu pertinente, le processus de validation reposant généralement sur une étude manuelle du système, en suivant des checklists afin d'assurer que pour tout scénario d'utilisation du système, la scène graphique respecte les exigences imposées.

Ce papier suit les travaux que nous avons présentés dans Béger *et al.* [6] afin de contribuer à la certification des systèmes interactifs en utilisant le langage **Smala**[1] et son environnement d'exécution **Djnn**[2]. Nous pensons que la formalisation de ces propriétés graphiques permettra d'automatiser le processus de validation de la scène graphique en fonction des exigences et en particulier celles liées à la visibilité des composants graphiques qui nous serviront d'exemple

1. `http://smala.io`
2. `http://djnn.net`

dans cet article. Bien que ces propriétés graphiques pourraient paraître simples, nous estimons que le gain, en temps et en réduction d'erreurs humaines, apporté par ce processus de validation automatique serait important.

Après avoir réalisé un état de l'art sur les propriétés étudiées pour les systèmes interactifs et sur les variables décrivant les éléments graphiques, nous présentons notre formalisme permettant de définir des propriétés graphiques. Nous introduisons ensuite notre cas d'étude, le système d'alerte de trafic et d'évitement de collision TCAS (*Traffic alert and Collision Avoidance System*), ainsi que les propriétés issues de la norme associée à l'instrument IVSI (*Instantaneous Vertical Speed Indicator*) qui produit la visualisation. Nous illustrons l'usage de notre formalisme en définissant formellement des propriétés graphiques issues de la norme puis, à partir de l'implémentation de l'instrument dans le langage Smala, nous montrons comment nous envisageons la validation automatique de ces propriétés.

2 État de l'art

Historiquement, les premières propriétés étudiées pour les logiciels et systèmes informatiques concernaient la sûreté (e.g. absence d'événement indésirable, bornitude) ainsi que la vivacité des programmes (e.g. retour à un état donné, absence d'interblocage) [23]. Les principales méthodes utilisées pour vérifier et valider formellement des systèmes sont la vérification de modèle par modèle checking [2], par preuve mathématique [4], l'analyse statique [15] ainsi que les processus de tests. Cependant, l'évolution de ces systèmes et l'apparition des IHM (Interfaces Homme-Machine) modernes font émerger de nouvelles propriétés qui challengent ces méthodes.

Une partie de ces nouvelles propriétés concernent le comportement de l'utilisateur : objectifs de l'utilisateur [7], attention de l'utilisateur [25], expérience et apprentissage de l'utilisateur [8]. Cerone et Elbegbayan [9] ont par exemple modélisé le comportement de l'utilisateur au cours des interactions avec une interface web représentant un forum de discussion. Ce modèle de comportement consiste à définir les objectifs de l'utilisateur, par exemple l'envoi de message sur le forum, pour ensuite restreindre les actions possibles de l'utilisateur en implémentant des contraintes dans l'interface par le biais de privilèges en fonction du niveau de l'utilisateur (e.g. connecté ou non).

Nous identifions également la classe des propriétés relatives aux IHM adressant par exemple la latence [20] ou les propriétés CARE [11]. Masci *et al.* [22] ont étudié la prédictabilité de l'IHM d'une pompe à perfusion, i.e. l'utilisateur est conscient de l'état interne du système par le biais des informations données par l'IHM et peut prédire son état interne futur en fonction des saisies de données possibles.

Une autre classe de propriétés concerne la sécurité [18]. Rukšėnas *et al.* [24] ont utilisé une architecture cognitive du comportement de l'utilisateur modélisée en SAL [14] afin d'étudier les risques de failles de sécurité dans le cadre de l'utilisation d'un système d'authentification.

Au niveau de la représentation graphique des données ou de l'état du système, Hjelmslev [16] donne deux plans permettant de définir un élément graphique et l'information qu'il contient. Cette information appartient au plan du contenu, aussi appelé plan des signifiés. L'élément graphique appartient au plan de l'expression ou plan des signifiants. Le signifiant peut être caractérisé par des paramètres associés à des éléments graphiques tels que les variables visuelles définies par Bertin et Barbut [3] (coordonnées, taille, valeur, grain, couleur, orientation et forme) et les attributs esthétiques de Wilkinson [28] (transparence). Jacques Bertin, qui était cartographe, portait son intérêt principalement sur les éléments graphiques placés sur une feuille de papier. De ce fait, les coordonnées étudiées se limitaient au plan (x, y). Dans le cas des systèmes informatiques où les IHM ont un affichage dynamique, il faut également ajouter la variable l correspondant à l'ordre ou couche d'affichage des composants graphiques. La couche d'affichage et le dynamisme de l'affichage induisent des propriétés liées à la superposition.

3 Cas d'étude : TCAS

Le TCAS (*Traffic alert and Collision Avoidance System*) est un système aéronautique ayant pour objectif d'améliorer la sécurité aérienne en prévenant les collisions entre aéronefs (avions, hélicoptères, etc.). Il est obligatoire sur tous les avions commerciaux (transportant plus de 19 passagers ou pesant plus de 5.7 tonnes en Europe). Il est défini par la norme ED-143 [1] et on peut trouver dans Williams [29] des informations détaillées de différents scénarios d'utilisation, la logique de fonctionnement ou les différents affichages du système.

a. IVSI - Compteur de vitesse verticale (pieds$\times$1000)
b. IVSI - Aiguille indiquant la vitesse verticale
c. TA - Position de l'avion (*own aircraft*)
d. TA - Portée de 2 NM autour de *own aircraft*
e. TA - Echelle de l'affichage du trafic environnant
f. TA - Trafic de niveau de menace *threat aircraft*
g. TA - Trafic de niveau de menace *intruder aircraft*
h. TA - Trafic de niveau de menace *proximate aircraft*
i. RA - Vitesse verticale à atteindre
j. RA - Vitesse verticale à éviter

FIGURE 1 – TCAS implémenté sur l'IVSI et réalisé en Smala

Par le biais de calculs effectués sur les informations reçues du trafic environnant, le TCAS fournit deux services aux pilotes : les TAs (*traffic alerts*) et les RAs (*resolution advisories*). Les TAs permettent d'informer les pilotes du type de trafic environnant (*threat aircraft*, *intruder aircraft*, *proximate aircraft*, *other aircraft*) en y associant une symbolique (forme et couleur). Les RAs permettent de notifier les pilotes des manœuvres à effectuer afin d'éviter tout risque de collision (atteindre une vitesse verticale donnée, conserver la vitesse verticale actuelle). Ces deux services sont fournis sous forme de notifications sonores et visuelles, ce qui fait du TCAS un **système critique multimodal**.

Un des éléments du TCAS est l'instrument intégré au cockpit fournissant la visualisation. Il en existe plusieurs. Nous présentons l'IVSI (*Instantaneous Vertical Speed Indicator*, figure 1) qui indique aux pilotes la vitesse verticale actuelle de leur aéronef, c'est-à-dire la vitesse instantanée de montée ou de descente. Cet instrument accueille aussi la visualisation des informations élaborées par le TCAS. Ainsi, un RA est visualisé par l'affichage conjoint d'un arc rouge, indiquant au pilote la plage de vitesses verticales à éviter, et d'un arc vert désignant la vitesse cible qu'il doit atteindre.

Nous avons extrait de la norme ED-143 les exigences relatives à la visualisation du TCAS, applicables à l'IVSI. Ces exigences, représentatives de la complexité de la norme graphique, sont présentées dans le tableau 1. Nous avons mis en évidence (cases sur fond gris) certaines des exigences qui serviront d'exemples pour la formalisation dans la section suivante.

4 Contribution

Notre objectif est de pouvoir vérifier automatiquement que l'implémentation de l'IVSI respecte les éléments issus de la norme ED-143. Pour cela, nous proposons ici une première formalisation des propriétés graphiques issues de cette norme. Nous appliquons ensuite ce formalisme à notre exemple d'instrument IVSI. Nous montrons enfin comment pourra se faire la vérification de ces propriétés à partir du graphe de scène extrait du code Smala, le langage qui nous a servi pour l'implémentation.

Ex.	Information	Forme	Couleur	Position
E_1	Propre position relative (*own aircraft*)	Avion (3 traits)	Blanc ou cyan ($\neq$ couleur *proximate* / *other aircraft*)	Centré horizontalement, 1/3 hauteur affichage
E_2	Portée de 2 NM autour de *own aircraft*	8 traits en cercle	Couleur *own aircraft*	Centré sur *own aircraft*, rayon en fonction de la portée
E_3	TA - Trafic de type *threat aircraft*	Carré plein	Rouge	Zone délimitée par le compteur de vitesse
E_4	TA - Trafic de type *intruder aircraft*	Cercle plein	Jaune ou ambre	Zone délimitée par le compteur de vitesse
E_5	TA - Trafic de type *proximate aircraft*	Losange plein	Blanc ou cyan ($\neq$ couleur *own aircraft*)	Zone délimitée par le compteur de vitesse
E_6	TA - Trafic de type *other aircraft*	Losange vide	Blanc ou cyan ($\neq$ couleur *own aircraft*)	Zone délimitée par le compteur de vitesse
E_7	TA - Altitude relative (en centaine de pieds)	"+" / "-" et deux chiffres	Couleur trafic	Au-dessus / en dessous du symbole du trafic
E_8	TA - Sens vertical	Flèche verticale (haut ou bas)	Couleur trafic	A droite du symbole du trafic
E_9	RA - Vitesse à atteindre	Arc de cercle	Vert	Compteur de vitesse
E_{10}	RA - Vitesse à éviter	Arc de cercle	Rouge	Compteur de vitesse

TABLE 1 – Liste des exigences relatives au graphique du TCAS implémenté sur l'IVSI

4.1 Formalisation des propriétés graphiques

Afin de définir formellement les propriétés liées à la scène graphique, nous considérons les variables visuelles suivantes : **coordonnées**, **couche (l)**, **taille**, **couleur**, **orientation**, **forme** et **transparence**. Il est à noter que nous proposons pour le moment une définition mathématique générale que nous restreindrons dans la suite de nos travaux en fonction des techniques de validation formelle que nous utiliserons.

Les éléments graphiques considérés peuvent être définis complètement à l'aide d'un point spécifique (origine) et d'un ensemble de grandeurs caractérisitiques gc. Par exemple, un rectangle peut être défini à l'aide de son origine (coin supérieur gauche), sa hauteur h et sa largeur w. Les coordonnées x et y représentent des pixels et sont des entiers naturels. La couche d'affichage l est différente pour chaque élément graphique. $l = 0$ indique la couche la plus profonde et donc la première affichée.

Définitions

Afin d'aider à la compréhension des propriétés, voici un ensemble de définitions relatives aux termes et symboles employés. Nous précisons entre des parenthèses les variables visuelles concernées.

- $e_i = \langle (x_0, y_0), l, gc, c, o \rangle$: élément graphique (**coordonnées, couche, taille, forme, couleur, transparence**)
- $(x_0(e_i), y_0(e_i)) \in \mathbb{N} \times \mathbb{N}$: coordonnées de l'origine de e_i (**coordonnées, forme**),
- $l(e_i) \in \mathbb{N}$: ordre ou couche d'affichage de e_i (**couche**) et $(e_1 \neq e_2) \Leftrightarrow (l(e_1) \neq l(e_2))$,
- $c(e_i) = \langle c_r(e_i), c_g(e_i), c_b(e_i) \rangle, c_j(e_i) \in [0, 255]$: couleur de e_i en RGB (**couleur**),
- $o(e_i) \in [0, 100]$: coefficient d'opacité de e_i (**transparence**),
- $\mathcal{D}(e_i)$: domaine de e_i (**coordonnées, forme, taille**).
  Le domaine est calculé grâce aux coordonnées de l'origine et des grandeurs caractéristiques de e_i. Pour un rectangle dont la largeur est parallèle à l'axe des abscisses : $\mathcal{D}(e_i) = \{(x, y) \mid x \in [x_0(e_i), x_0(e_i) + w(e_i)], y \in [y_0(e_i), y_0(e_i) + h(e_i)]\}$

Propriétés

Nous avons identifié un premier ensemble de propriétés de base qui nous serviront à exprimer les propriétés plus complexes relatives à la visibilité des éléments graphiques. Le tableau 2 présente chacune de ces propriétés avec une définition dans notre formalisme et une illustration.

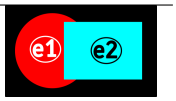
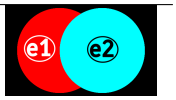
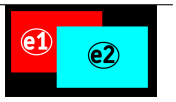
Propriété	Formalisation	Illustration
ordre d'affichage de e_1 inférieur à celui de e_2	$e_1 <_d e_2 \Leftrightarrow l(e_1) < l(e_2)$	
intersection de e_1 et e_2	$e_1 \bigcap_? e_2 \Leftrightarrow (\mathcal{D}(e_1) \cap \mathcal{D}(e_2)) \neq \emptyset$	
égalité des couleurs de e_1 et e_2	$e_1 =_c e_2 \Leftrightarrow c_i(e_1) = c_i(e_2), \forall i \in [r,g,b]$ $e_1 \neq_c e_2 \Leftrightarrow \exists i \in [r,g,b] \mid c_i(e_1) \neq c_i(e_2)$	
superposition de e_1 par e_2	$e_1 <_{sup} e_2 \Leftrightarrow e_1 \bigcap_? e_2 \wedge e_1 <_d e_2$	
masquage (tout ou partie) de e_1 par e_2	$e_1 <_{mask} e_2 \Leftrightarrow e_1 <_{sup} e_2 \wedge o(e_2) > 0$	

TABLE 2 – Formalisation des propriétés graphiques de base

4.2 Application au TCAS

Afin d'utiliser notre formalisme pour exprimer les exigences du TCAS, la notation ta_o représentera l'élément graphique associé à l'information TA *other aircraft*. De manière analogue, nous aurons : ta_p (TA *proximate aircraft*), ta_i (TA *intruder aircraft*), ta_t pour (TA *threat aircraft*) et own (position relative *own aircraft*). Nous utiliserons également vsi pour représenter l'élément graphique correspondant au compteur de vitesse verticale, ext pour le fond hors de la zone délimitée par le compteur de vitesse ainsi que red, $yellow$, $amber$, $cyan$, $white$ représentant respectivement les couleurs rouge, jaune, ambre, cyan et blanc.

Nous illustrons les propriétés d'intersection et d'égalité des couleurs avec les propriétés extraites de la norme du TCAS.

La propriété d'intersection $e_1 \bigcap_? e_2$ nous permet de définir les exigences suivantes du tableau 1 ($E_{i,j}$ désignant la caractéristique de la colonne j pour l'exigence E_i) :

— $E_{3,position} = \neg(ta_t \bigcap_? vsi) \wedge \neg(ta_t \bigcap_? ext)$
— $E_{4,position} = \neg(ta_i \bigcap_? vsi) \wedge \neg(ta_i \bigcap_? ext)$
— $E_{5,position} = \neg(ta_p \bigcap_? vsi) \wedge \neg(ta_p \bigcap_? ext)$
— $E_{6,position} = \neg(ta_o \bigcap_? vsi) \wedge \neg(ta_o \bigcap_? ext)$

L'exigence $E_{3,position}$ est à comprendre comme suit : l'intersection entre ta_t (élément graphique d'un trafic de type *threat aircraft*) et vsi (élément graphique du compteur de vitesse verticale) doit être vide et de même pour l'intersection entre ta_t et la zone hors du compteur de vitesse.

Avec la propriété d'égalité des couleurs $e_1 =_c e_2$ nous définissons ces exigences :

— $E_{1,couleur} = (own =_c white \vee own =_c cyan) \wedge ta_p \neq_c own \wedge ta_o \neq_c own$,
— $E_{3,couleur} = ta_t =_c red$
— $E_{4,couleur} = ta_i =_c yellow \vee ta_i =_c amber$
— $E_{5,couleur} = (ta_p =_c white \vee ta_p =_c cyan) \wedge ta_p \neq_c own$
— $E_{6,couleur} = (ta_o =_c white \vee ta_o =_c cyan) \wedge ta_o \neq_c own$

L'exigence $E_{5,couleur}$ est à comprendre comme suit : la couleur de ta_p (élément graphique d'un trafic de type *proximate aircraft*) doit être blanc ou cyan et ne doit pas être la même que celle de own (élément graphique de la propre position relative).

4.3 Validation dans Smala

Le langage Smala [21], associé à son environnement d'exécution Djnn est notre point d'entrée afin d'étudier concrètement l'expression et la validation formelle des propriétés relatives à la scène graphique des IHM. Des précédents travaux de l'équipe (Chatty *et al.* [10]) ont montré comment exploiter le graphe d'activation que l'on peut produire automatiquement depuis le code des applications Smala. Le graphe d'activation est une structure contenant l'ensemble des composants (graphiques ou non) du programme et qui permet de visualiser la propagation des évènements et l'ordre d'exécution de chaque composant (tous les éléments du graphe d'activation sont ordonnés ce qui détermine l'ordre d'exécution, selon un parcours en profondeur de gauche à droite). En l'état, ce graphe d'activation déduit du code Smala nous permet de valider par exemple la propriété d'accessibilité suivante pour le TCAS : l'événement *"type de trafic == threat_aircraft"* sera toujours suivi de l'événement *"affichage symbolique threat_aircraft"*.

Pour l'étude de la scène graphique des applications Smala, nous réalisons une simplification de ce graphe d'activation afin de n'en garder que les composants graphiques et donc obtenir le graphe de scène de l'application. La sémantique du langage et l'ordre de parcours du graphe d'activation étant conservés, nous pouvons raisonner sur ce graphe de scène comme sur le graphe de d'activation. Dans un premier temps, nous considérons un graphe de scène statique. Nous connaissons donc l'ordre d'affichage des composants graphiques et nous pouvons ainsi prendre en compte la première de nos variables visuelles pour les composants : **couche**. Les instructions graphiques de Smala sont calquées sur la norme SVG (Scalable Vector Graphics), un format de données ASCII conçu pour décrire des ensembles de graphiques vectoriels [3]. Pour chaque objet défini dans la norme, il existe un composant Smala qui le représente. Ainsi, en plus d'une API de dessin simple et consistante pour le programmeur, le chargement d'un fichier SVG permet de réifier chaque élément graphique sous la forme d'un composant Smala. Cela permet de concevoir une partie de l'interface graphique dans un logiciel dédié (Illustrator, Inkscape…) et par la suite de programmer en Smala des interactions avec (et entre) ces différents composants. Nous avons utilisé ce processus dans notre implémentation de l'IVSI, la partie graphique est définie dans un fichier SVG, les interactions sont décrites en Smala. Le graphe de scène contient donc toutes les propriétés propres à la norme SVG et notamment celles en rapport avec nos variables visuelles non encore décrites : **coordonnées**, **taille**, **valeur**, **grain**, **couleur**, **orientation**, **forme** et **transparence**.

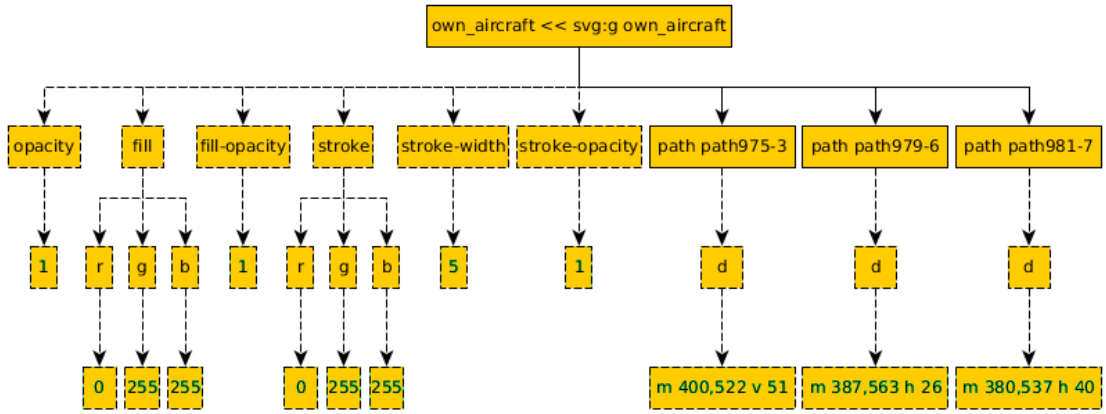


FIGURE 2 – Graphe de scène du composant graphique *own aircraft* du TCAS

La figure 2 présente un exemple de graphe de scène que nous avons réalisé manuellement à partir du code Smala et du fichier SVG pour le composant *own aircraft* qui peut être vu sur la figure 1 (symbole c). Ce graphe nous permet de valider, dans un premier temps par lecture sans outillage, les exigences graphiques suivantes :

3. https://www.w3.org/TR/SVG11

— $E_{1,symbole}$: le composant $own\_aircraft$ est composé de trois *path* (un vertical et deux horizontaux) correspondant aux trois traits représentant l'avion,
— $E_{1,couleur}$: à l'état initial, les valeurs *r*, *g* et *b* de *fill* (remplissage) et *stroke* (contour) sont respectivement à 0, 255 et 255 ce qui correspond à la couleur cyan.

En plus du respect de ces exigences, la sémantique opérationnelle du langage Smala nous permet d'avoir des informations supplémentaires sur l'ordre d'affichage des trois traits. Au cours du processus d'affichage de la scène graphique, le graphe est parcouru en profondeur, de gauche à droite. Ainsi, le composant *path975-3* est affiché avant *path979-6*, lui-même affiché avant *path981-7*. Dans notre formalisme, cela revient à écrire $path975\text{-}3 <_d path979\text{-}6 <_d path981\text{-}7$.

5 Conclusion et perspectives

Les propriétés liées à la scène graphique des IHM sont peu étudiées par les méthodes formelles. Nous pensons que formaliser ces propriétés permettrait d'une part de mieux les étudier (en particulier celles portant sur la visibilité des composants graphiques), et d'autre part de contribuer à l'automatisation du processus de validation des interfaces. À partir de cette problématique, nous avons commencé à formaliser certaines propriétés de base des composants graphiques telles que l'**intersection**, la **superposition**, le **masquage** (tout ou partie) et l'**égalité de couleur**. Nous avons utilisé ces propriétés afin de définir formellement certaines exigences graphiques de la visualisation d'un système critique, le TCAS, et utilisé le graphe de scène issu de l'implémentation en Smala afin d'étudier les liens entre nos propriétés et le graphe.

Pour l'instant, nous générons manuellement les graphes de scène des applications Smala complétés avec les informations nous permettant d'avoir une vue sur les différentes variables visuelles qui nous intéressent. Nous pouvons valider certaines propriétés graphiques à partir de ce graphe et des variables accessibles en faisant le lien avec notre formalisme. Nous travaillons actuellement sur un outil permettant de générer automatiquement ces graphes de scène à partir du code Smala et des fichiers SVG.

De plus, nous ne considérons actuellement que le graphe de scène de manière statique. Nous augmenterons les définitions afin de pouvoir prendre en compte la dynamicité de la scène graphique.

À partir des catégories de modèles formels que nous avons présentées dans Béger *et al.* [6], nous avons réfléchi aux modèles utilisables pour étudier ces propriétés. La logique de Hoare [17] permet de vérifier la véracité d'une assertion donnée après exécution (de tout ou partie) d'un programme, en fonction de pré-conditions. Le plugin WP de Frama-C [12] utilise notamment la logique de Hoare pour vérifier des propriétés par annotation de code. C'est en nous basant sur cette logique que nous prévoyons de vérifier automatiquement les propriétés graphiques par le biais d'annotation du code Smala par analyse statique.

Enfin, afin de couvrir davantage d'exigences graphiques, nous réfléchissons à étendre notre formalisme à de nouvelles propriétés telles que la distance entre deux composants graphiques, la position relative entre deux composants graphiques et la confusion des couleurs résultant d'une proximité sur les composantes.

6 Remerciements

Ce travail est en partie financée par le projet ANR FORMEDICIS, ANR-16-CE25-0007.

Références

[1] Ed 143 - minimum operational performance standards for traffic alert and collision avoidance system ii (tcas ii), April 2013.

[2] B. BERARD, M. BIDOIT, A. FINKEL, F. LAROUSSINIE, A. PETIT, L. PETRUCCI et P. SCHNOEBELEN : *Systems and Software Verification : Model-Checking Techniques and*

Tools. Springer Publishing Company, Incorporated, 1st édn, 2010. ISBN 3642074782, 9783642074783.

[3] J. BERTIN et M. BARBUT : *Sémiologie graphique : les diagrammes, les réseaux, les cartes*. Gauthier Villars, 1973.

[4] S. BOLDO, C. LELAY et G. MELQUIOND : Formalization of Real Analysis : A Survey of Proof Assistants and Libraries. *Mathematical Structures in Computer Science*, 26(7):1196–1233, oct. 2016.

[5] J. BOUCHET, L. MADANI, L. NIGAY, C. ORIAT et I. PARISSIS : Formal testing of multimodal interactive systems. *In* J. GULLIKSEN, M. B. HARNING, P. PALANQUE, G. C. van der VEER et J. WESSON, éds : *Engineering Interactive Systems*, p. 36–52, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-92698-6.

[6] P. BÉGER, S. LERICHE et D. PRUN : Vers la certification de programmes interactifs Djnn. *In Afadl 2018, 17èmes journées Approches Formelles dans l'Assistance au Developpement de Logiciels* , Grenoble, France, juin 2018.

[7] A. CERONE : Closure and attention activation in human automatic behaviour : A framework for the formal analysis of interactive systems. 45, 01 2011.

[8] A. CERONE : Towards a cognitive architecture for the formal analysis of human behaviour and learning. *In Software Technologies : Applications and Foundations*, p. 216–232, Cham, 2018. Springer International Publishing. ISBN 978-3-030-04771-9.

[9] A. CERONE et N. ELBEGBAYAN : Model-checking driven design of interactive systems. *Electronic Notes in Theoretical Computer Science*, 183:3 – 20, 2007. ISSN 1571-0661. Proceedings of the First International Workshop on Formal Methods for Interactive Systems.

[10] S. CHATTY, M. MAGNAUDET et D. PRUN : Verification of properties of interactive components from their executable code. *In Proceedings of the 7th ACM SIGCHI*, EICS '15, p. 276–285, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3646-8.

[11] J. COUTAZ, L. NIGAY, D. SALBER, A. BLANDFORD, J. MAY et R. M. YOUNG : *Four Easy Pieces for Assessing the Usability of Multimodal Interaction : The Care Properties*, p. 115–120. Springer US, Boston, MA, 1995. ISBN 978-1-5041-2896-4.

[12] P. CUOQ, F. KIRCHNER, N. KOSMATOV, V. PREVOSTO, J. SIGNOLES et B. YAKOBOWSKI : Frama-c. *In Software Engineering and Formal Methods*, p. 233–247, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[13] B. D'AUSBOURG : Using model checking for the automatic validation of user interfaces systems. *In* P. MARKOPOULOS et P. JOHNSON, éds : *Design, Specification and Verification of Interactive Systems '98*, p. 242–260, Vienna, 1998. Springer Vienna. ISBN 978-3-7091-3693-5.

[14] L. M. de MOURA, S. OWRE et N. SHANKAR : The sal language manual. 2003.

[15] A. GOSAIN et G. SHARMA : Static analysis : A survey of techniques and tools. *In Intelligent Computing and Applications*, p. 581–591, New Delhi, 2015. Springer India. ISBN 978-81-322-2268-2.

[16] L. HJELMSLEV : *Prolegomena to a theory of language*. Num. 7 de Prolegomena to a Theory of Language. University of Wisconsin Press, 1961.

[17] C. A. R. HOARE : An axiomatic basis for computer programming. *Commun. ACM*, 12 (10):576–580, oct. 1969. ISSN 0001-0782.

[18] C. W. JOHNSON : Using assurance cases and boolean logic driven markov processes to formalise cyber security concerns for safety-critical interaction with global navigation satellite systems. 45, 01 2011.

[19] N. KAMEL et Y. AIT-AMEUR : Modèle formel général pour le traitement d'interactions multimodales. *In IHM 04*, p. 219–222, Namure, Belgique, 03 2004.

[20] S. LERICHE, S. CONVERSY, C. PICARD, D. PRUN et M. MAGNAUDET : Towards Handling Latency in Interactive Software. *In FMIS 2018, 7th International Workshop on Formal Methods for Interactive Systems*, vol. 11176, p. pp 233–239 /ISBN : 978–3–030–04770–2, Toulouse, France, juin 2018. Springer.

[21] M. MAGNAUDET, S. CHATTY, S. CONVERSY, S. LERICHE, C. PICARD et D. PRUN :

Djnn/Smala : A Conceptual Framework and a Language for Interaction-Oriented Programming. *Proceedings of the ACM on Human-Computer Interaction* , 2(EICS):1 – 27, juin 2018. URL `https://hal-enac.archives-ouvertes.fr/hal-01815222`.

[22] P. MASCI, R. RUKŠĖNAS, P. OLADIMEJI, A. CAUCHI, A. GIMBLETT, Y. LI, P. CURZON et H. THIMBLEBY : On formalising interactive number entry on infusion pumps. 45, 01 2011.

[23] S. OWICKI et L. LAMPORT : Proving liveness properties of concurrent programs. *ACM Trans. Program. Lang. Syst.*, 4(3):455–495, juil. 1982. ISSN 0164-0925.

[24] R. RUKŠĖNAS, P. CURZON et A. BLANDFORD : Detecting cognitive causes of confidentiality leaks. *Electronic Notes in Theoretical Computer Science*, 183:21 – 38, 2007. ISSN 1571-0661. Proceedings of the First International Workshop on Formal Methods for Interactive Systems.

[25] L. SU, H. BOWMAN et P. BARNARD : Performance of reactive interfaces in stimulus rich environments, applying formal methods and cognitive frameworks. *Electronic Notes in Theoretical Computer Science*, 208:95 – 111, 2008. ISSN 1571-0661. Proceedings of the 2nd International Workshop on Formal Methods for Interactive Systems.

[26] E. R. TUFTE : *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT, USA, 1986. ISBN 0-9613921-0-X.

[27] C. WARE : *Information Visualization : Perception for Design*. Morgan Kaufmann Series in Interactive Technologies. Morgan Kaufmann, Amsterdam, 3 édn, 2012. ISBN 978-0-12-381464-7.

[28] L. WILKINSON : *The Grammar of Graphics (Statistics and Computing)*. Springer-Verlag, Berlin, Heidelberg, 2005. ISBN 0387245448.

[29] E. WILLIAMS : Airborne collision avoidance system. *In Proceedings of the 9th Australian Workshop on Safety Critical Systems and Software - Volume 47*, SCS '04, p. 97–110, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc. ISBN 1-920-68229-5.