HAL
open science

Nouvelle génération d'Interfaces Homme-Machine pour mieux agir et mieux comprendre

Jean-Daniel Fekete

To cite this version:

Jean-Daniel Fekete. Nouvelle génération d'Interfaces Homme-Machine pour mieux agir et mieux comprendre. Interface homme-machine [cs.HC]. Université Paris Sud - Paris XI, 2005. tel-00876183

HAL Id: tel-00876183

https://theses.hal.science/tel-00876183

Submitted on 24 Oct 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HABILITATION À DIRIGER DES RECHERCHES

Présentée par

Jean-Daniel Fekete

Spécialité : informatique

Nouvelle génération d'Interfaces Homme-Machine pour mieux agir et mieux comprendre

2 mai 2005

Mme Coutaz, Joëlle	Rapporteur
M. Greenberg, Saul	Rapporteur
M. Shneiderman, Ben	Rapporteur
M. Beaudouin-Lafon, Michel	Examinateur
M. Ganascia, Jean-Gabriel	Examinateur
M. Mélançon, Guy	Examinateur
M. Puech, Claude	Examinateur

Habilitation à Diriger les Recherches préparée au sein de l'**INRIA Futurs** et du **L**aboratoire de **R**echerche en **I**nformatique

Remerciements

La vie d'un chercheur est fait de beaucoup de travail, nécessitant beaucoup de patience de la part de ses proches. Je remercie Nicole et Adrien pour avoir été patients lorsqu'il le fallait et m'avoir toujours soutenu.

Je remercie tous les membres de mon jury pour leurs conseils avisés et leur support. En particulier, je suis reconnaissant et honoré que Ben Shneiderman, de l'université du Maryland, ait bien voulu être rapporteur. Ben Shneiderman a toujours témoigné d'un grand intérêt pour mes recherches et m'a prodigué des conseils positifs et utiles pour les mener à bien. Je tiens donc à lui témoigner ici ma reconnaissance et mon amitié.

Joëlle Coutaz a bien voulu rapporter sur mon habilitation, après avoir déjà accepté de participer au jury de ma thèse. J'en conclus donc qu'elle n'est pas encore lassée de mes travaux ! Joëlle est non seulement une chercheuse, professeure à l'université de Grenoble, renommée internationalement mais aussi une infatigable animatrice de la recherche en IHM française. Nos rencontres, au gré des conférences et réunions, ont toujours été enrichissantes et utiles pour mes axes de recherche. Je la remercie pour sa participation à ce jury.

Saul Greenberg dirige le laboratoire de recherche « Grouplab » à l'université de Calgary. J'ai réalisé, lors d'un séjour qu'il a fait en France, à quel point Grouplab était proche du projet IN-SITU dont je fais partie à l'INRIA : il combine des préoccupations d'interaction Homme-Machine avancées, de collecticiel et de visualisation d'information en les abordant d'un point de vue conceptuel sans pour autant négliger leur mise en œuvre. Je lui suis reconnaissant d'avoir accepté d'être rapporteur et fait l'effort de lire mon document en français.

Je remercie les examinateurs de mon jury :

- Claude Puech, directeur de l'INRIA Futurs, pour m'avoir toujours encouragé dans mes travaux de recherche et apporté un grand soutien à notre projet de recherche.
- Jean-Gabriel Ganascia, Professeur à l'université Pierre et Marie Curie, a exprimé un grand intérêt à mes travaux de visualisation et d'analyse des sources manuscrites. Je le remercie ici d'apporter un éclairage « Intelligence Artificielle » et « Cognition » à ce jury.

– Guy Mélançon, Professeur à l'université de Montpellier III, est une référence internationale en visualisation d'information et placement de graphes. Son parcours issu des mathématiques et allant vers l'informatique interactive éclairera ce jury d'un œil exigeant de théoricien.

Enfin, je voudrais remercier plus spécialement Michel Beaudouin-Lafon qui a bien voulu participer à mon jury, mais qui a surtout supervisé et accompagné mes travaux de recherche depuis le début de ma thèse. Je tiens à lui témoigner ici ma reconnaissance. Michel, avec Wendy Mackay, responsable du projet IN-SITU, ont créé un groupe de recherche, au sein de l'INRIA et du Laboratoire de Recherche en Informatique de l'université Paris-Sud, dans lequel mon travail de recherche et d'encadrement sont à la fois agréables, passionnants, enrichissants et féconds.

Table des matières

Remerciements **i**

Introduction **1**
Diversification 1
Changement d'échelle 2
Comprendre ou trouver 4
Grands défis et voies de recherche 5

1 Outils pour l'IHM et leur évolution **7**
1.1 Evolution des outils pour l'IHM 7
1.1.1 Stéréotypes 8
1.1.2 Architecture des graphes de scène 9
1.1.3 Discussion 10
1.2 Contributions à de nouveaux outils 11
1.2.1 Nouveaux modèles de sortie 11
1.2.2 Prise en compte des nouveaux matériels graphiques 13
1.2.3 Nouveaux modèles d'entrée 15
1.2.4 Articuler les nouveaux modèles 16

2 Gérer l'augmentation quantitative des données **19**
2.1 Visualisation d'information 20
2.2 Méthodes d'évaluation pour la visualisation 22
2.3 Contributions en visualisation d'information 23

3 Comprendre plus, comprendre plus vite, comprendre mieux **27**
3.1 Comprendre l'histoire à partir des sources 28
3.2 Comprendre l'activité créative littéraire 28
3.3 Comprendre le comportement d'un programme 32

4 Conclusion et perspectives **33**
4.1 Perspectives en architecture des systèmes interactifs 33

4.2 Perspectives pour la visualisation d'information 34

Bibliographie **37**

Annexes 51

1 Input Device Selection and Interaction Configuration with ICon **51**

2 The MaggLite Post-WIMP Toolkit **59**

3 Excentric Labeling : Dynamic Neighborhood Labeling for Data Visualization **69**

4 Readability of Graphs Using Node-Link and Matrix-Based Representations **77**

5 Les leçons tirées des deux compétitions de visualisation d'information **91**

6 Compus : Visualization and Analysis of Structured Documents For Understanding Social Life in the 16th Century **97**

7 Peeking in Solver Strategies : Using Explanations Visualization of Dynamic Graphs for Constraint Programming **107**

Introduction

L'interaction Homme-Machine est un domaine de recherche et d'application mûr et établi. Cependant, depuis une dizaine d'années, il subit une mutation importante liée en particulier à la diversification des supports et des usages ainsi qu'au changement d'échelle des données qui transitent ou résident sur nos ordinateurs.

Diversification

La diversification s'exprime de plusieurs manières :

- des plates-formes nouvelles apparaissent sur le marché et ou sous forme de prototypes de recherche : téléphone portable, assistant personnel, TabletPC, PocketPC, montre incluant un assistant personnel, tableau de bord de voiture numérique et ordinateur vestimentaire ;
- des dispositifs d'entrée nouveaux sont disponibles qu'ils soient spécifiques à une plate-forme ou qu'ils soient génériques : souris avec un nombre croissant de boutons, de molettes ou de surfaces tactiles, tablettes avec pointeurs sensibles à la pression et à l'inclinaison, manettes de jeu, souris à retour d'effort, stylets sur écran pour les assistants personnels, TabletPC et PocketPC, reconnaissance de la parole et de la vidéo ;
- des dispositifs de sortie nouveaux sont disponibles : très petits écrans pour téléphones portables ou montres, écrans de résolution croissante pour assistants personnels, écrans de haute résolution ou de très grande taille pour ordinateurs de bureaux, grilles d'écrans ou de projecteurs, encre numérique ;
- les utilisateurs se diversifient, tant en âge qu'en compétences ou handicaps : l'informatique se diffuse dans la société.

Cette diversité est aujourd'hui une grande source de difficultés pour la recherche et pour l'industrie. Les modèles architecturaux et les outils de conception et de réalisation de systèmes interactifs n'arrivent plus à la gérer convenablement.

Nous décrivons en chapitre 1 notre vision unifiée de l'architecture logicielle des systèmes interactifs pour faciliter la construction d'applications graphiques *hautement interactives* et *profondément adaptables*. Nous montrons quelques ou-

tils fondant cette vision et qui permettent de construire des applications graphiques interactives qui tirent parti des capacités spécifiques de la plate-forme sur laquelle elles s'exécutent. Il ne s'agit pas ici d'adaptation automatique sur laquelle nous sommes très réservés. Nous pensons que les architectures logicielles pour l'IHM avancée (non stéréotypée) doivent offrir des mécanismes simples et extensibles, et séparer le plus clairement possible les deux aspects de l'interaction que sont le traitement des entrées et celui des sorties pour faciliter la progression indépendante des deux.

Changement d'échelle

En plus de cette diversité, l'IHM doit aussi gérer le changement d'échelle des données qui transitent sur nos ordinateurs et que nous manipulons. Ce changement d'échelle nous vient à la fois de l'accès aux données via Internet, mais aussi de la part croissante de communication qui transite sur nos ordinateurs tels le courrier électronique, les photographies numériques et la vidéo numérique. Cet accroissement a été d'environ 30% par an entre 1999 et 2002 selon les recherches du groupe « How Much Information 2003 ? »[1].

L'accroissement à lui seul est important, mais la quantité est encore plus impressionnante : non seulement les données sont produites, mais elles sont de plus en plus accessibles. Selon les études de la société *Search Engine Watch* (`http://searchenginewatch.com/`) visibles sur la figure 0.1, le nombre de pages indexées par les moteurs de recherche a augmenté de manière exponentielle depuis dix ans. Cette augmentation est aussi soutenue par l'accroissement des capacités des disques durs.

Selon [Thompson et al.00], l'accroissement de la capacité de stockage par unité de surface a été plus qu'exponentielle depuis les années 70 et le prix est resté à peu près constant par unité de surface ou a baissé exponentiellement pour une capacité de stockage donnée (figure 0.2).

Face à cette mutation, le domaine de l'interaction Homme-Machine a été jusqu'à présent en retard. Les éléments de l'interface ont bien sûr évolué, mais les ordinateurs de bureau actuels se présentent de manière très similaires à leurs ancêtres des années 80. L'écran a doublé de résolution mais un utilisateur des premiers Macintosh ne serait pas surpris. Chaque nouvelle version des systèmes d'exploitation modernes présente des avancées incrémentales intéressantes mais qui marquent le pas face au changement d'échelle.

Pour gérer la diversité et le changement d'échelle, la seule réponse de l'industrie semble être la prolifération de solutions ad hoc qui mène à une grande

[1] `http://www.sims.berkeley.edu/research/projects/how-much-info-2003/`

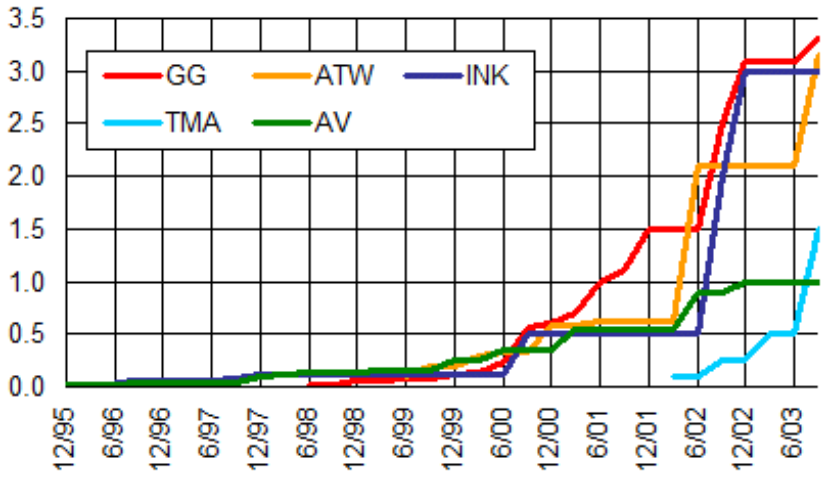


FIG. 0.1 : Evolution du nombre de documents indexés par les moteurs de recherche sur le Web (en milliards)

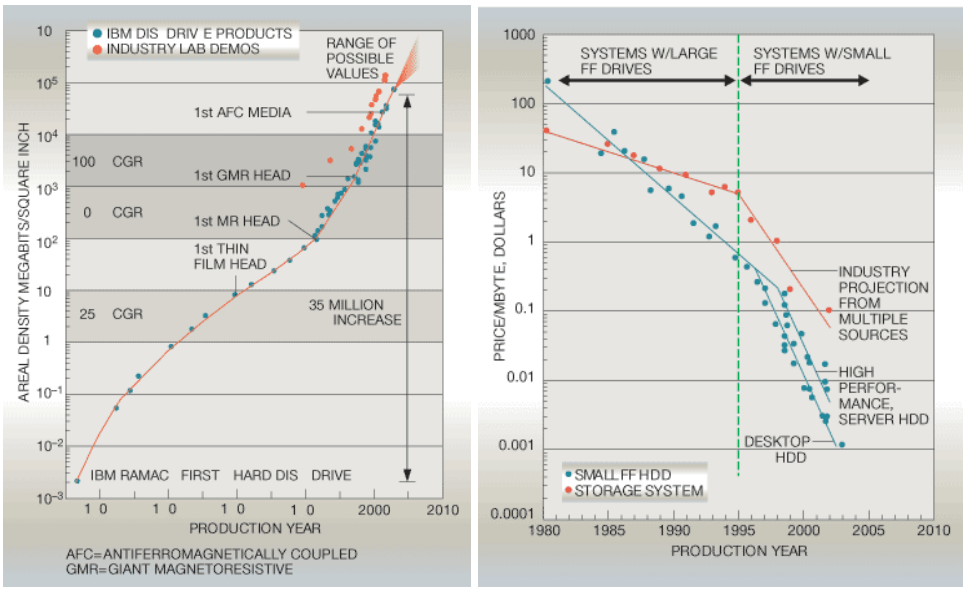


FIG. 0.2 : Evolution des capacités et prix du stockage magnétique depuis 40 ans (extrait de [Thompson et al.00])

fragmentation des systèmes tandis que les concepts n'évoluent pas vraiment.

Le chapitre 2 présente quelques voies que nous avons abordées pour tenter de résoudre les problèmes de changements d'échelles.

Comprendre ou trouver

Pour gérer le changement d'échelle, les systèmes commencent à proposer des mécanismes d'indexation automatique qui facilitent la recherche. C'est le cas du moteur de recherche Google sur Internet, qui offre maintenant une version personnelle pour chercher sur son propre ordinateur : Google Desktop. Cependant, en dehors de ce genre d'ajouts, le paradigme dominant reste le système de fichiers très semblable à celui d'Unix des années 70.

Faciliter la recherche est important, mais nous arguons que cela ne remplace pas une vue d'ensemble et une structuration des données. Trouver rapidement une information est utile, mais savoir comment cette information est organisée permet de *comprendre*, et cette compréhension est une arme importante pour manipuler et gérer des masses de données.

Pour illustrer la différence entre trouver et comprendre, on peut imaginer le monde avant la découverte de la carte topographique. Pour faire un voyage, il fallait alors disposer d'une description du chemin à suivre. L'invention de la carte a permit de représenter non seulement un chemin, mais aussi une infinité de chemins dont la difficulté est visible grâce à la représentation topographique de la carte. Lorsqu'on se perdait en comprenant mal une description de chemin, il fallait revenir en arrière pour reprendre la linéarité de la description. Avec une carte, il devient possible de récupérer un autre chemin et d'identifier des zones mal connues ou inconnues.

Aujourd'hui, les interfaces de recherche nous donnent des chemins vers l'information qui semble pertinente (au moteur de recherche). Une requête à Google concernant l'interaction Homme-Machine retourne des pages qui décrivent chacune une vision du domaine de l'interaction Homme-Machine. Peut-être en les lisant toutes, est-il possible de mieux appréhender le domaine ? Mais quel crédit donner à telle ou telle page ? Les résultats de la requête ne nous permettent pas de comprendre globalement le domaine de l'IHM, seulement d'obtenir des vues multiples et éventuellement contradictoires sans vue d'ensemble. Pour comprendre, il faut une vue d'ensemble — une structure visible et compréhensible — et c'est ce que nous voulons fournir à l'aide de la visualisation d'information.

Grands défis et voies de recherche

Le 20e siècle a connu une évolution importante des méthodes et de l'épistémologie des sciences naturelles. Karl Popper [Popper35] a donné naissance à une vision très exigeante des sciences naturelles où chaque théorie scientifique doit être avancée avec une série de condition permettant de la mettre à l'épreuve. Cette exigence, bien que parfaitement justifiée, a été à l'origine d'un découragement des scientifiques qui perdaient l'aspect positif de leur domaine et devenaient les bourreaux de leurs propres découvertes. Le positivisme est bien terminé mais le poids de la vérité scientifique est difficile à porter.

Lakatos [Lakatos95] a tenté de réintroduire un aspect positif à la recherche scientifique en proposant que les domaines scientifiques cherchent des frontières ou « programmes de recherche » et des « voies » pour les atteindre. Lakatos a aussi montré que les mathématiques pouvaient être considérées comme sciences naturelles dans la mesure où elles étaient produites par des esprits humains et que tout théorème complexe, même vérifié par la communauté, pouvait être entaché d'erreurs. En suivant ce point de vue, l'informatique est aussi une science naturelle et peut bénéficier de la méthode de Lakatos pour lui assurer un avancement positif.

Pour le domaine de l'interaction Homme-Machine, quels pourraient être les grands défis et les voies de recherche ? La société marchande s'est fixée des objectifs d'amélioration de la productivité à l'aide de l'informatique : faire plus, faire mieux pour moins cher.

D'un point de vue scientifique et humaniste, je fixerai comme grands défis : comprendre plus, comprendre plus vite, comprendre mieux.

Il s'agit donc de trouver des théories cognitives, des modèles et des outils informatiques pour nous aider à comprendre et pas seulement pour agir. L'hypertexte et le World Wide Web ont déjà provoqué une révolution dans la diffusion des informations, comme le livre l'a provoquée à la Renaissance, mais à une échelle bien supérieure [Shneiderman02].

Ces grands défis posent immédiatement le problème de l'évaluation des résultats. Quelles métriques pouvons-nous adopter pour juger de nos progrès ?

Pour répondre à cette question, nous avons initié un travail d'évaluation empirique pour l'IHM et la visualisation d'information. Le domaine de l'IHM et de la visualisation d'information cherchent encore leurs théories et modèles prédictifs qui permettraient de prévoir la qualité d'une interface ou d'une technique de visualisation avant même de l'avoir réalisée. Ces théories ou modèles existent dans des domaines très spécifiques (Loi de Fitts [Fitts54], vision préattentive [Treisman85]) mais ne peuvent encore être appliqués à des systèmes interactifs complets [Card et al.83]. Pour avancer dans l'évaluation, nous avons donc initié la conception de *benchmarks* en IHM et visualisation d'information afin de pouvoir comparer des systèmes et techniques existants, à défaut de modéliser ou prévoir leur efficacité

analytiquement. Le chapitre 2.2 décrit notre contribution dans ce domaine.

Chapitre 1

Outils pour l'IHM et leur évolution

Nous l'avons souligné dans l'introduction : les outils actuels pour l'IHM ne supportent pas l'accroissement de diversité des contextes et usages, d'où une prolifération de ces outils, chaque nouvel usage venant avec un nouvel outil.

1.1 Evolution des outils pour l'IHM

Nous pouvons analyser l'évolution des boîtes à outils d'IHM depuis les années 75. Avec l'apparition de l'Alto de Xerox et du langage Smalltalk [Tesler81], les boîtes à outils s'orientent vers les technologies à objets. Alan Kay, un des pionniers de l'environnement Smalltalk, a ainsi défini ses objectifs : « Simple things should be simple. Complex things should be possible. » Smalltalk sur les machines Alto de Xerox a été le premier langage à offrir un environnement de programmation cohérent pour le développement d'applications graphiques et interactives. Un concept fondateur mis en avant par Smalltalk a été l'idée de séparation entre les *modèles* — la représentation d'un concept à base d'objet dans un programme Smalltalk — et ses représentations graphiques : les *vues*. Entre le modèle et ses vues, le *contrôleur* a une fonction : interpréter les interactions appliquées à la vue — à l'aide de la souris et du clavier — en terme de modification sémantique du modèle. Ce concept est connu sous le nom MVC pour modèle, vue et contrôleur.

Par exemple, un nombre entier est représenté par une instance de la classe Nombre en Smalltalk. Cette instance peut être affichée à l'écran dans une interface sous plusieurs formes : champ textuel affichant la valeur, jauge linéaire ou angulaire, etc. Chacune de ces formes est une *vue* du nombre. Un *contrôleur* spécifique existe entre le nombre et chacune de ses vues. Il met à jour la vue en fonction de la valeur du nombre et, lorsque l'utilisateur manipule la représentation graphique à l'aide d'une souris ou d'un clavier, il change le nombre.

Cette séparation conceptuelle a été raffinée par le modèle architectural PAC

(Présentation, Abstraction, Contrôleur) de Joëlle Coutaz [Coutaz87]. Les composantes présentation et abstraction sont des généralisations des vues et modèles de MVC. Le contrôleur a quand à lui deux fonctions distinctes : la traduction des attributs sémantiques du modèle en attributs graphiques sur la présentation, et l'interprétation des interactions appliquées à la vue en terme de modification sémantique de l'abstraction.

Par la suite, les langages et environnements de programmation ont beaucoup changé. Les premiers environnements interactifs ayant eu un succès commercial ont utilisé des langages compilés et beaucoup moins dynamiques que Smalltalk : Object Pascal ou C pour les premiers Macintosh, C pour les premières versions de Windows ou pour le système X sous Unix et ses variantes. Le modèle objet a été remplacé par un modèle à composants, intégrant la vue et le contrôleur dans un même bloc. Ces composants communiquent avec l'extérieur à l'aide de mécanismes variés et plus ou moins commodes comme les *callbacks* [McCormack88], événements, appels de scripts [Goodman87], variables actives [Ousterhout94] ou objets de notification [Geary et al.97] pour ne citer que ceux là.

L'aspect monolithique des composants d'interaction a été très tôt reconnu comme un problème et plusieurs boîtes à outils de recherche ont tenté d'apporter un peu de modularité à cette organisation logicielle. Garnet et plus tard Amulet [Myers et al.90, Myers95] ont tenté de séparer les objets graphiques, leur placement et leur interaction. Pour le placement, Garnet et Amulet ont utilisés des contraintes unidirectionnelles tandis que, pour la gestion des entrées, ils utilisaient des *interacteurs* [Myers91]. Garnet représente la dernière génération de boîtes à outils construite sur un langage dynamique (Lisp) étendu spécifiquement pour faciliter la construction d'interfaces. Amulet a été une tentative d'adaptation des concepts applicables aux langages dynamiques dans un langage compilé (C++). Les interacteurs ont permis d'isoler un peu le graphique de l'interaction mais ils sont apparus au moment où les plateformes matérielles ont commencé à proliférer sous des formes très diverses.

Les boîtes à outils graphiques interactives ont alors évolué dans deux directions : la capture de stéréotypes et le passage des composants aux graphes de scènes.

1.1.1 Stéréotypes

La capture de stéréotypes pour la construction d'applications graphiques interactives a débuté avec MacApp [Schmucker87]. L'idée était la suivante : puisque les applications graphiques interactives sont difficiles à construire, il faut trouver des structures communes à tous ces programmes et donner des mécanismes pour organiser ces structures. MacApp a été suivi de plusieurs générations de *squelettes d'applications* ou *Application Frameworks* [Lewis et al.95] comme ET++ [Wei-

nand et al.88] ou Unidraw [Vlissides et al.89]. Ces architectures ont abouti aux « patrons de conception » ou *Design Patterns* [Gamma et al.94] qui ont abstrait les squelettes d'application.

Cependant, en stéréotypant la construction d'applications interactives, on a rendu leur évolution très difficile. Lorsque les plateformes interactives nouvelles sont apparues tels les assistants personnels ou les environnements de réalité virtuelle, elles ont développé des squelettes différents, adaptés à leur problématique spécifique faute de pouvoir adapter les squelettes existants. Il en a résulté une prolifération d'environnements interactifs qui, chacun, résoud assez bien quelques problèmes spécifiques, mais globalement n'ont pas apporté d'innovations importantes depuis dix ans [Fekete et al.01, Shneiderman et al.04b].

1.1.2 Architecture des graphes de scène

En parallèle de cette évolution vers des stéréotypes, certaines boîtes à outils ont tenté de séparer la partie graphique de la partie interactive afin de séparer les problèmes.

Les interacteurs de Garnet ont été dans cette direction pour l'interaction. Pour l'aspect graphique, InterViews [Calder et al.90] a été le premier système à tenter de définir des objets graphiques simples pour les interfaces. Il a été suivi de boîtes à outils plus radicales utilisant des *graphes de scène*.

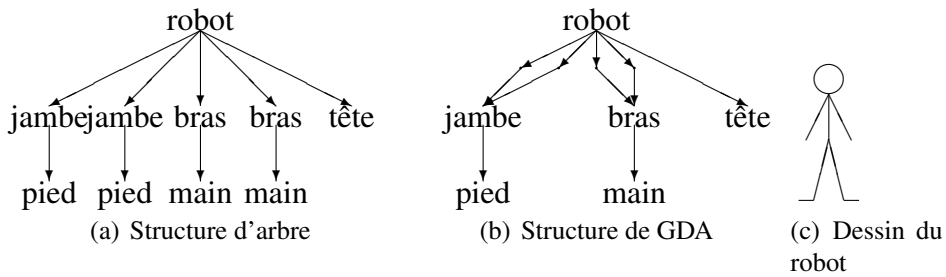
Les graphes de scène sont des structures de données très répandues dans les boîtes à outils graphiques 3D, telles OpenInventor [Open Inventor Architecture Group94] ou OpenSceneGraph (`www.openscenegraph.org`).

Une scène graphique est composée d'un graphe orienté acyclique de nœuds, chaque nœud pouvant représenter une géométrie, un attribut graphique ou un groupement de nœuds. L'affichage d'une scène se fait en affichant chaque nœud dans un parcours préfixe du graphe.

Cette structure de graphe permet de définir une hiérarchie dans les parties constituant une scène. Le fait qu'il s'agisse d'un graphe orienté et acyclique plutôt que d'un arbre permet de partager des nœuds, à la fois pour des raisons d'héritage et aussi de réduction de place mémoire lorsque des parties de modèles sont volumineux et utilisés de nombreuses fois dans une scène.

Par exemple, la figure 1.1 montre une scène représentant un simple robot structuré sous la forme d'un arbre ou d'un graphe orienté acyclique (DAG). Les articulations sont représentées par des nœuds de transformation tandis que les membres sont des nœuds de géométrie.

Depuis une dizaine d'années, des boîtes à outils 2D expérimentales prônent l'utilisation de ces graphes de scène pour l'interaction en montrant qu'elles favorisent la richesse graphique sans rendre plus complexe l'interaction [Bederson et al.00, Bederson et al.04].



(a) Structure d'arbre (b) Structure de GDA (c) Dessin du robot

FIG. 1.1 : Graphe de scène pour modéliser un robot

1.1.3 Discussion

Malgré tout, les boîtes à outils actuelles convergent vers un modèle unique à composants et non plus à objet [Lorenz et al.01]. Depuis les années 90, toutes les boîtes à outils offrent des composants graphiques interactifs qui encapsulent une apparence et un comportement interactif. Les modèles de communication entre composants sont essentiellement basés sur l'envoi d'événement et la notification. Depuis les années 90, ce modèle n'a pas évolué tandis que le nombre des composants a augmenté (légèrement) et que le nombre de boîtes à outils d'interaction a littéralement explosé.

Que peut-on reprocher à ces boîtes à outils ?

1. Elles stéréotypent les applications.
2. Elles ne permettent pas d'étendre les composants, seulement de les réutiliser.
3. Elles ne sont pas conçues pour que les interactions soient extensibles.
4. Elles ne facilitent pas l'évaluation de la qualité des applications graphiques qu'elles permettent de construire.

Et pourtant, les boîtes à outils prolifèrent. Un recensement des boîtes à outils maintenu à jour sur le site `http://www.atai.org/guitool/` en dénombre 109 non commerciales — dont 31 qualifiées de « bonnes » — et 35 commerciales. Le site de logiciels libres `sourceforge.net` recense 89 projets dans la catégorie « interfaces utilisateurs » dont environ la moitié prétend être une boîte à outil graphique.

Même en ne se limitant qu'aux boîtes à outils populaires, la vitesse à laquelle elles changent force les programmeurs à changer rapidement : Motif/MacApp dans les années 90, Microsoft MFC ou Java/AWT dans les années 95, Java/Swing ou C#/.NET dans les années 2000. Quelles différences peut-on voir à l'écran en examinant les générations de systèmes de la figure 1.2 ? Pas beaucoup et sur des points bien précis : la couleur s'est répandue à la fin des années 80 et la résolution

moyenne a un peu progressé. Pour le reste, nous avons toujours une souris, un clavier, un écran, des menus et des icônes, et la vitesse à laquelle nous interagissons, a probablement un peu augmenté, mais certainement pas dans les proportions des données que nous avons maintenant à manipuler.

1.2 Contributions à de nouveaux outils

Nous pensons qu'il existe une alternative à la prolifération de systèmes ad hoc : la conception et validation de modèles plus flexibles et unificateurs sur lesquelles nous avons travaillé et apporté des avancées.

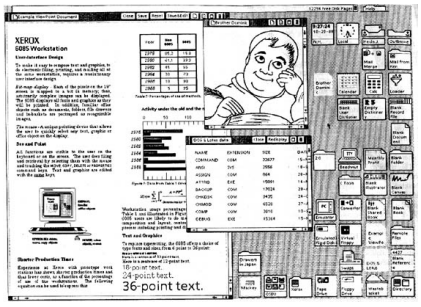
Nous prônons une approche *système* des outils plutôt qu'une approche *utilisateur*. En cela, nous sommes à l'opposé des tendances actuelles. De notre point de vue, les approches centrées sur l' utilisateur [Shneiderman et al.04a] permettent de comprendre le problème, mais la mise en oeuvre des solutions est, *in fine*, un problème technique, filtré par des contraintes liées aux utilisateurs. La variété des utilisateurs et des contextes d'usage n'ont pas changé la nature des ordinateurs, basée sur le modèle de Von Neuman depuis un demi-siècle. De la même manière, nous pensons qu'il existe des solutions techniques génériques, reposant sur des modèles conceptuels informatiques unificateurs et appropriés pour décrire les besoins des IHM. En adoptant une approche ad hoc, le domaine de l'IHM a, de notre point de vue, bloqué l'évolution des langages et modèles adaptés à ses problèmes, ce qui a obligé notre domaine à travailler à un niveau d'abstraction très bas avec des outils très primitifs.

Nos travaux ont porté sur trois formes d'améliorations :

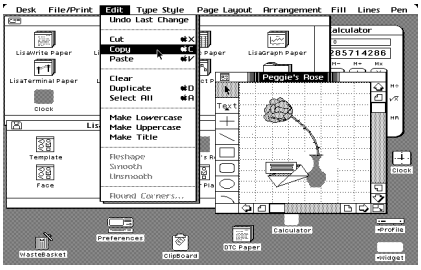
- un modèle en couches pour faciliter la séparation des tâches entre interaction et graphique ;
- un système de gestion des entrées étendues basé sur un modèle de réseau réactif ;
- une architecture de graphe mixte pour combiner un réseau réactif et un graphe de scène.

1.2.1 Nouveaux modèles de sortie

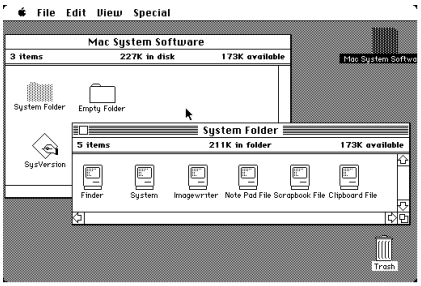
Les écrans prolifèrent sous des formes très variées : téléphones portables, appareils photos numériques, lecteurs de DVD portable, assistants numériques, etc. Les constructeurs de processeurs graphiques améliorent sans cesse les performances et la miniaturisation, permettant l'utilisation de graphique 3D accélérée sur des téléphones portables par exemple. Depuis dix ans, la vitesse des processeurs graphiques a pratiquement doublé tous les ans.



(a) 1981 Xerox Star

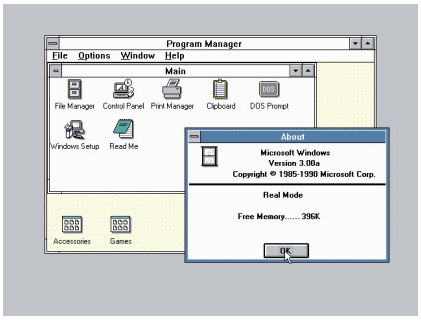


(b) 1983 Apple Lisa

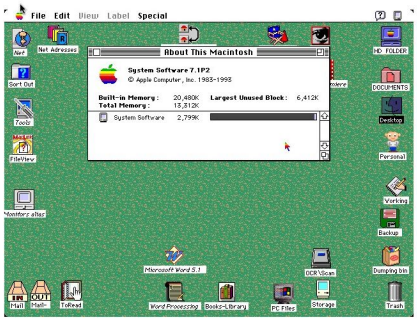


(c) 1984 Macintosh System 1

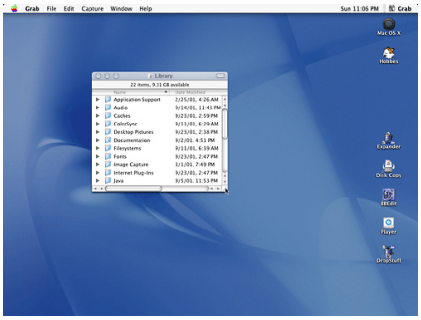
(d) 1985 NeXTSTEP (OpenStep here)



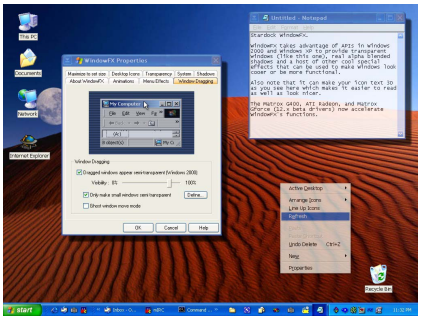
(e) 1990 Windows 3.0



(f) 1992 Macintosh System 7



(g) 2001 MacOS X 10.1



(h) 2001 Windows XP

FIG. 1.2 : Comparaison de systèmes de fenêtrages produits dans les années 80, 90 et 2000

Cependant, il semblerait que seuls les jeux vidéo arrivent à suivre l'évolution des capacités. Nos interfaces graphiques restent similaires à ce qu'elles étaient il y a dix ans. Les boîtes à outils graphiques reposent toujours sur les mêmes bibliothèques graphiques.

Vouloir utiliser la nouvelle puissance disponible semble logique, mais pour quelles raisons est-ce important ? Il y en a au moins quatre :

- la richesse graphique améliore le confort de travail ;
- la vitesse permet d'augmenter le nombre d'objets manipulables interactivement, facilitant un travail interactif sur des données plus volumineuses ;
- la vitesse permet l'utilisation de transitions animées dans les interfaces et les transitions évitent les changements brusques de configurations qui peuvent déconcerter les utilisateurs ;
- certains effets graphiques comme la transparence ou les traitements d'image [Baudisch et al.04, Gossett et al.04] améliorent la lisibilité des interfaces et peuvent être calculés directement sur les nouvelles cartes graphiques.

Utiliser pleinement les nouvelles capacités des processeurs graphiques reste aujourd'hui difficile car l'architecture matérielle de ces processeurs favorise un mode de programmation très particulier. Il n'est pas possible d'assurer la compatibilité avec les anciennes bibliothèques graphiques et de bénéficier pleinement de la vitesse des nouvelles.

Nos travaux initiaux ont montré qu'une architecture graphique en couches permettait de séparer et de simplifier la spécification de l'interaction et du graphique [Fekete et al.96]. Nous avons validé notre approche initiale en réalisant des éditeurs graphiques utilisés dans des environnements professionnels très exigeants en terme d'intégration et de performances [Fekete et al.95].

1.2.2 Prise en compte des nouveaux matériels graphiques

Les bibliothèques graphiques récentes comme Jazz ou Piccolo [Bederson et al.00, Bederson et al.04] fonctionnent toujours selon un modèle très semblable aux langages interprétés : chaque instruction est effectuée dans l'ordre spécifié et est totalement indépendante des instructions précédentes et suivantes. Les langages interprétés ont été très populaires jusqu'aux années 80 et à l'avènement des architectures RISC. Ils ont ensuite été remplacés par les langages compilés qui permettaient de tirer parti des nouveaux processeurs.

Pour profiter pleinement de leurs capacités, les nouveaux processeurs graphiques nécessitent une phase de compilation. Pire encore, ils nécessitent la mise au point de structures de contrôle et de données spécifiques, semblables à celles inventées pour les ordinateurs vectoriels. Les nouveaux processeurs graphiques disposent de multiples unités de calculs parallèles qui effectuent des calculs en

cascade (pipeline). Pour bénéficier de toute leur vitesse, il faut utiliser toutes les unités parallèles et éviter de casser la cascade.

Dans le système SVGL [Conversy et al.02], nous avons tenté d'utiliser le langage de graphe de scène SVG [Ferraiolo et al.03] spécifié par le consortium WWW et de l'afficher en utilisant le système graphique OpenGL [Opengl architecture review board et al.03]. SVGL est une expérimentation qui n'a pas porté tous ses fruits car SVG s'est avéré trop complexe pour nos expérimentations. Nous avons cependant obtenu des accélérations significatives de l'affichage comparé aux versions industrielles de visualiseurs de SVG uniquement en utilisant OpenGL, avec quelques optimisations mais sans compilation.

Les langages de graphes de scène sont parfaitement adaptés à la programmation d'applications graphiques interactives ou quelques milliers d'objets graphiques sont affichés sur un écran. La quasi totalité des applications graphiques interactives actuelles entre dans ce cadre. Quelques bibliothèques graphiques utilisant les graphes de scènes 2D sont déjà utilisables depuis quelques années [Bederson et al.00, Pietriga02, Heer et al.05, Lecolinet03]. Les applications construites à partir de ces bibliothèques sont plus variées et riches interactivement que si elles avaient été programmées à l'aide de bibliothèques de composants [Bederson et al.04]. En comparaison, nos travaux se sont aussi occupés des optimisations graphiques.

Constatant que les architectures logicielles à base de composants ne suivaient plus les besoins, nous avons cherché à concevoir et valider une architecture séparant clairement les entrées et les sorties, en basant la sortie graphique sur un graphe de scène. Nous décrivons nos travaux dans les sections suivantes.

Dans [Fekete03a], nous montrons que la bibliothèque graphique standard du langage Java peut être accélérée d'un facteur 10 à 200 en conservant la compatibilité ascendante. Dans [Fekete et al.02], nous utilisons des techniques plus avancées qui permettent d'obtenir des gains encore supérieurs d'un ordre de grandeur.

Pour bénéficier de ces accélérations, il est donc indispensable de changer de paradigme de programmation pour les applications graphiques. Nous avons étudié deux nouveaux type de paradigmes : les graphes de scène [Conversy et al.02, Huot et al.04] et le pipeline graphique de la visualisation d'information.

Même si ces architectures à base de graphe de scène 2D permettent d'obtenir de meilleures performances graphiques, les applications de visualisation d'information nécessitent encore plus de vitesse et nécessitent une autre architecture graphique. C'est ce que nous avons étudié et présenté dans plusieurs articles, soit d'un point de vue expérimental pour cerner les limites accessibles dans [Fekete et al.02], soit pour décrire une architecture de boîte à outils permettant de manipuler des grands jeux de données très rapidement à l'aide de structures de données relativement aisées à manipuler, très rapides à afficher et très compactes en mémoire. L'architecture que nous avons proposée est très semblable à celle proposée par

le système Tulip [Auber01], développé parallèlement au LaBri à l'université de Bordeaux.

1.2.3 Nouveaux modèles d'entrée

La prolifération des dispositifs d'interaction nouveaux et des besoins de personnes plus particulières — que ce soit des compétences supplémentaires comme les graphistes et les musiciens, ou des déficiences comme certains handicaps — nous ont amenés à porter nos recherches vers des paradigmes d'entrée qui soient extensibles, relativement faciles à programmer et autant que possible fondés sur des bases claires et sûres.

Les travaux initiaux de Philippe Brun [Brun et al.95] et du groupe de recherche ALF du Groupement de Recherche (GDR) I3 nous ont permis d'explorer une très grande quantité de langages, systèmes et formalismes destinés à décrire l'interaction sous une forme ou une autre. Parmi ces langages, nous nous sommes concentrés sur la famille des langages réactifs [Berry et al.85] qui nous semblait offrir un bon compromis entre pouvoir d'expression et simplicité d'implémentation dans le cadre de l'IHM.

Le système ICon [Dragicevic et al.04b, Dragicevic et al.04a, Fekete et al.00a, Dragicevic et al.99] conçu et réalisé par Pierre Dragicevic pendant sa thèse, permet d'accéder à des dispositifs étendus à partir d'applications graphiques interactives programmées en Java et de définir des styles d'interaction sans avoir à changer l'application graphique reposant sur ICon. Un article publié à la conférence IHM-HCI 2000 est reproduit page 51. Le système a permis de concevoir et de réaliser très rapidement des techniques d'interactions originales. Il a aussi permis de trouver des configurations matérielles/logicielles très spécifiques à des applications particulières [Huot04].

ICon offre aussi un environnement de programmation visuelle par flot de données et la possibilité de configurer une application graphique « à chaud » pendant son exécution. Ces possibilités sont très intéressantes à la fois pour prototyper de nouvelles interactions, mais aussi pour adapter un programme à son environnement qui peut changer dynamiquement. ICon a ouvert une voie extrêmement féconde en permettant d'expérimenter en quelques heures des configurations en entrée ou des styles d'interaction qui nécessitaient des mois de travail avec les technologies précédentes.

D'autres boîtes à outils ont été développées pour accéder à des dispositifs d'entrée étendus comme les Phidgets [Greenberg et al.02]. Contrairement à ICon, les Phidgets mettent l'accent sur la création de nouveaux dispositifs physiques à partir d'une astucieuse conception de circuits électroniques très simple, l'interface de programmation restant très classique. ICon peut être utilisé pour accéder aux Phidgets aussi bien qu'à d'autres types de dispositifs matériels.

D'autres formalismes sont actuellement étudiés pour décrire l'interaction à partir de machines à états hiérarchiques (MEH) en particulier. Blanch [Blanch02] décrit une approche consistant à augmenter le langage de programmation en lui incluant des constructions syntaxiques pour l'interaction à base de MEH. Chatty [Chatty et al.04] utilise aussi ces MEH mais en séparant la description de l'interaction du langage de programmation sous-jacent, les MEH étant décrites dans un fichier externe en XML.

1.2.4 Articuler les nouveaux modèles

Avons-nous progressé en proposant de séparer l'architecture graphique interactive à base de composants qui prévaut aujourd'hui par plusieurs modèles séparés : un pour gérer les entrées et un autre pour gérer l'affichage ? Conceptuellement, la réponse est certainement positive mais pratiquement il reste encore à montrer comment deux modèles peuvent cohabiter et interagir en restant compréhensibles.

Nous avons proposé, dans le système MaggLite [Huot et al.04] (page 59), la notion de « graphe mixte » d'interaction pour décrire une architecture où un graphe de scène décrit les objets graphiques et crée des points d'entrées où le système réactif gérant l'interaction peut se raccrocher. Ce système permet une très grande richesse graphique et une très grande richesse d'interaction, bien supérieur aux systèmes existants. Cette flexibilité n'est pas qu'un exercice de style car il simplifie considérablement la construction d'applications graphiques interactives avancées. A titre d'exemple, un générateur d'interface pour MaggLite a été fait avec MaggLite en une journée !

Sur le même modèle, mais en parallèle de nos travaux, la société IntuiLab a réalisé une boîte à outils similaire à des fins industriels. Leur expérience [Chatty et al.04] semble confirmer le fait que l'architecture en graphe mixte facilite le développement tant du point de vue de la qualité que de la vitesse de programmation. Elle montre aussi qu'en utilisant un standard comme le langage SVG, il devient possible de demander à des graphistes d'utiliser leurs outils favoris pour dessiner des interfaces sans compromis sur la qualité graphique.

En parallèle et depuis plusieurs années, plusieurs projets de recherche tentent de décrire les interfaces et interactions sous une forme beaucoup plus abstraite afin de permettre à la fois une spécification de haut niveau, des vérifications [Szekely et al.93] et plus récemment la génération d'interfaces sur des machines spécifiques [Thevenin et al.99]. Notre démarche est différente et complémentaire car elle propose des mécanismes génériques pour faire fonctionner ces interfaces et éventuellement pour en vérifier certaines propriétés tout en restant à un niveau d'abstraction relativement proche de la machine.

Nous pensons donc que les architectures à base de graphes mixtes vont devenir plus populaires dans le monde industriel. Ils ouvrent aussi la voie à des recherches sur l'optimisation de graphes de scène 2D et à un éclaircissement de la notion d'instrument [BL00] qui semble être le concept émergeant pour joindre l'interaction de bas niveau basé sur les signaux émis par des dispositifs d'entrée et des interactions de haut niveau que veulent voir les utilisateurs et les programmeurs d'applications.

Chapitre 2

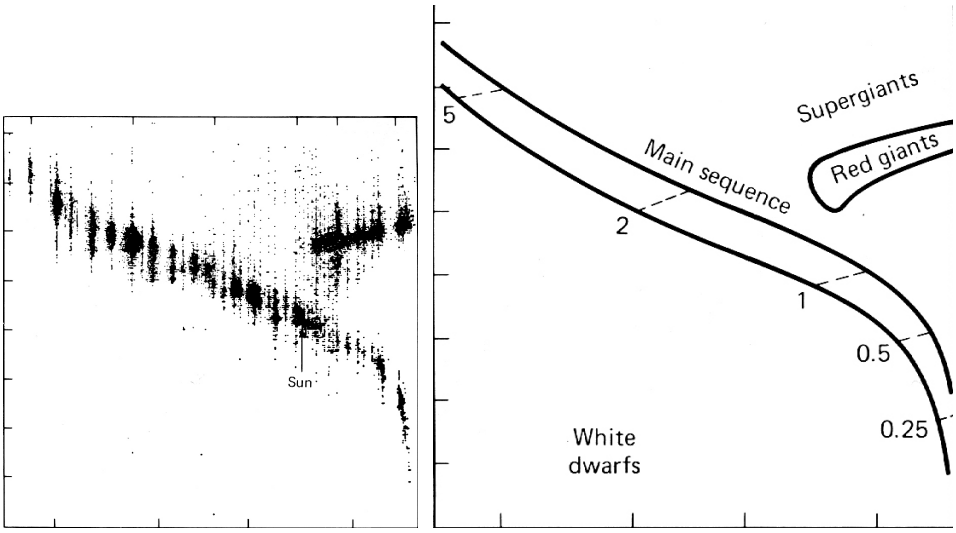
Gérer l'augmentation quantitative des données

L'augmentation des données produites ou accessibles par l'homme a été prodigieuse depuis vingt ans. Pour tirer parti de ces données, l'informatique a d'abord conçu des systèmes pour les gérer : les gestionnaires de bases de données. Ces systèmes permettent aujourd'hui de s'assurer que les données existantes et à venir ne seront pas perdues et resteront consultables. Dans un deuxième temps, des systèmes ont été conçus pour explorer les données afin d'essayer de trouver des régularités, des tendances, des exceptions ou des stéréotypes. Les banques par exemple sont très friandes de modèles sur leurs clients afin de les classer en groupes cohérents et de proposer des offres ciblées à chaque groupe. Ce domaine d'étude est la fouille de données, basé sur les analyses statistiques et l'intelligence artificielle.

Depuis une quinzaine d'années, le domaine de la visualisation d'information s'est attaqué au problème fondamental de l'analyse des données en s'appuyant sur notre perception visuelle. La visualisation d'information se base sur les capacités de traitement de la perception humaine pour effectuer l'analyse des données. Elle cherche les représentations graphiques les plus appropriées pour que notre système perceptif puisse faire ses analyses.

Peut-on vraiment faire confiance en notre perception ? On peut répondre de trois manières à cette question :

- le psychologue peut montrer qu'en utilisant la vision préattentive [Treisman85], l'homme est capable de réaliser des tâches particulières avec une grande rapidité et une grande précision ;
- l'expérience accumulée par l'utilisation de la visualisation d'information dans des domaines variés nous montre que des résultats sont effectivement obtenus ;
- enfin, le bon sens nous permet de nous demander si les autres méthodes exploratoires nous disent toute la vérité : la réponse est non !



(a) Diagramme de Hertzsprung Russell (b) Interprétation usuelle du diagramme de Hertzsprung Russell

Un exemple intéressant est le diagramme de Hertzsprung Russell (figure 2.1(a)), représentant la température des étoiles sur l'axe des X et leur magnitude sur l'axe des Y. Ce diagramme est une représentation sous forme de nuage de points des données issues d'observations astronomiques.

L'interprétation de ce diagramme est assez facile à faire visuellement : on distingue quatre régions décrites dans à la figure 2.1(b). Il s'avère que cette interprétation a été jusqu'à présent impossible à faire à partir de systèmes automatiques de classification [Spence et al.93].

2.1 Visualisation d'information

La visualisation d'information a donc pour objectif d'utiliser la perception visuelle et l'interaction pour faire apparaître des structures visuelles facilitant et accélérant l'analyse et la compréhension de jeux de données.

Le domaine de la visualisation d'information existe depuis le début des années 90. Il s'est détaché de trois domaines connexes : l'interaction Homme-Machine, l'analyse statistique et la cartographie, et la visualisation scientifique. Il est resté proche de ces trois domaines mais a pris son autonomie en s'organisant avec des conférences, des livres de référence, des outils spécifiques ainsi que des modèles et méthodes propres.

Les représentations graphiques ont une histoire longue. Tufte [Tufte86] fait remonter les premières cartes à la chine ancienne, soit 2000 ans avant notre ère.

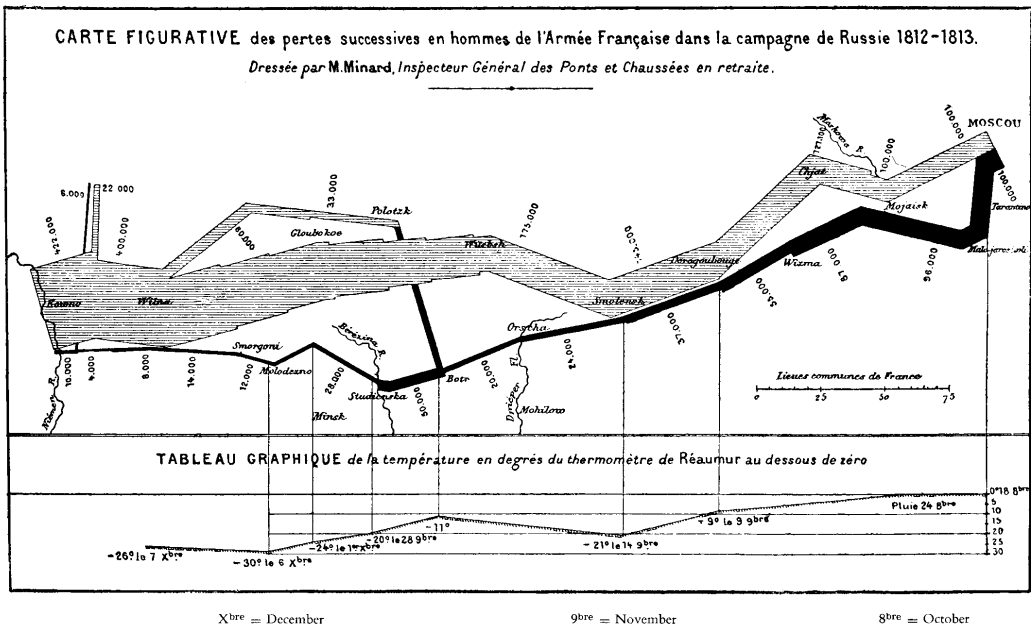


FIG. 2.1 : Carte figurative de Etienne-Jules Minard (1830-1904)

Des représentations de plus en plus abstraites ont été élaborées au cours du temps pour exprimer de manière synthétique et visuelle des informations quantitatives. La figure 2.1 montre un graphique célèbre représentant la campagne de Russie de Napoléon où plusieurs informations sont superposées pour mettre en évidence la taille de l'armée au cours de la campagne en fonction de sa position, ainsi que la température du chemin du retour. Ce diagramme permet de comprendre « d'un coup d'oeil » la situation catastrophique de la retraite de Russie.

Le début des années 90 a vu apparaître la notion de *requêtes dynamiques* [Ahlberg et al.92] permettant d'agir interactivement, continûment et de manière réversible sur des représentations graphiques afin de les filtrer. La combinaison entre les représentations graphiques de données abstraites et le filtrage interactif a décuplé les possibilités d'analyses et d'exploration, aboutissant au nouveau domaine qu'est la visualisation d'information.

Au cours des années 94 à 97, la visualisation d'information est devenu un domaine important de l'interaction Homme-Machine. De nouvelles représentations ont été inventées et expérimentées pour de multiples structures de données : les données temporelles, les tables, les arbres, les graphes, les documents et les programmes et algorithmes pour ne citer qu'eux, dont les contributions principales ont été reprises dans deux monographies importantes [Card et al.99, Stasko et al.97].

Après les années 97, le domaine s'est encore enrichi de techniques multiples et surtout de modèles et méthodes pour évaluer les techniques. Healey a été parmi les premiers chercheurs à faire le lien entre les études sur la vision préattentive [Treisman85] et la visualisation. Le domaine s'est alors enrichi de techniques d'évaluation et de théories pour tenter de prédire les performances d'utilisateurs face à des techniques de visualisation. Nous commençons à comprendre comment notre système perceptif fonctionne et décode des graphiques et comment nous pouvons l'aider. Cependant, nos théories sont encore trop imparfaites pour prévoir les performances humaines ; au mieux, elles expliquent a posteriori ces performances et peuvent éviter des erreurs de codage.

Les recherches en visualisation d'information continuent à progresser dans plusieurs directions : de nouvelles techniques de représentation, de nouvelles techniques d'interaction, la combinaison de plusieurs techniques utilisées simultanément, le passage à l'échelle de techniques connues ou l'étude de techniques d'agrégation pour visualiser de très grosses quantités de données. En plus de ces domaines de recherche, un effort a été récemment porté sur l'évaluation de la visualisation d'information.

2.2 Méthodes d'évaluation pour la visualisation

Depuis une dizaine d'années, le domaine d'où est issue la visualisation d'information — l'interaction Homme-Machine — a déjà mis au point plusieurs méthodes d'évaluation. Ces méthodes sont maintenant bien connues et utilisées régulièrement par les chercheurs du domaine[1]. Elles sont essentiellement issues de la psychologie mais aussi de la sociologie, ainsi que d'autres domaines de l'informatique plus théorique [Dix95]. Le problème des théories et modèles pour l'IHM a reçu, depuis quelques années, une attention croissante : quelques sites pédagogiques ont émergé sur l'Internet tels « Theories in Computer Human Interaction » (TICHI[2]) et plusieurs communications ou ouvrages ont abordé la question [Shneiderman02, Shneiderman et al.04a, Czerwinski04]. Des méthodes de conception et de validation ergonomiques ont essayé d'intégrer la visualisation d'information [Nigay et al.98, Winckler et al.04].

Cependant, ces méthodes ne suffisent pas pour évaluer les techniques de visualisation d'information [Plaisant04]. En effet, la visualisation d'information doit permettre de faire des analyses et des découvertes non triviales et il est difficile de trouver des méthodes pour décider si un système facilite effectivement ces découvertes. Les métriques ne sont pas claires (qu'est-ce qu'une découverte ? Doit-

[1] Le site CHARM `http://www.otal.umd.edu/charm/` décrit les principales méthodes.

[2] `http://www.cs.umd.edu/class/fall2002/cmsc838s/tichi/`.

on mesurer le nombre de découverte par unité de temps et comment ? Comment prendre en compte l'apprentissage nécessaire pour appréhender de nouvelles représentations abstraites ? etc.)

Catherine Plaisant et moi-même avons beaucoup travaillé sur ce problème : nous avons organisé deux ateliers de travail lors des conférences CHI à Minneapolis et à Vienne en Autriche. L'objectif était de trouver des axes de recherche clairs et de poser des méthodologies pour l'évaluation des techniques de visualisation d'information. Nous avons ainsi décidé d'organiser une compétition de visualisation lors de la conférence annuelle InfoVis afin de constituer des benchmarks, c'est-à-dire une méthode pratique pouvant faire avancer le domaine actuellement.

Michel Beaudouin-Lafon et Mountaz Hascoët avaient initiés en 2000 une actions spéciale du CNRS sur l'évaluation des systèmes de visualisation d'information [3]. Nous avons lancé les compétitions de visualisation comme système levier pour pousser les chercheurs et praticiens du domaine à s'investir dans les benchmarks.

L'article [Fekete et al.04], inclus page 91, décrit quelques leçons que nous avons tirées des deux premières compétitions de visualisation. Ces leçons nous serviront à l'organisation de compétition d'interaction que nous allons organiser en collaboration avec la conférence UIST de l'ACM.

La mise à disponibilité de benchmarks dans le domaine de la visualisation d'information et de l'interaction permet enfin de disposer d'informations qualitatives et quantitatives fiables et reproductibles dans le temps sur l'évolution des techniques. Jusqu'à maintenant, ces comparaisons étaient très difficiles à réaliser, faciles à biaiser et d'une faible portée scientifique. Bien entendu, ces benchmarks ne résolvent pas tous les problèmes de l'évaluation, loin s'en faut.

Il est essentiel de continuer à chercher des modèles liés à la perception, à la cognition et à la boucle perception/action nous permettant de prévoir les performances et les caractéristiques techniques de visualisations et d'interactions avant même de les avoir implémentées sur une machine. Hélas, ces modèles sont encore hors d'atteintes et des méthodes empiriques seront encore utiles pendant longtemps.

2.3 Contributions en visualisation d'information

Nos contributions au domaine sont sur plusieurs plans :

- en ce qui concerne les nouvelles techniques de visualisation et d'interaction, nous avons conçu les « Labels excentriques » : un système de représentation dynamique de labels textuels indispensables lorsque la densité d'objets af-

[3] `http://www.lirmm.fr/InfoViz/ASEval/`

fichés ne permet pas un placement de label statique [Fekete et al.99b] (voir l'article page 69) ;
- en ce qui concerne le passage à l'échelle, nous avons conçu et réalisé un système de visualisation d'un très grand nombre d'objets (voir figure 2.2 [Fekete et al.02] ;
- nous avons réalisé des évaluations comparatives de techniques de visualisations pour les graphes [Ghoniem et al.05a] (voir l'article page 91) ;
- nous avons conçu et réalisé une boîte à outils pour la visualisation d'information [Fekete04b] afin de faciliter la mise en œuvre et l'expérimentation de techniques de visualisations ;
- nous avons appliqué la visualisation d'information à des problèmes concrets complexes comme l'analyse et la mise au point de programmes par contraintes [Ghoniem et al.05b, Ghoniem et al.04c], la visualisation de traces d'exécution de programmes, la visualisation du processus d'écriture pour des œuvres littéraires [Crasson et al.04a] ou à l'analyse de sources historiques du XVIe siècle [Fekete et al.00b] ;
- pour améliorer les méthodes d'évaluation comparatives entre les techniques de visualisation, nous avons initié et organisé des compétitions de visualisation d'information et constitué un site de benchmarks (`www.cs.umd.edu/hcil/InfovisRepository`) [Fekete et al.04] (voir l'article page 91).

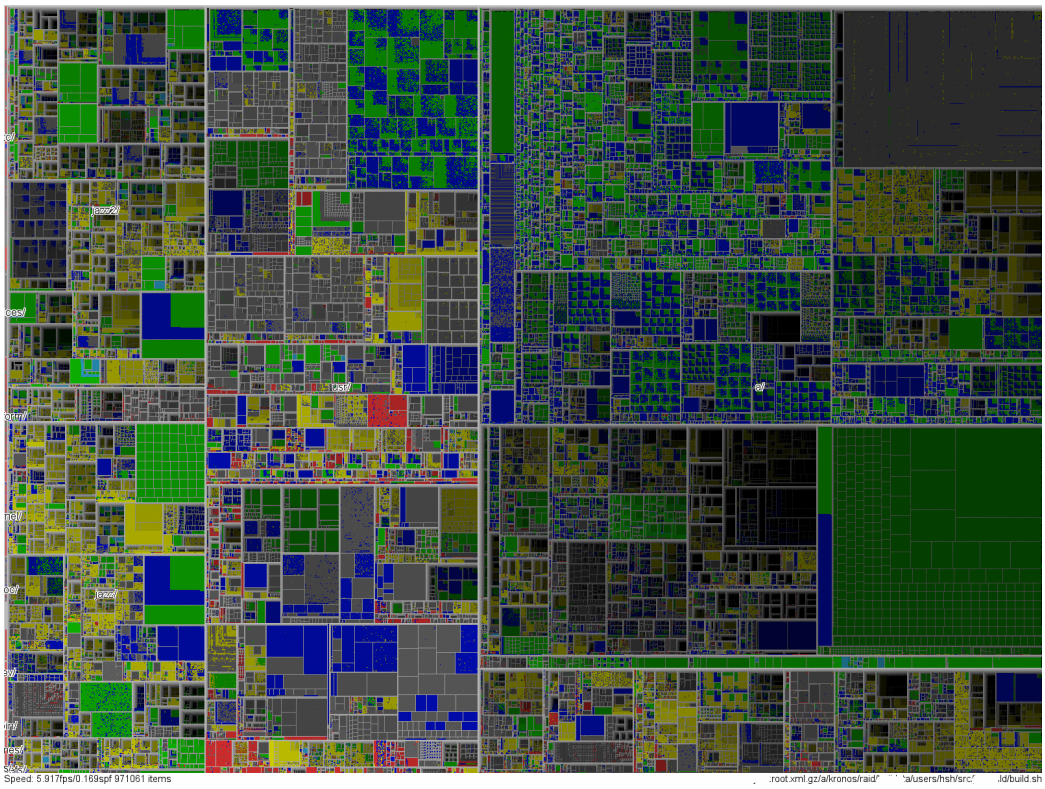


FIG. 2.2 : Visualisation d'un million de fichiers sous forme de Treemap [Fekete et al.02]

Chapitre 3

Comprendre plus, comprendre plus vite, comprendre mieux

Y a-t-il une limite à la compréhension humaine ? La réponse est certainement positive, mais comment connaître cette limite et évaluer si nous en sommes proches ?

Nos travaux ont porté sur l'utilisation de la visualisation d'information pour accélérer et améliorer la compréhension de phénomènes complexes. Dans [Card et al.99], les auteurs citent plusieurs exemples où une représentation externe permet d'améliorer la compréhension d'un phénomène ou la résolution d'un problème. Dans l'introduction de [Glasgow et al.95], Aaron Sloman rappelle les positions fondamentales qu'il a énoncées sur les liaisons entre les zones corticales de la perception visuelle qui sont vraisemblablement « réutilisées » par des fonctions cognitives supérieures pour réaliser des inférences sûres à partir de connexité géométrique : les géomètres se représenteraient des problèmes visuellement pour profiter d'opérations cognitives effectués à un stade très bas de la perception.

Tous ces éléments nous donnent des pistes de recherche et des tentatives d'explications sur l'efficacité de la visualisation pour résoudre des problèmes.

Nos travaux ont porté sur l'utilisation de la visualisation dans trois domaines très différents :

- les sources historiques, pour comprendre les structures récurrentes à partir de sources écrites ;
- la création littéraire, pour comprendre les processus créatifs d'écrivains célèbres,
- les solveurs de contraintes, pour comprendre leur comportement dynamique.

3.1 Comprendre l'histoire à partir des sources

Dans l'article [Fekete et al.00b] (voir page 97), nous décrivons un travail initial sur l'exploitation de sources textuelles historiques. Le travail d'historien, depuis un siècle et demi, consiste principalement à extraire de sources historiques des éléments permettant de chercher des tendances historiques, mettre en évidence des évolutions ou des ruptures, ou collecter des phénomènes intéressant pour une raison ou pour une autre.

Karl Popper a bien expliqué dans son œuvre que l'induction n'existait pas [Popper35]. La construction de théorie ou de modèle se fait à partir de la génération d'une grande quantité d'hypothèses vraisemblable et de son filtrage par la réalité constatée.

Dans notre travail, nous avons essayé de concevoir des techniques de visualisation et d'indexation permettant de vérifier des hypothèses le plus rapidement possible sur un corpus de documents afin d'accélérer le filtrage et ainsi d'augmenter le nombre d'hypothèses validées ou invalidées par unité de temps.

Il est difficile d'évaluer l'efficacité réelle d'un tel système, mais lorsqu'on le compare aux méthodes utilisées aujourd'hui pour dépouiller les documents historiques, basées sur la production de textes sur papier et de grilles d'analyses sur fiches cartonnées, le progrès est incontestable.

3.2 Comprendre l'activité créative littéraire

Les processus créatifs restent un mystère : comment un auteur littéraire construit-il une œuvre à succès ?

Depuis 2002, nous travaillons en collaboration avec l'Institut de Textes Modernes (ITEM CNRS UMR 8132) et la Bibliothèque Nationale de France pour rendre accessible les brouillons d'auteurs littéraires connus comme Gustave Flaubert, Marcel Proust ou Paul Valéry. L'ITEM conduit des recherches à la frontière entre la linguistique et la littérature pour analyser les processus créatifs littéraires : la génétique textuelle.

Pour comprendre les processus de création littéraire, nous avons conçu un modèle permettant de décrire les opérations effectuées concrètement par un auteur sur un texte. En effet, un brouillon manuscrit n'est pas du tout similaire à un texte, comme on peut le voir sur la figure 3.1. Plutôt que de nous reposer sur un modèle linguistique de réécriture, nous avons adopté un modèle simple d'actes d'écritures où un document se décompose en régions graphiques, contenant des marques scripto-graphiques : fragments textuels ou marques graphiques ayant éventuellement une fonction.

En décomposant une œuvre en un ensemble de primitives scripto-graphiques,

il est possible de décrire des feuillets de manière précise sans avoir besoin d'interpréter trop tôt les intentions de l'auteur. Indiquer qu'un fragment textuel a été rayé reste du domaine de la constatation. En revanche, décider que l'auteur a réalisé une *substitution* en rayant d'abord un mot pour le remplacer par un autre mot écrit dans la marge et relié par un trait après le mot rayé est une interprétation de haut niveau qui peut souvent être discutée. Nous préférons donc indiquer les éléments scripto-graphiques constitutifs : le mot initial, la rature, le mot dans la marge et le trait reliant les mot de la marge au mot rayé. C'est le niveau auquel nous décrivons les feuillets manuscrits complexes.

Une fois ce travail initial — cette transcription *haute fidélité* — réalisé, nous pouvons nous intéresser à des phénomènes de plus haut niveau : la structure de l'écriture. Un brouillon d'auteur est modélisable par un graphe hypothétique où des fragments scripto-graphiques sont reliés par au moins trois types de relations : temporelle, *syntagmatique* (la suite des fragments d'une version de l'œuvre) et *paradigmatique* (les réécritures d'un même fragment). Ce graphe est hypothétique car toutes les relations ne sont pas déductibles de la lecture du manuscrit. Ce graphe peut aussi être vu à plusieurs niveaux d'abstraction. Une personne néophyte voudra voir la structure reliant les feuillets entre eux pour pouvoir naviguer dans l'œuvre. Un spécialiste voudra voir les détails de l'évolution d'une section de l'œuvre.

Nous travaillons actuellement sur des outils d'édition et de visualisation permettant de saisir et de représenter le modèle génératif de construction littéraire afin de pouvoir l'analyser. Nous avons décrit en [Crasson et al.04b, Crasson et al.04a] notre approche initiale. Pour la première fois, nous pensons pouvoir donner accès au processus créatif et pas seulement à son expression finale.

Notre objectif est de rendre visible les processus de création et pas uniquement le produit de la création humaine. Comprendre comment un auteur littéraire à construit son œuvre permet de mieux comprendre l'œuvre (bien qu'elle lève des voiles sur des parties cachées délibérément par l'auteur). Par exemple, la figure 3.2 montre comment Aurèle Crasson représente les liens entre des feuillets d'un manuscrit d'Edmond Jabès.

De manière plus générale, la transmission des connaissances se fait presque uniquement sur l'étude des productions humaines : articles de chercheurs ou livres d'auteurs par exemple. Les processus ne sont que très rarement connus.

Et pourtant, le processus est extrêmement important à la fois pour la compréhension des œuvres, mais surtout dans la compréhension des savoir-faire et des processus créatifs ou génératifs et nous pensons qu'une bonne modélisation et des visualisations adaptées sont indispensables à faire avancer ce domaine de recherche.

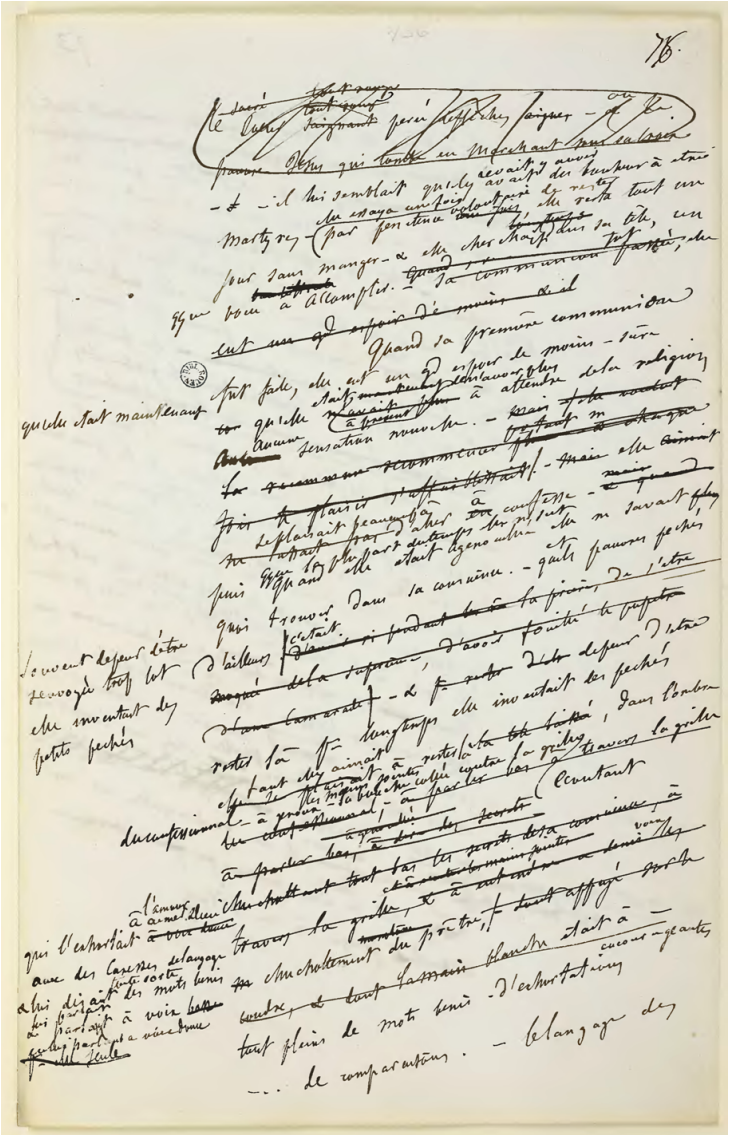


FIG. 3.1 : Page de brouillon de « Madame Bovary » de Gustave Flaubert conservée à la bibliothèque municipale de Rouen

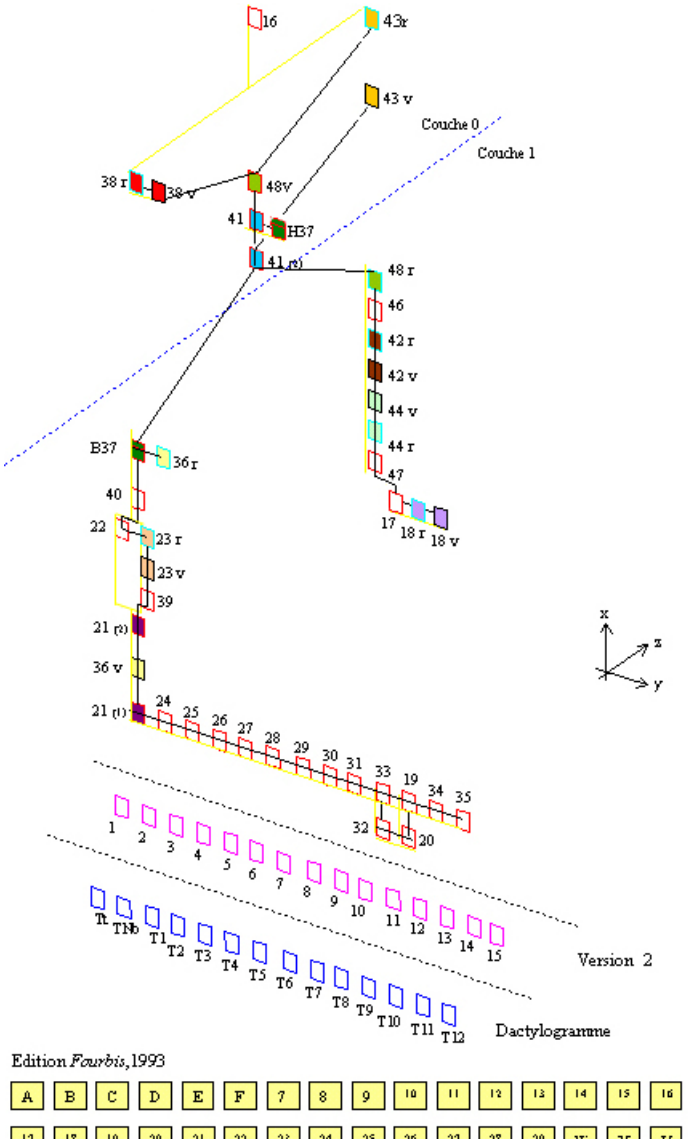


FIG. 3.2 : Structure d'un manuscrit d'Edmond Jabès selon Aurèle Crasson. L'axe paradigmatique est représenté verticalement, l'axe syntagmatique de gauche à droite et le troisième axe concerne une version hypothétique qui n'a pas été retenue par l'auteur

3.3 Comprendre le comportement d'un programme

Nos systèmes informatiques vont de plus en plus vite et ont un comportement de plus en plus complexe. Comment peut-on être sûr qu'ils donnent des résultats corrects et comment pouvons-nous même comprendre ce qu'ils font ? C'est un problème qui prend une importance croissante au fur et à mesure que nous laissons plus de responsabilités aux systèmes automatiques.

Dans le projet OAdymPPaC[1] financé par le Réseau National de recherche et d'innovation en Technologies Logicielles (RNTL), nous avons utilisé des techniques de visualisation d'information pour essayer de rendre visible le comportement de programmes de résolution de contraintes.

Dans [Ghoniem et al.05b] (voir l'article page 107), nous montrons comment la visualisation a aidé à la résolution et à la compréhension du comportement de systèmes très complexes. Ce qui était une « boîte noire » devient compréhensible par des spécialistes. Cette compréhension permet d'améliorer encore le système et d'orienter les chercheurs vers des voies nouvelles.

Nous pensons que la visualisation peut être utilisée efficacement pour un grand nombre de problèmes similaires où des systèmes sont trop complexes pour être compris structurellement mais où leur comportement peut être compris lorsqu'ils est visualisé avec des représentations et des interactions bien choisies. Nous avons commencé à travailler sur le problème de l'analyse des traces de programmes impératifs et l'analyse du comportement interne de microprocesseurs modernes.

[1] `contraintes.inria.fr/OADymPPaC/`

Chapitre 4

Conclusion et perspectives

Il nous reste à continuer à améliorer les modèles, théories et systèmes permettant de mieux tirer parti des capacités combinées de l'Homme et de la Machine pour comprendre plus, comprendre plus vite et comprendre mieux.

Nous avons toujours essayé de mener de front trois objectifs concurrents : améliorer les modèles existants ou concevoir de nouveaux modèles conceptuels, montrer qu'ils étaient adéquats dans des contextes d'usage réels et faciliter la réutilisation des outils que nous avons réalisés. Nous voulons garder ce mode de fonctionnement qui paraît efficace pour faire avancer nos recherches.

La visualisation d'information est un domaine qui a fait ses preuves et a montré un potentiel extrêmement prometteur. Nous voulons continuer à l'explorer avec toujours la même méthode : chercher de nouveaux modèles (qui manquent encore cruellement), les appliquer à des domaines intéressants et complexes, et faciliter l'évolution du domaine.

Dans les années à venir, nous voulons renforcer nos axes de recherche sur l'architecture des systèmes interactifs, la visualisation d'information et l'amélioration des méthodes d'évaluation en IHM et en visualisation d'information. Nous pensons que les champs d'applications de la nouvelle génération d'interfaces Homme-Machine sur laquelle nous avons travaillé et nous voulons continuer à travailler n'ont été qu'à peine effleurés et qu'ils nous offrent de vastes perspectives de recherche touchant à des disciplines variées et riches telles l'informatique, la psychologie cognitive et les sciences humaines.

4.1 Perspectives en architecture des systèmes interactifs

L'utilisation de formalismes et architectures logicielles basées sur des systèmes réactifs pour les entrées et sur des graphes de scène pour la sortie semble

une voie prometteuse. Il nous faut encore articuler convenablement les graphes mixtes et c'est un des axes sur lesquels nous voulons continuer à travailler.

Nous avons déjà proposé un modèle de graphe mixte dans [Huot et al.04] mais nous voudrions l'étendre dans plusieurs directions :

- la séparation de la gestion d'entrée de bas niveau et des styles d'interaction ;
- la possibilité de gérer l'interaction haptique ou selon des modalités multiples en entrée comme en sortie.

La définition des styles d'interaction directement à partir d'un formalisme réactif est possible mais peu élégant. Nous voudrions que les styles d'interaction soient des objets de premier ordre afin de pouvoir spécifier des applications graphiques interactives à partir de combinaison de haut niveau : interaction de style classique (WIMP) ou basé sur la trace (Ink-based) ou immergée, ou encore basée sur fusion parole-geste. Cette séparation devrait permettre la spécification claire de la notion de *paradigme d'interaction* qui reste aujourd'hui allusif. Avec la prolifération actuelle des plateformes d'interaction, cette étude devrait faciliter la réutilisation d'application interactives profondément adaptées aux plateformes.

De manière plus générale, nous voudrions unifier les modèles architecturaux multiples utilisés dans les domaines connexes à l'interaction Homme-Machine comme l'informatique graphique et la réalité virtuelle, la multimodalité basée sur la reconnaissance vocale et gestuelle, ainsi que l'informatique enfouie (*Ubiquitous computing*). Tous ces domaines de recherche et d'applications utilisent des modèles architecturaux légèrement différents principalement pour des raisons historiques. Proposer une unification élargirait encore la compréhension de l'interaction et faciliterait la recherche de nouveaux paradigmes et la mise en œuvre d'applications reposant sur ces paradigmes.

4.2 Perspectives pour la visualisation d'information

Dans le domaine de la visualisation d'information, notre travail va s'orienter vers les études de méthodes et modèles pour améliorer la lisibilité et l'utilisabilité. Nous avons identifié trois axes sur lesquels nous voulons nous concentrer :

- les infrastructures logicielles pour faciliter l'expérimentation et l'évaluation des techniques de visualisation ;
- le développement de *benchmarks* pour faciliter la comparaison de systèmes de visualisation ;
- le raffinement des métriques pour l'évaluation des techniques de visualisation.

Nous avons déjà travaillé sur des boîtes à outils pour faciliter la conception et réalisation de nouvelles techniques de visualisation et d'interaction, ainsi que leur déploiement [Fekete04b, Fekete et al.02]. Nous voudrions étendre ces boîtes

à outils pour faciliter l'évaluation des techniques de visualisation :

- permettre la capture de sessions pour permettre une analyse sur l'utilisation longitudinale de techniques de visualisation dans des domaines particuliers ;
- offrir des squelettes d'applications d'évaluations comparatives afin de faciliter et de rationaliser les évaluations empiriques de techniques de visualisation.

Bien entendu, ces outils d'évaluation ne sont pas une fin en soi ; ils sont nécessaires à l'avancement du domaine de la visualisation et faciliteront les tests systématiques, par exemple de nouvelles technique de visualisation ou d'interaction appliquées à des benchmarks existants. Il faciliteront aussi la vérification de modèles cognitifs ou perceptifs qui sont aujourd'hui naissant et qui nécessitent un raffinement à l'avenir afin de les rendre plus fiables ou prédictifs.

Concernant les métriques pour l'évaluation de la visualisation, nous sommes actuellement limités. La plupart des évaluations utilisent deux métriques : la vitesse et le nombre d'erreurs. Nous pensons que d'autres métriques seraient plus intéressantes mais elles sont difficiles à caractériser. Par exemple, Geoges Robertson de Microsoft Research voudrait mesurer l'utilité alors que nous savons mesurer l'utilisabilité. Chris North de Virginia Tech. voudrait mesurer le nombre de « découvertes » (*insights*) par unité de temps et catégoriser ces découvertes.

Nous pensons que les méthodes appliquées à l'évaluation des compétences de lecture en pédagogie peuvent nous inspirer pour réaliser des tests de compétences de lecture de visualisations [Rastier89, DL78, Oecd et al.03]. Les tests de lecture ont été étudiés depuis longtemps par les pédagogues désireux de mesurer les progrès individuels ou collectifs de l'alphabétisation. Nous constatons que la visualisation d'information nécessite un apprentissage et requiert plusieurs niveaux de compréhension pour être efficace. À notre connaissance, très peu de travaux ont été fait dans cette direction et nous voudrions nous y pencher. Il s'agit de distinguer plusieurs niveaux de lecture et des tests calibrés pour comprendre comment des sujets comprennent une visualisation en fonction de leur familiarité à la visualisation ou au phénomène visualisé, ou à l'inverse d'évaluer des visualisations en fonction du temps ou de la difficulté pour arriver à certains niveaux de compréhension.

Nous pensons que ces nouvelles méthodes permettront au domaine de la visualisation d'information d'évoluer vers de techniques plus efficaces et vers une adoption de la visualisation d'information par les fabricants de logiciels et leurs utilisateurs.

Bibliographie

[Ahlberg et al.92] Christopher Ahlberg, Christopher Williamson et Ben Shneiderman. – Dynamic queries for information exploration : an implementation and evaluation. *CHI '92 : Proceedings of the SIGCHI conference on Human factors in computing systems*. pp. 619–626. – ACM Press, 1992.

[Andre et al.95a] Jacques André, Jean-Daniel Fekete et Hélène Richy. – Traitement mixte image/texte de documents anciens. *Actes du congrès GUTenberg, La Grande Motte*, pp. 75–85. – juin 1995.

[Andre et al.95b] Jacques André, Hélène Richy et Jean-Daniel Fekete. – Editing tools for WWWing ancient texts. – Position paper for the ERCIM/W4G Int. Workshop on WWW Authoring and Integration Tools, février 1995.

[Auber01] David Auber. – Tulip. *9th Symposium on Graph Drawing, GD 2001*, éd. par Sebastian Leipert Petra Mutzel, Mickael Jünger, pp. 335–337. – Vienna, Austria, 2001.

[Baudisch et al.04] Patrick Baudisch et Carl Gutwin. – Multiblending : displaying overlapping windows simultaneously without the drawbacks of alpha blending. *CHI '04 : Proceedings of the 2004 conference on Human factors in computing systems*. pp. 367–374. – ACM Press, 2004.

[Bederson et al.00] Benjamin B. Bederson, Jon Meyer et Lance Good. – Jazz : an extensible zoomable user interface graphics toolkit in java. *UIST '00 : Proceedings of the 13th annual ACM symposium on User interface software and technology*. pp. 171–180. – ACM Press, 2000.

[Bederson et al.04] Benjamin B. Bederson, Jesse Grosjean et Jon Meyer. – Toolkit design for interactive structured graphics. *Transactions on Software Engineering*, vol. 30, n8, 2004, pp. 535–546.

[Berry et al.85] Gérard Berry et Laurent Cosserat. – The esterel synchronous programming language and its mathematical semantics. *Seminar on Concurrency, Carnegie-Mellon University*. pp. 389–448. – Springer-Verlag, 1985.

[BL et al.91] Michel Beaudouin-Lafon, Yves Berteaud, Stéphane Chatty, Jean-Daniel Fekete et Thomas Baudel. – *The X Television – C++ Library for*

Direct Manipulation Interfaces. – Technical report, LRI, Université de Paris-Sud, France, février 1991. version 2.0.

[BL00] Michel Beaudouin-Lafon. – Instrumental interaction : an interaction model for designing post-WIMP user interfaces. *Proc. ACM Conference on Human Factors in Computing Systems (CHI 2000)*. pp. 446–453. – The Hague, The Netherlands, 2000.

[Blanch02] Renaud Blanch. – Programmer l'interaction avec des machines à états hiérarchiques. *IHM '02 : Proceedings of the 14th French-speaking conference on Human-computer interaction (Conférence Francophone sur l'Interaction Homme-Machine)*. pp. 129–136. – ACM Press, 2002.

[Brun et al.95] Philippe Brun et Michel Beaudouin-Lafon. – A taxonomy and evaluation of formalisms for the specification of interactive systems. *Conference on Human-Computer Interaction, HCI'95*. BCS-HCI, pp. 197–212. – Cambridge University Press, août 1995.

[Calder et al.90] Paul R. Calder et Mark A. Linton. – Glyphs : Flyweight Objects for User Interfaces. *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, pp. 92–101. – 1990.

[Card et al.83] Stuart K. Card, T.P. Moran et Alan P. Newell. – *The Psychology of Human-Computer Interaction*. – Hillsdale, NJ, Erlbaum, 1983.

[Card et al.99] Stuart K. Card, Jock D. Mackinlay et Ben Shneiderman (édité par). – *Readings in information visualization : using vision to think*. – Morgan Kaufmann Publishers Inc., 1999.

[Chatty et al.04] Stéphane Chatty, Stéphane Sire, Jean-Luc Vinot, Patrick Lecoanet, Alexandre Lemort et Christophe Mertz. – Revisiting visual interface programming : creating gui tools for designers and programmers. *UIST '04 : Proceedings of the 17th annual ACM symposium on User interface software and technology*. pp. 267–276. – ACM Press, 2004.

[Consel et al.89] Charles Consel, Alain Deutsch, Renaud Dumeur et Jean-Daniel Fekete. – CSKIM : An extended dialect of scheme. *Bigre Buulletin*, juillet 1989.

[Conversy et al.02] Stéphane Conversy et Jean-Daniel Fekete. – *The svgl toolkit : enabling fast rendering of rich 2D graphics*. – Rapport technique n 02/1/INFO, École des Mines de Nantes, janvier 2002.

[Coutaz87] Joöelle Coutaz. – PAC, an implementation model for dialog design. *Proceedings of INTERACT'87*, pp. 431–436. – septembre 1987.

[Crasson et al.04a] Aurèle Crasson et Jean-Daniel Fekete. – Du corpus à la région : Représentations multi-échelles pour les manuscrits modernes. *In Linguistica Computazionale, Digital Technology and Philological Disciplines*, vol. XX-XXI, octobre 2004, pp. 111–128.

[Crasson et al.04b] Aurèle Crasson et Jean-Daniel Fekete. – Structuration des manuscrits : Du corpus à la région. *Proceedings of CIFED 2004*, pp. 162–168. – La Rochelle, France, juin 2004.

[Czerwinski04] Mary Czerwinski. – Bridging the gap from theory to practice : the path toward innovation in human-computer interaction. *UIST '04 : Proceedings of the 17th annual ACM symposium on User interface software and technology*. pp. 1–1. – ACM Press, 2004.

[Dix95] Alan J. Dix. – *Perspectives on HCI : Diverse Approaches*, chap. Formal Methods, pp. 9–43. – Academic Press, 1995.

[DL78] Gilbert De Landsheere. – *Le test de closure : mesure de la lisibilité et de la compréhension*. – Labor-Nathan, 1978.

[Dragicevic et al.99] Pierre Dragicevic et Jean-Daniel Fekete. – étude d'une boîte à outils multi-dispositifs. *Actes de la 11e conférence francophone sur l'Interaction Homme-Machine (IHM 99)*. pp. 55–62. – Cepadues, novembre 1999.

[Dragicevic et al.01] Pierre Dragicevic et Jean-Daniel Fekete. – Input device selection and interaction configuration with icon. *Proceedings of the International Conference IHM-HCI 2001*, éd. par A. Blandford, J. Vanderdonckt et P. Gray. pp. 543–448. – Lille, France, 2001.

[Dragicevic et al.04a] Pierre Dragicevic et Jean-Daniel Fekete. – The input configurator toolkit : Towards high input adaptability in interactive applications. *Proc. Conference on Advanced Visual Interfaces, AVI 2004*. pp. 244–247. – Gallipoli (Lecce), Italy, mai 2004.

[Dragicevic et al.04b] Pierre Dragicevic et Jean-Daniel Fekete. – Support for input adaptability in the icon toolkit. *Proceedings of the Sixth International Conference on Multimodal Interfaces (ICMI 2004)*. p. in press. – Penn State University, PA, octobre 2004.

[Dragicevic04] Pierre Dragicevic. – *Un modèle de configurations d'entrée pour des systèmes interactifs multi-dispositifs hautement configurables*. – Thèse de PhD, Université de Nantes, mars 2004.

[Fekete et al.89] Jean-Daniel Fekete, Benoît Hap et Renaud Dumeur. – GENESE : Narrowing the gap between experts and systems. *Proceedings of the IEEE Engineering in Medicine & Biology Society Conference*. – novembre 1989.

[Fekete et al.95] Jean-Daniel Fekete, Érick Bizouarn, Éric Cournarie, Thierry Galas et Frédéric Taillefer. – Tictactoon : A paperless system for professional 2d animation. *Computer Graphics (SIGGRAPH' 95 Proceedings)*, pp. 79–90. – août 1995.

[Fekete et al.96] Jean-Daniel Fekete et Michel Beaudouin-Lafon. – Using the multi-layer model for building interactive graphical applications. *Proceedings of the UIST'96 Conf., Seattle, USA*, pp. 109–118. – novembre 1996.

[Fekete et al.98a] Jean-Daniel Fekete et Nicole Dufournaud. – Utilisation de TEI comme support méthodologique au dépouillement de sources manuscrites. *Actes du Vième Colloque National de L'Association Française pour l'Histoire et l'Informatique (AHI)*. – novembre 1998.

[Fekete et al.98b] Jean-Daniel Fekete et Martin Richard. – *Esterel meets Java : Building Reactive Synchronous Systems with Java*. – Rapport technique n98-3-info, 4 rue Alfred Kastler, La Chantrerie, 44307 Nantes Cedex, Ecole des Mines de Nantes, 1998.

[Fekete et al.98c] Jean-Daniel Fekete, Martin Richard et Pierre Dragicevic. – Specification and verification of interactors : A tour of esterel. *Formal Aspects of Human Computer Interaction Workshop (FAHCI'98)*. Sheffield Hallam University. – Sheffield, UK, septembre 1998.

[Fekete et al.99a] Jean-Daniel Fekete et Nicole Dufournaud. – Analyse historique de sources manuscrites : application de TEI à un corpus de lettres de rémission du xvie siècle. *Document Numérique, numéro spécial "Les documents anciens",*, vol. 3, n1–2, 1999, pp. 117–134.

[Fekete et al.99b] Jean-Daniel Fekete et Catherine Plaisant. – Excentric labeling : dynamic neighborhood labeling for data visualization. *Proceedings of the SIGCHI conference on Human factors in computing systems*. pp. 512–519. – ACM Press, 1999.

[Fekete et al.00a] Jean-Daniel Fekete et Pierre Dragicevic. – Une architecture multi-dispositifs. *Actes du Colloque sur les Interfaces Multimodales*. – Grenoble, France, mai 2000.

[Fekete et al.00b] Jean-Daniel Fekete et Nicole Dufournaud. – Compus : visualization and analysis of structured documents for understanding social life in the 16th century. *Proceedings of the fifth ACM conference on Digital libraries*. pp. 47–55. – ACM Press, 2000.

[Fekete et al.00c] Jean-Daniel Fekete et David Salesin (édité par). – *NPAR '00 : Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*. – ACM Press, 2000.

[Fekete et al.01] Jean-Daniel Fekete et Patrick Girard. – *Systèmes d'information et Interaction Homme-Machine*, chap. Environnements de développement de systèmes interactifs, pp. 23–48. – Hermes, 2001.

[Fekete et al.02] Jean-Daniel Fekete et Catherine Plaisant. – Interactive information visualization of a million items. *Proc. IEEE Symposium on Information*

Visualization 2002 (InfoVis 2002). pp. 117–124. – Boston, USA, octobre 2002.

[Fekete et al.03] Jean-Daniel Fekete, David Wang, Niem Dang et Catherine Plaisant. – Overlaying graph links on treemaps. – IEEE Symposium on Information Visualization Conference Compendium (demonstration), octobre 2003.

[Fekete et al.04] Jean-Daniel Fekete et Catherine Plaisant. – Les leçons tirées des deux compétitions de visualisation d'information. *Proceedings of IHM 2004*. pp. 7–12. – Namur, Belgium, septembre 2004.

[Fekete91] Jean-Daniel Fekete. – Un modèle abstrait pour la construction d'editeurs graphiques interactifs. *Proc. Troisième journées sur l'Ingénierie des Interfaces Homme-Machine (IHM)*. – 1991.

[Fekete92] Jean-Daniel Fekete. – A multi-layer graphic model for building interactive graphical applications. *Graphics Interface'92, Vancouver, Canada*, pp. 294–300. – mai 1992.

[Fekete96a] Jean-Daniel Fekete. – Les trois services du noyau sémantique indispensables à l'ihm. *Actes des 8es journées sur 'ingénierie des l'Interaction Homme-Machine (IHM 96)*, pp. 45–50. – Grenoble, France, septembre 1996.

[Fekete96b] Jean-Daniel Fekete. – Les trois services du noyau sémantique indispensables à l'ihm. *Séance pleinière des journées du GDR Programmation du CNRS, Orléans*. – novembre 1996.

[Fekete96c] Jean-Daniel Fekete. – *Un modèle multicouche pour la construction d'applications graphiques interactives*. – Thèse de PhD, Université d'Orsay Paris-Sud, janvier 1996.

[Fekete98a] Jean-Daniel Fekete (édité par). – *Actes des 10ièmes journées francophones d'Interaction Homme Machine (IHM 98)*. Association Francophone d'Interaction Homme Machine (AFIHM). – Cepadues, septembre 1998, viii+149p.

[Fekete98b] Jean-Daniel Fekete. – Expérience de codage de document à intérêt graphique à l'aide de TEI. *Actes du congrès Eurotex 98*, pp. 131–142. – Saint Malo, mars 1998.

[Fekete02] Jean-Daniel Fekete (édité par). – *UIST '02 : Proceedings of the 15th annual ACM symposium on User interface software and technology*. – ACM Press, 2002. Conference Chair-Michel Beaudouin-Lafon.

[Fekete03a] Jean-Daniel Fekete. – *The Infovis Toolkit*. – Rapport technique n RR-4818, INRIA Futurs, mai 2003.

[Fekete03b] Jean-Daniel Fekete. – The infovis toolkit. – IEEE Symposium on Information Visualization Conference Compendium (demonstration), octobre 2003.

[Fekete03c] Jean-Daniel Fekete. – The infovis toolkit. – ACM Symposium on User Interface Software and Technology (UIST 2002) Conference Compendium, novembre 2003.

[Fekete04a] Jean-Daniel Fekete (édité par). – *Actes de la 16e Conférence sur l'Interaction Homme-Machine (IHM 2004)*, AFIHM. – Namur, Belgium, ACM Press, septembre 2004.

[Fekete04b] Jean-Daniel Fekete. – The infovis toolkit. *Proceedings of the 10th IEEE Symposium on Information Visualization (InfoVis 04)*. pp. 167–174. – Austin, TX, octobre 2004.

[Fekete04c] Jean-Daniel Fekete. – La boîte à outils infovis. *Proceedings of IHM 2004*. pp. 69–76. – Namur, Belgium, septembre 2004.

[Fekete04d] Jean-Daniel Fekete. – Le format pdf. *Techniques de l'ingénieur*. – Techniques de l'ingénieur, novembre 2004.

[Fekete04e] Jean-Daniel Fekete. – Le langage postscript. *Techniques de l'ingénieur*. – Techniques de l'ingénieur, novembre 2004.

[Ferraiolo et al.03] Jon Ferraiolo, Jun Fujisawa et Dean Jackson. – *Scalable Vector Graphics (SVG) 1.1 Specification*. – Rapport technique, W3C, janvier 2003.

[Fitts54] Paul M. Fitts. – The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, vol. 47, n6, June 1954, pp. 381–391.

[Gamma et al.94] Erich Gamma, Richard Helm, Ralph Johnson et John Vlissides. – *Design Patterns*. – Reading, MA, USA, Addison-Wesley, 1994.

[Gangnet et al.94] Michel Gangnet, Jean-Manuel Van Thong et Jean-Daniel Fekete. – Automated gap closing for freehand drawing. *Siggraph'94 Technical Sketch, Orlando*. – 1994.

[Geary et al.97] David M. Geary et Alan L. McClellan. – *Graphic Java : mastering the AWT*. – 2550 Garcia Avenue, Mountain View, CA 94043-1100, USA, SunSoft Press, 1997, second édition, *SunSoft Press Java series Java series (Mountain View, CA)*, 900p.

[Ghoniem et al.01] Mohammad Ghoniem et Jean-Daniel Fekete. – Animating treemaps. *Workshop on Treemap Implementations and Applications*. University of Maryland. – College Park, mai 2001.

[Ghoniem et al.02] Mohammad Ghoniem et Jean-Daniel Fekete. – Visualisation de graphes de co-activité par matrices d'adjacence. *Actes 14ème conférence francophone sur l'Interaction Homme-Machine (IHM 2002)*. pp. 279–282. – Poitiers, France, octobre 2002.

[Ghoniem et al.03a] Mohammad Ghoniem et Jean-Daniel Fekete. – Visualisation matricielle des graphes et manipulation directe de hiérarchies de clusters (demonstration). *Actes 15ème conférence francophone sur l'Interaction Homme-Machine (IHM 2003)*. pp. 206–207. – Caen, France, novembre 2003.

[Ghoniem et al.03b] Mohammad Ghoniem, Narendra Jussien et Jean-Daniel Fekete. – Visualizing explanations to exhibit dynamic structure in constraint problems. *Proc. 3th International Workshop on User Interaction in Constraint Satisfaction*. – Kinsale, Ireland, septembre 2003.

[Ghoniem et al.04a] Mohammad Ghoniem et Jean-Daniel Fekete. – Comparaison de la lisibilité des graphes en représentation noeuds-liens et matricielle. *Proceedings of IHM 2004*. pp. 77–84. – Namur, Belgium, septembre 2004.

[Ghoniem et al.04b] Mohammad Ghoniem, Jean-Daniel Fekete et Philippe Castagliola. – A comparison of the readability of graphs using node-link and matrix-based representations. *Proceedings of the 10th IEEE Symposium on Information Visualization (InfoVis 04)*. pp. 17–24. – Austin, TX, octobre 2004.

[Ghoniem et al.04c] Mohammad Ghoniem, Narendra Jussien et Jean-Daniel Fekete. – Visexp : visualizing constraint solver dynamics using explanations. *Proc. of the 17th International Conference of the Florida Artificial Intelligence Research Society (FLAIRS 2004)*. – AAAI Press, mai 2004.

[Ghoniem et al.05a] Mohammad Ghoniem, Jean-Daniel Fekete et Philippe Castagliola. – Readability of graphs using node-link and matrix-based representations : Controlled experiment and statistical analysis. *Information Visualization Journal*, vol. 4, 2005, p. à paraître.

[Ghoniem et al.05b] Mohammad Ghoniem, Jean-Daniel Fekete, Narendra Jussien et Hadrien Cambazard. – Peeking in solver strategies using explanations visualization of dynamic graphs for constraint programming. *Proceedings of the ACM Symposium of Software Visualization (SoftVis'05)*. p. à paraître. – ACM Press, mai 2005.

[Glasgow et al.95] Janice Glasgow, N. Hari Narayanan et B. Chandrasekaran (édité par). – *Diagrammatic Reasoning : Cognitive and Computational Perspectives*. – MIT Press, 1995.

[Goodman87] Danny Goodman. – *The Complete HyperCard Handbook*. – Bantam, 1987.

[Gossett et al.04] Nathan Gossett et Baoquan Chen. – Paint inspired color mixing and compositing for visualization. *INFOVIS '04 : Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'04)*. pp. 113–118. – IEEE Computer Society, 2004.

[Greenberg et al.02] Saul Greenberg et Michael Boyle. – Customizable physical interfaces for interacting with conventional applications. *UIST '02 : Proceedings of the 15th annual ACM symposium on User interface software and technology*. pp. 31–40. – ACM Press, 2002.

[Heer et al.05] Jeffrey Heer, Stuart K. Card, et James A. Landay. – prefuse : a toolkit for interactive information visualization. *CHI 2005, Human Factors in Computing Systems*. p. to appear. – ACM PRess, avril 2005.

[Huot et al.04] Stéphane Huot, Cédric Dumas, Pierre Dragicevic, Jean-Daniel Fekete et Gérard Hégron. – The MaggLite Post-WIMP Toolkit : Draw It, Connect It and Run It. *Proceedings of ACM Symposium on User Interface Software and Technology (UIST 2004)*. pp. 257–266. – Santa Fe, NM, octobre 2004.

[Huot04] Stéphane Huot. – Svalabard : une table à dessin virtuelle pour la modélisation 3d. *Proceedings of IHM 2004*, éd. par Jean-Daniel Fekete. AFIHM. – Namur, Belgium, septembre 2004.

[Lakatos95] Imre Lakatos. – *The methodology of scientific research programmes*. – Cambridge University Press, 1995, *Philosophical papers, volume I, edited by John Worrall and Gregory Currie*.

[Lecolinet03] Eric Lecolinet. – A molecular architecture for creating advanced guis. *UIST '03 : Proceedings of the 16th annual ACM symposium on User interface software and technology*. pp. 135–144. – ACM Press, 2003.

[Lewis et al.95] Ted Lewis, Larry Rosenstein, Wolfgang Pree, Andre Weinand, Erich Gamma, Paul Calder, Glenn Andert, John Vlissides et Kurt Schmucker. – *Object-Oriented Application Frameworks*. – Prentice-Hall, 1995.

[Lorenz et al.01] David H. Lorenz et John Vlissides. – Designing components versus objects : A transformational approach. *Proceedings of the 23rd International Conference on Software Engineering*. pp. 253–262. – IEEE Computer Society, 2001.

[Martin perandres et al.02] Domingo Martin Perandres, Jean-Daniel Fekete et Juan Carlos Torres Cantero. – Flattening 3D objects using silhouettes. *Proc. Eurographics 2002*. – Saarbrücken, Germany, septembre 2002.

[McCormack88] Joel McCormack. – An overview of the X toolkit. *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software*, 1988, pp. 46–55.

[Moore65] Gordon E. Moore. – Cramming more components onto integrated circuits. *Electronics*, April 19 1965.

[Myers et al.90] Brad A. Myers, Dario Giuse et Roger Dannenberg et al. – GARNET comprehensive support for graphical, highly interactive user interfaces. *COMPUTER magazine*, novembre 1990.

[Myers91] Brad A. Myers. – Separating Application Code from Toolkits : Eliminating the Spaghetti of Call-Backs. *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, pp. 211–220. – 1991.

[Myers95] Brad A. Myers. – The garnet and amulet user interface development environments. *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, p. 334. – 1995.

[Nigay et al.98] Laurence Nigay et Frédéric Vernier. – Design method of interaction techniques for large information spaces. *In proceedings of AVI'98*, pp. 37–46. – L'aquilla, Italy, 1998.

[Oecd et al.03] OECD et UNESCO. – *Literacy Skills for the World of Tomorrow : Further Results from PISA 2000*. – Rapport technique, OECD and UNESCO INSTITUTE FOR STATISTICS (UIS), 2003.

[Open Inventor Architecture Group94] Open Inventor Architecture Group. – *Open Inventor C++ Reference Manual : The Official Reference Document for Open Systems*. – Addison Wesley, 1994, vi + 767p.

[Opengl architecture review board et al.03] OpenGL Architecture Review Board, Dave Shreiner, Mason Woo, Jackie Neider, Tom Davis, OpenGL Architecture Review Board et David Shreiner. – *OpenGL Programming Guide : The Official Guide to Learning OpenGL, Version 1.4*. – Addison Wesley, novembre 2003.

[Ousterhout94] John K. Ousterhout. – *Tcl and the Tk Toolkit*. – Reading, MA, USA, Addison-Wesley, 1994, xx + 458p.

[Pietriga02] Emmanuel Pietriga. – *Environnements et langages de programmation visuels pour le traitement de documents structurés*. – Thèse de PhD, INPG, Grenoble, novembre 2002.

[Plaisant04] Catherine Plaisant. – The challenge of information visualization evaluation. *AVI '04 : Proceedings of the working conference on Advanced visual interfaces*. pp. 109–116. – ACM Press, 2004.

[Plénacoste et al.01] Patricia Plénacoste, Eric Lecolinet, Stuart Pook, Cédric Dumas et Jean-Daniel Fekete. – Bibliothèques : comparaisons entre le réel et le virtuel en 3d, 2d zoomable et 2d arborescent. – video proceedings of IHM-HCI 2001, Lille, France, septembre 2001.

[Popper35] Karl Popper. – *Logik der Forschung*. – Julius Springer Verlag, 1935.

[Rastier89] François Rastier. – *Sens et Textualité*. – Hachette, 1989.

[Schmucker87] Kurt J. Schmucker. – *Readings in Human-Computer Interaction : A Multidisciplinary Approach*, chap. MacApp : An Application Framework,

pp. 591–594. – Los Altos, CA 94022, Morgan-Kaufmann Publishers Inc., 1987.

[Shneiderman et al.04a] Ben Shneiderman et Catherine Plaisant. – *Designing the User Interface : Strategies for Effective Human-Computer Interaction*. – Addison Wesley, mai 2004, 4th édition.

[Shneiderman et al.04b] Ben Shneiderman, Catherine Plaisant et Jean-Daniel Fekete. – *Designing the User Interface*, chap. Software Tools. – Addison Wesley, mai 2004, 4th édition.

[Shneiderman02] Ben Shneiderman. – *Leonardo's Laptop : Human Needs and the New Computing Technologies*. – MIT Press, octobre 2002.

[Spence et al.93] I. Spence et R. F. Garrison. – A remarkable scatterplot. *The American Statistician*, 1993, pp. 12–19.

[Stasko et al.97] John T. Stasko, Marc H. Brown et Blaine A. Price (édité par). – *Software Visualization*. – MIT Press, 1997.

[Szekely et al.93] Pedro Szekely, Ping Luo et Robert Neches. – Beyond interface builders : model-based interface tools. *CHI '93 : Proceedings of the SIGCHI conference on Human factors in computing systems*. pp. 383–390. – ACM Press, 1993.

[Tesler81] Larry Tesler. – The smalltalk environment. *Byte*, vol. 6, n8, août 1981, pp. 90–147.

[Thevenin et al.99] David Thévenin et Joëlle Coutaz. – Plasticity of user interfaces : Framework and research agenda. *Proc. of IFIP TC 13 Int. Conf. on Human-Computer Interaction (Interact'99)*, éd. par A. Sasse et Chris Johnson. IFIP, pp. 110–117. – London, août 1999.

[Thompson et al.00] D. A. Thompson et J. S. Best. – The future of magnetic data storage technology. *IBM Journal of Research and Development*, vol. 44, n3, mai 2000, pp. 311–321.

[Treisman85] Anne Treisman. – Preattentive processing in vision. *Comput. Vision Graph. Image Process.*, vol. 31, n2, 1985, pp. 156–177.

[Tufte86] Edward R. Tufte. – *The visual display of quantitative information*. – Graphics Press, 1986.

[Vlissides et al.89] John M. Vlissides et Mark A. Linton. – Unidraw : A Framework for Building Domain-Specific Graphical Editors. *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, pp. 158–167. – 1989.

[Weinand et al.88] Andre Weinand, Erich Gamma et Rudolf Marty. – ET++an object oriented application framework in C++. *ACM SIGPLAN Notices*, vol. 23, n11, novembre 1988, pp. 46–57.

[Winckler et al.04] Marco Winckler, Philippe A. Palanque et Carla Maria Dal Sasso Freitas. – Tasks and scenario-based evaluation of information visualization techniques. *TAMODIA*, pp. 165–172. – 2004.

Annexes

Input Device Selection and Interaction Configuration with ICON

Pierre Dragicevic
Ecole des Mines de Nantes
4, rue Alfred Kastler, La Chantrerie
44307 Nantes Cedex, France

Pierre.Dragicevic@emn.fr

Jean-Daniel Fekete
Ecole des Mines de Nantes
4, rue Alfred Kastler, La Chantrerie
44307 Nantes Cedex, France
+33 2 51858208
Jean-Daniel.Fekete@emn.fr

ABSTRACT
This paper describes ICON, a novel editor designed to configure a set of input devices and connect them to actions into a graphical interactive application. ICON allows physically challenged users to connect alternative input devices and/or configure their interaction techniques according to their needs. It allows skilled users – graphic designers or musicians for example – to configure any ICON aware application to use their favorite input devices and interaction techniques (bimanual, voice enabled, etc.).

ICON works with Java Swing and requires applications to describe their interaction styles in terms of ICON modules. By using ICON, users can adapt more deeply than before their applications and programmers can easily provide extensibility to their applications.

Keywords
Multiple inputs, input devices, interaction techniques, toolkits, assistive technologies

INTRODUCTION
Today, interactive desktop applications manage a very limited set of input devices, typically one mouse and one keyboard. However, the population of users requiring or simply possessing alternative input devices is growing, as well as the number of new devices available.

Given the proliferation of input devices and the importance of tasks performed by users on a computer, being able to adapt an existing application to one or several input devices is an important issue for physically challenged users, musicians, video game players, graphic artists etc. These users find their environment more usable or simply improved when they use their favorite input devices with existing applications.

However, the complexity of supporting alternative input devices is currently very high: each application has to explicitly implement some code to manage each device and all the desired interaction techniques using these devices. At best, specialized applications support a limited set of devices suited to their task.

In this paper, we describe ICON (Input Configurator), a system for selecting alternative input devices and configuring their interaction techniques interactively.

Using ICON, users can connect additional input devices – such as tablets, voice recognition software, assistive devices or electronic musical instruments – to an ICON aware application and/or assign specific interactive behavior to connected devices. Figure 1 shows a typical configuration.

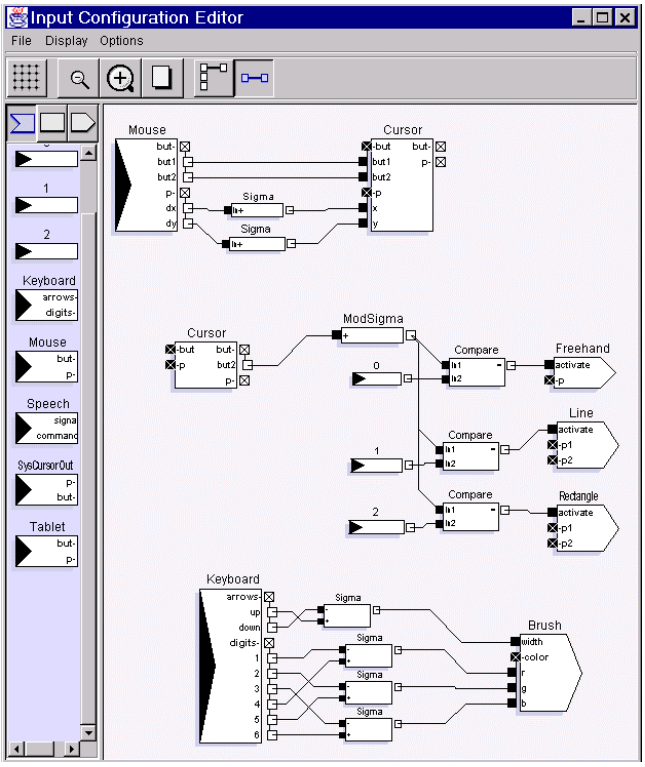


Figure 1: Screenshot of ICON showing part of the configuration of a drawing editor. The right mouse changes the selected tool and keypad keys change the color and line width attributes. Mouse relative positions are added and sent to a "cursor" module that abstracts a 2D locator device.

Adding new devices and configuring them using ICON not only opens applications to the use of alternative, better suited devices but also tremendously simplifies the integration of new interaction techniques published each year on conferences like CHI (for example [9, 12, 23]) but very seldom implemented on commercial products.

ICON is implemented using Java Swing and requires applications to externalize their interaction techniques to be effective. The effort required to do so is very modest compared to the benefits.

Typical ICON users need not be computer scientists; a good understanding of computer systems is sufficient. Users with disabilities or special requirements may need the help of such "power" users to configure their application.

This paper first describes a typical scenario a user would follow to edit the input configuration of a simple application. After comparing ICON to related work, we describe the ICON editor in details; we give then more implementation details and discuss practical issues for the use of ICON in applications.

SCENARIO

In this section, we show how John, a typical ICON user, may adapt a sample application called "IconDraw" that can draw lines, rectangles and freehand curves. By invoking the "configure" menu of IconDraw, John starts ICON that displays the current configuration as in Figure 1. This dataflow diagram shows several connected blocks called modules. Each module has input or output slots where links can be connected. Only connected slots are displayed in the figures. The two input devices of the default configuration – the mouse and the keyboard – are on the left, connected to some processing modules and finally to IconDraw's interaction tools appearing as input modules.

In IconDraw's standard configuration, the mouse is connected to a cursor module which displays feedback and is used as a virtual device. The right mouse button is used to cycle through the drawing tools. Keyboard keys are used to change the color and size of the lines. John can then change the configuration, save it to a file or load it from the configuration files available for IconDraw.

Stabilizing the Mouse Position

John remembers his friend Jane wants a drawing program but couldn't use a regular one because she suffers from Parkinson's disease that provokes uncontrollable hand shaking. With ICON, he can stabilize the pointer position by inserting a low-pass filter – averaging pointer positions to remove quick moves – between the mouse device and the cursor as shown in Figure 2.

To insert a LowPass module, John drags it from the left pane – where all existing modules are shown – and drops it in the editor pane. Clicking on one slot and dragging into another creates a connection. The configuration is effective immediately in IconDraw. When finished, John saves the configuration and sends it by email to Jane.

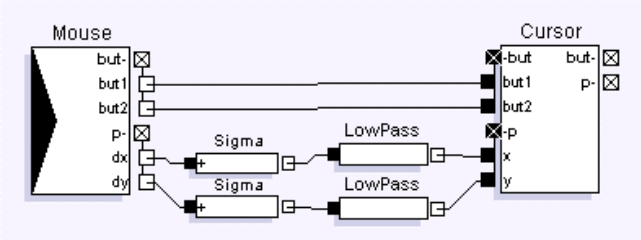


Figure 2: LowPass filters are inserted between the mouse and the cursor to stabilize the position.

Adding a Pressure Sensitive Stylus

John is a graphic designer and has a tablet with a pressure sensitive stylus. To use it inside IconDraw, he needs to disconnect the mouse, drag the tablet module in the ICON pane and connect it to the cursor through scale modules.

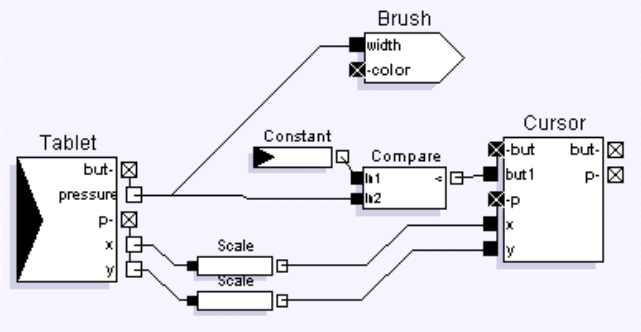


Figure 3: Adding a pressure sensitive stylus to IconDraw where the pressure changes the line width of the drawing.

To further use the pressure for changing the line width when drawing, he needs to connect the pressure slot of the stylus to the size slot of the brush used by the drawing tools, as shown in Figure 3. Brushes abstract graphic attributes just like cursors abstract positional devices. John can now draw with the stylus and have varying width strokes when using the freehand drawing tool.

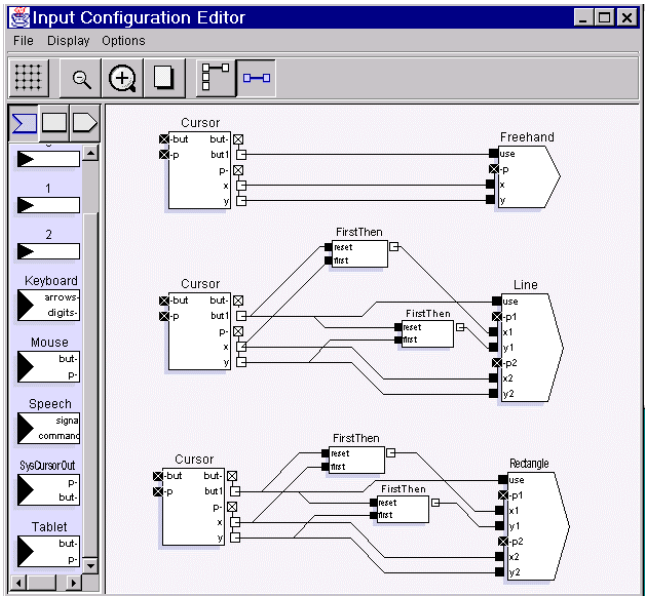


Figure 4: creation tools of IconDraw represented as input modules.

Configuring for Bimanual Interaction

Figure 4 shows part of the configuration controlling IconDraw's interaction tools. John now wants to use both his mouse and the stylus. A bimanual configuration requires a second pointer that can be dropped from the left pane into the ICON pane. Figure 5 shows the configuration required to create a line using bimanual interaction: one should be connected to the "p1" slot and the second to the "p2" slot. A combination of boolean modules determine when the creation mode is triggered and when it is finished.

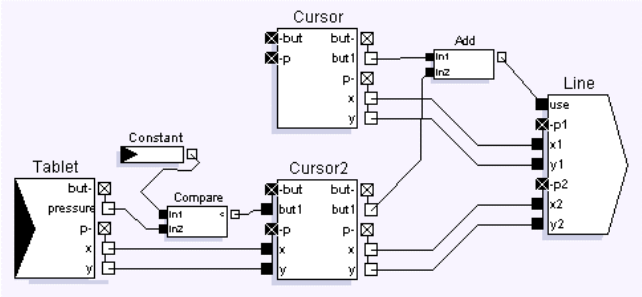


Figure 5: Configuration for creating a line with bimanual interaction.

Other Configurations

Current input modules also include voice and gesture recognition. John could use it to control the selected tool or to change the color if he wishes, effectively adapting the program to his skills and environmental particularities such as limited desk space or noisy environment.

RELATED WORK

There have been several attempts at simplifying the connection of alternative input devices or specifying the configuration of interactive applications.

Assistive technologies

Assistive technologies include hardware devices and software adapters. They are designed to allow disabled users to work on a desktop computer. Software adapters can be external applications that take their input from various special hardware devices and translate them as if they were actions on the mouse and keyboard. NeatTool [6] is such a system and configurations are designed using a rich graphical dataflow system. However, systems like NeatTool are limited to using existing interaction techniques of an application. More generally, external software adapters have problems when they need to maintain the internal states of an application like the current selected drawing tool.

In contrast, internal software adapters are becoming more common. Microsoft Active Accessibility® [20] and Java Swing [1] have provisions for accessibility and programmers can modify existing applications to adapt them to various input and output configurations. However, accessibility functions are not well designed for continuous interaction such as drag and drop or line drawing, and no graphical configuration tools exist yet, requiring a programmer's skill and sometimes source access to use them.

Games

Most current games offer some configuration options and all 3D video games offer a large choice of supported input devices. However, most of the games have a set of standard configurations and adapt alternative devices by compatibility. A regular action game will provide one configuration using the keyboard and another using positional devices (mouse, joystick or any compatible device). Sometimes, very specific devices are also managed like a force feedback joystick for flight simulators or a driving wheel for car racing. However, no general mechanism could allow a pianist to use a midi keyboard on a car racing program for example. The configuration is usually done through a simple form based interface or a simple script-like language which only allows direct bindings of device channels [13].

Furthermore, alternative input devices can only be used to play the game but other controls such as menus or dialogs can only be controlled by the regular mouse and keyboard.

3D Toolkits

3D toolkits and animation environments are usually aware of alternative input devices. The AVID/SoftImage system implements a "channel" library to connect any valuator device as a set of channels [2]. These channels can in turn be connected to internal values inside 3D models or trigger the internal constraint solver to perform sophisticated direct animations. However, channels are limited to these direct animations and cannot be used to enhance the interaction of the 3D modeling tool itself for instance. The World toolkit [25] can use any kind of device if it is described as an array of relative positional values. Again, this view of input device is meant for animation or direct control in VR environments but not for the interactive creation of objects or control of menus. Furthermore, the configuration of these input devices has to be programmed.

Recently, Jacob proposed a new architectural model to manage the interaction [15] using VRED, a dataflow system similar to ICON. Due to its complexity, VRED is meant to be used by expert programmers since it interacts deeply with the internals of animation programs.

2D Toolkits

There has been some attempts at simplifying the integration of alternative input devices in applications, such as the X Input Extension [8]. However, very few 2D programs use it and, when they do, their level of configuration is very limited. The X Toolkit [24] specifies a textual format to configure application bindings but its syntax is complex and requires the application to be restarted when it changes.

Myers described an interactive system for visual programming of interactions using interactors [22] in the Garnet environment. [21]. Garnet is still oriented towards programmers and interaction techniques cannot be changed

dynamically during the execution of an application. Furthermore, Garnet only manages one mouse and one keyboard.

Other systems for managing multiple devices exist such as MMM, Chatty's two-handed aware toolkit and Hourcade's MID library [5, 7, 14] but they all offer tools to programmers instead of users.

Classification

Table 1 summarizes this section, classifying existing systems in term of support for configurability and multiple input devices (MID). Configurability is classified in 6 categories: (1) none, (2) direct binding from device and events to program actions (i.e. without higher level control such as conditional binding), (3) environments configurable by users, (4) environments configurable by a programmer using a specialized language, (5) environments requiring a regular programming language for configuration, and (6) for completeness, adaptive environments that could automatically adapt interaction techniques and input devices to user's needs. No existing systems belong to this category yet. MID are classified in 4 categories: aware of only a fixed set of devices, aware of accessibility services and aware of many (a fixed set of classes) or any alternative devices.

Configure / MID	**Fixed set**	**Accessi-bility**	**Many**	**Any**
None	Most applications			
Direct binding	Video games		Midi config.	
User oriented	Hardware accessibility NeatTool	Software accessibil ity	Softimage Channels	**ICON**
Limited programmer oriented	Garnet interactors			VRED
Programming	Most toolkits	Java SWING MFC	World Tk	MMM Chatty MID
Adaptive				

Table 1: classification of current systems in term of support for configuration and multiple input devices.

THE ICON SYSTEM

The ICON Editor allows users to view and edit the mappings between all the available input devices and an application. It is based on a dataflow model, with the underlying semantics of reactive synchronous languages such as Esterel [4] or Lustre [11]. In this dataflow model, modules are input devices, processing devices, and application objects. Links are connections between input and output slots of the modules. A configuration is built by dragging modules into the ICON workspace and connecting them to perform high level actions, expressed as input modules. This section first describes how to use ICON, then how to design an ICON Aware application.

Using ICON

Modules that can be used to build an input configuration are available in three repositories : an output module repository, a processing module repository, and an input module repository (see Figure 1 on the left). Each type of module has its own graphical representation. New compound modules can also be created at will.

Output module repository: When the editor is launched, it asks the input APIs (WinTab [18] for the tablets, JavaSpeech [26] for the voice recognition engine, Direct Input [16] and USB [3] for yet other devices) for the set of connected input devices, and their capabilities. Device's capabilities are interpreted as a set of typed channels. For each detected device, a module is created and filled with output slots (corresponding to the device's channels), and is added to the output module repository. Timer modules also belong to this repository. With timer modules, users can detect timeouts, idle cursors, perform auto repeat or other time-oriented constructs.

Processing module repository: The editor contains a set of processing modules loaded from a library. A processing module has both input and output slots. There are three categories of processing modules: control modules, primitive modules and utilities. Control modules are useful to implement tests and control switches. Primitive modules include arithmetic, Boolean operations, comparisons and memorization of previous values. Utilities include debug modules and modules that could be built by composing primitives but are faster and smaller as primitives themselves.

Input module repository: This repository contains application-specific modules that are loaded when the user chooses an application to edit. These output modules show what the application needs in terms of input. It also contains global output devices such as the standard text output, the system cursor, or a Text-To-Speech engine.

Applications choose the level of granularity and how they describe their interactions in term of input modules. For example, all the atomic commands usually attached to menu items or buttons can be exposed as single input modules or can be grouped and exposed as one larger module. Exposing control objects as input modules is supported at the toolkit level. For application specific interactions, programmers have to provide suitable input modules according to the level of configurability they want to provide. More details are given in the next section.

Compound modules: Part of an input configuration can be used to create a user-defined module, by simply selecting a group of modules and issuing the "create compound" command. Selected modules and all the connections

between them are moved into a new compound module. External connections are also preserved: slots are automatically created on the compound device to enable external connections to internal modules.

Module properties: In addition to slots, some modules have properties that can be edited in a separate window. All modules have the *name*, *help,* and *enabled* properties. Other properties are mostly numerical parameters in mathematical processors.

Properties can also exist in input modules depending on the input API. As an example, recognizer devices issued from the JavaSpeech API have an array of string describing their vocabulary (empty by default). The properties describing the way an input module interprets user's actions to generate data is sometimes called the *device context*. Several instances of the same output module can live separately, each having its own device context.

Module cloning and shortcuts: Modules and groups of modules can be cloned in the workspace by dragging them while holding the control key. During this operation, all property values are copied. For example, cloning a processing module is useful when we have to perform the same operations elsewhere in the configuration. But an input module such as a mouse has a unique data source, and its clone will produce exactly the same data. However, cloning an input module can be useful to describe different device contexts (e.g. different working vocabularies for the same speech recognizer). The semantics of cloning an output module depends on how the application manages several instances of this module.

Another useful feature is module shortcuts. They allow the user to display the same device in different places of the workspace, so that an input configuration looks much more clearer(Figure 1-5 use shortcuts). A shortcut is made by dragging a device while holding both the shift and control keys.

Connections : Connections are created by dragging from one slot to another. Inconsistent connections – i.e. connections between input or output slots, type-incompatible slots, or connections that generate a cycle – are forbidden. Only authorized slots are highlighted during the dragging operation. ICON provides facilities for modifying connections, such as group deleting or fast reconnecting (changing one side of an existing connection).

Hierarchical slots: The configuration editor has a hierarchical representation of slots (Figure.6), which facilitates browsing of numerous slots. This also allows the structure of input devices to be preserved. Furthermore, hierarchical slots can be used to manipulate complex types.

Extended/Minimal Display : There are two display modes for modules. Extended mode shows all slots. Minimal mode shows only used slots, which reduces the visual complexity of a configuration. Entering a module with mouse cursor automatically displays it in extended mode.

Panning/Zooming : The user can pan and zoom on the workspace, and easily work on large configurations. It is also possible to enlarge and shrink individual modules.

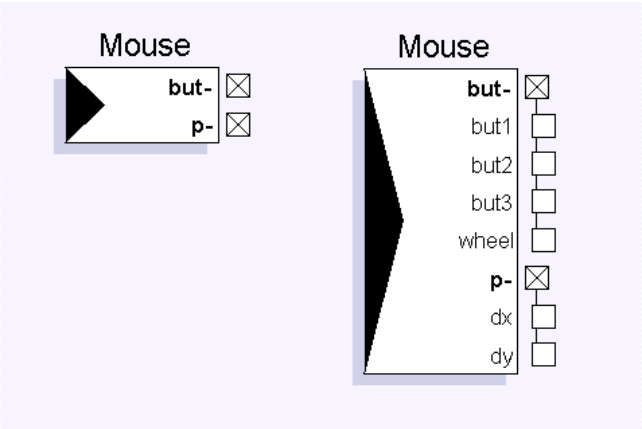


Figure 6: Hierarchical slots of the mouse device

Designing ICON Aware Applications

ICON changes the programming paradigm used by interactive systems. Current systems rely on an event based model, an event dispatching strategy and a state management programmed in a general purpose language such as Java or C. ICON uses another paradigm where values are propagated through a network of operations directly into actions. This paradigm is used extensively and successfully in automatic control systems, a domain we consider close to input configuration management.

Configurations in ICON are very similar to controllers in the traditional Smalltalk Model View Controller (MVC) triplet[17]. Externalizing the controller requires to externalize a protocol to communicate between a Model and a View. This protocol is modeled as input and output modules.

```
public class ToolModule extends Module {
// INPUT SLOTS
protected final IntSlot tool = new IntSlot("tool");
protected ToolBox toolbox;

public ToolModule(String name) {
  super(name);
  addInSlot(tool);
  toolbox = (Toolbox)getComponentNamed("ToolBox");
}

public void changed(Change change) {
  if (change.hasChanged(tool)) {
    toolbox.setTool(tool.getIntValue());
  }
}
}
```

Figure 7: Implementation of an input module displaying the currently selected tool in IconDraw.

Concretely, new modules have to be programmed in three circumstances: for a new application that needs to export some new Views and Models, when an unimplemented input device is available and when a new processing is required such as a gesture classifier. Programming a new

module involves modest efforts as shown in Figure 7. Existing modules have an average size of 50 lines, the largest being the speech recognition module, 512 lines long.

IMPLEMENTATION ISSUES

ICON is currently implemented in Java 1.2 and relies on Java Swing [10] with some specific extensions. Connections to the low level input APIs are usually implemented in C++ using the Java Native Interface [19]. The implementation is divided in three parts: the reactive language interpreter, the native modules API and the graphical editor ICON.

The Reactive Language Interpreter

The language underlying the execution of ICON is derived from Lustre and Esterel [4, 11]. Instead of defining a new semantics, we have relied on the well established synchronous reactive language semantics for the propagation of values and the flow control. Modules are like digital circuits and connections are like wires. Values are propagated during a clock tick that occurs at regular intervals or when a device requests it and at least one value has changed in the network. We have adapted and improved the value model of these languages by introducing hierarchical compound slots. A slot can either be of atomic type like integer or string, or by a collection of named slots. These compound or hierarchical slots simplify greatly the readability and construction of configurations. The interpreter is 4000 lines of source code long.

```
class Module {
    attribute String name;
    attribute boolean enabled;
    void addInSlot(InSlot s);
    void removeInSlot(InSlot s);
    List getInSlots();
    void addOutSlot(OutSlot v);
    void removeOutSlot(OutSlot v);
    List getOutSlot();
    boolean open(Frame f);
    protected boolean doOpen(Frame f);
    void close();
    protected void doClose();
    abstract void changed(Change c);
}
```

Figure 8: Module API for ICON.

The Module API

A module is implemented as a Java object with a list of input and output slots as shown in Figure 8. The interpreter calls the "changed" method of the object at each internal clock tick when at least one of its input slots has received a new value. The method can then compute new values for its output slots and the signal propagates through connections. New modules are simple to implement with this interface.

The Configuration Editor

The configuration editor modifies the reactive network interactively and relies on the reflection mechanisms of Java to expose module internals to the users. It represents about 4000 lines of source code.

DISCUSSION

The use of ICON on real program raises several issues that we discuss here. Among them, one can wonder about the kind of users who will use ICON, the expressive power of configurations edited with ICON and the practicality of the approach.

ICON Users

We don't expect all users to design large specific configurations of their applications with ICON. Instead, applications should come with a set of sensible configurations and users should usually modify small portions to suit their needs. However, with the possibility of sharing configurations with other users, we expect configurations to improve incrementally.

Some users may need special configurations, either because they have disabilities or because they have particular skills with some set of devices. These users will need the help of expert ICON users or programmers but still, they could adapt the programs to their abilities.

Expressive Power

ICON is a full language, although it doesn't allow run-time created modules or recursion. Considering the experience in the field of reactive synchronous languages, we have chosen to stay away from these run-time modifications. We also believe this is not a serious issue because users are not expected to build large dynamic configurations. As for the readability of dataflow systems, we haven't conducted experiments but they seem quite readable for the configurations we experienced. For larger configurations, the textual form of the ICON language could be better for some users, although compound modules and hierarchical slots enhance the readability by hiding details.

Practicality

ICON is currently in early stages: most of the specification of the interaction of interactive graphical applications can be configured outside the application but we haven't modified all the Swing components yet to be ICON aware. However, we have modified the Java Swing architecture to suit our needs and these modifications are quite small and not intrusive. We believe modern toolkits could benefit from this externalization of the interaction in term of modularity and extensibility. For example, the support of text input methods is currently implemented at a very low level in Java Swing and requires a complicated internal mechanism that could be more simply described using ICON. The same is true for some aspects of accessibility.

From an application programmer's point of view, interaction techniques can be exposed with any level of granularity and abstraction, providing a smooth transition path from no ICON support at all to full ICON support and extensibility.

CONCLUSION AND FUTURE DIRECTIONS

We have shown how ICON enables users to configure applications to available devices as well as to their skills. Such configurations previously required access to the source code of the application and were impossible to improve incrementally.

Configuring an application is important for disabled users, but also to users with special skills or working on a special environment (limited space, noisy, etc.) Furthermore, new devices or interaction modes can be tested and adapted to existing programs. This is even more important for combined devices and their interaction techniques. When a voice recognition software is used in conjunction with a pressure and tilt sensitive stylus, a large number of attributes are produced continuously and the best way to use them together has to be tested by trials and errors. Currently, this kind of testing can only be done by the application's programmers. ICON transfers this effort to any user who is motivated.

For future directions, we are currently implementing a C++ version of ICON's library to configure 3D virtual reality applications that require higher performance than Java can provide. We are also enriching our collection of supported input device managers to play with more exotic devices, including force feedback.

REFERENCES

1. Andrews, M. Accessibility and the Swing Set, Sun Microsystems Inc., http://java.sun.com, 1999.
2. Avid Inc. Channel Developer's Kit, Softimage Inc., 2000, www.softimage.com.
3. Axelson, J. *Usb Complete : Everything You Need to Develop Custom Usb Peripherals*. Lakeview Research, 1999.
4. Berry, G. and Cosserat, L., The synchronous programming languages Esterel and its mathematical semantics. in *Seminar on Concurrency*, (1984), Springer Verlag, 389–448.
5. Bier, E.A. and Freeman, S., MMM: A User Interface Architecture for Shared Editors on a Single Screen. in *Proceedings of the ACM Symposium on User Interface Software and Technology*, (1991), ACM, 79-86.
6. Carbone, M., Ensminger, P., Hawkins, T., Leadbeater, S., Lipson, E., O'Donnell, M. and Rajunas, J. NeatTool Tutorial, 2000, http://www.pulsar.org/neattools/.
7. Chatty, S., Extending a Graphical Toolkit for Two-Handed Interaction. in *Proceedings of the ACM Symposium on User Interface Software and Technology*, (1994), 195-204.
8. Ferguson, P. The X11 Input Extension: Reference Pages. *The X Resource*, *4* (1), 1992 195--270.
9. Frohlich, B. and Plate, J., The Cubic Mouse: A New Device for Three-Dimensional Input. in *Proceedings of ACM CHI 2000 Conference on Human Factors in Computing Systems*, (2000), 526-531.
10. Geary, D.M. *Graphic Java 2, Volume 2, Swing, 3/e*. Prentice Hall PTR, 1999.
11. Halbwachs, N., Caspi, P., Raymond, P. and Pilaud, D., The synchronous dataflow programming language Lustre. in *Proceedings of the IEEE*, (1991), IEEE, 1305-1320.
12. Hinckley, K. and Sinclair, M., Touch-Sensing Input Devices. in *Proceedings of ACM CHI 99 Conference on Human Factors in Computing Systems*, (1999), 223-230.
13. Honeywell, S. *Quake III Arena: Prima's Official Strategy Guide*. Prima Publishing, 1999.
14. Hourcade, J.P. and Bederson, B.B. Architecture and Implementation of a Java Package for Multiple Input Devices (MID), Human-Computer Interaction Laboratory, University of Maryland, College Park, MD 20742, USA, 1999, http://www.cs.umd.edu/hcil/mid.
15. Jacob, R.J.K., Deligiannidis, L. and Morrison, S. A Software Model and Specification Language for Non-WIMP User Interfaces. *ACM Transactions on Computer-Human Interaction*, *6* (1), 1999 1-46.
16. Kovach, P.J. *Inside Direct3d*. Microsoft Press, 2000.
17. Krasner, G.E. and Pope, S.T. A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. *Journal of Object-Oriented Programming*, *1* (3), 1988 26--49.
18. LCS/Telegraphics. The Wintab Developers' Kit, LCS/Telegraphics,, http://www.pointing.com/WINTAB.HTM, 1999.
19. Liang, S. *The Java Native Interface : Programmer's Guide and Specification*. Addison-Wesley Publisher, 1999.
20. Microsoft Corporation. Microsoft Active Accessibility SDK 1.2, Microsoft Corporation, 1999.
21. Myers, B. The Garnet User Interface Development Environment. in *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, 1991, 486.
22. Myers, B.A. A New Model for Handling Input. *ACM Transactions on Information Systems*, *8* (3), 1990 289-320.
23. Myers, B.A., Lie, K.P. and Yang, B.-C., Two-Handed Input Using a PDA and a Mouse. in

Proceedings of ACM CHI 2000 Conference on Human Factors in Computing Systems, (2000), 41-48.

24. Nye, A. and O'Reilly, T. X Toolkit Intrinsics Programming Manual., *4*, 1993 567.

25. Sense8 Corp. The World Toolkit Manual, Sense8, 1999.

26. Sun Microsystems Inc. Java Speech API Programmer's Guide, Sun Microsystems Inc,, http://java.sun.com/products/java-media/speech/forDevelopers/jsapi-guide/index.html, 1998.

The MaggLite Post-WIMP Toolkit: Draw It, Connect It and Run It

Stéphane Huot, Cédric Dumas
Ecole des Mines de Nantes
4, rue Alfred Kastler
44307, Nantes, France
Tel: (+33) 251-858-241
{shuot, cdumas}@emn.fr

Pierre Dragicevic
LIIHS-IRIT
118, route de Narbonne
31062, Toulouse, France
Tel: (+33) 561-556-965
dragice@irit.fr

Jean-Daniel Fekete
INRIA Futurs/LRI Bât. 490
Université Paris-Sud
91405 ORSAY, France
Tel: (+33) 169-153-460
Jean-Daniel.Fekete@inria.fr

Gérard Hégron
CERMA UMR CNRS 1563
EAN, Rue Massenet
44319, Nantes, France
Tel: (+33) 240-594-324
hegron@cerma.archi.fr

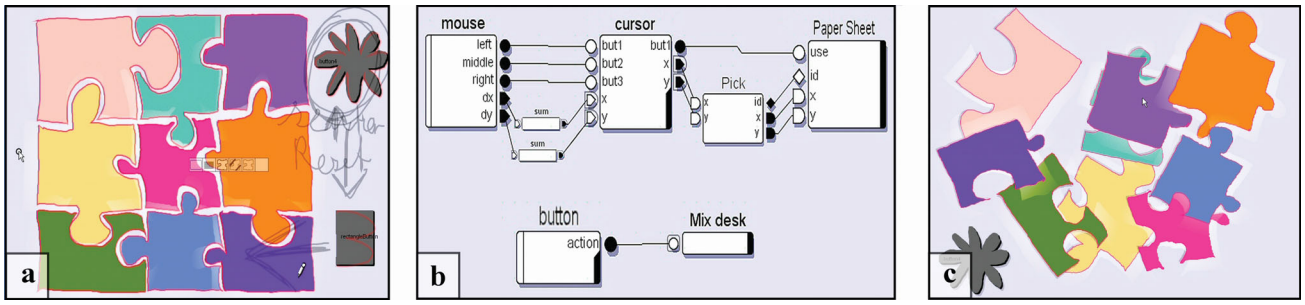


Figure 1: Designing a puzzle UI with MaggLite. (a) Drawing UI. (b) Configuring interactions. (c) Using the application.

ABSTRACT
This article presents MaggLite, a toolkit and sketch-based interface builder allowing fast and interactive design of post-WIMP user interfaces. MaggLite improves design of advanced UIs thanks to its novel *mixed-graph* architecture that dynamically combines scene-graphs with interaction-graphs. *Scene-graphs* provide mechanisms to describe and produce rich graphical effects, whereas *interaction-graphs* allow expressive and fine-grained description of advanced interaction techniques and behaviors such as multiple pointers management, toolglasses, bimanual interaction, gesture, and speech recognition. Both graphs can be built interactively by sketching the UI and specifying the interaction using a dataflow visual language. Communication between the two graphs is managed at runtime by components we call *Interaction Access Points*. While developers can extend the toolkit by refining built-in generic mechanisms, UI designers can quickly and interactively design, prototype and test advanced user interfaces by applying the MaggLite principle: "draw it, connect it and run it".

Categories and Subject Descriptors: H5.2 [**Information Interfaces**]: User Interfaces—Graphical user interfaces (GUI), User interface management systems (UIMS), Interaction styles, Prototyping, Input devices and strategies; D2.11 [**Software Engineering**]: Software Architectures.

Additional Keywords and Phrases: GUI toolkits, GUI architectures, ICON, MaggLite, interaction techniques, interaction design.

© ACM, 2004. This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology (UIST 2004), {ISBN: 1-58113-957-8, (2004)} http://doi.acm.org/10.1145/??????.??????

INTRODUCTION

Most GUI toolkits are based on heavyweight components that encapsulate presentation and interaction into single monolithic components. Using such architectures, extending either the presentation or the interaction of components is challenging, sometimes even impossible. Without extensive built-in support for Post-WIMP interaction techniques and alternative input, developing advanced GUIs is a long and expensive process. Although prolific work has been carried out on new user interface toolkits, there is still a strong need for tools that would allow easy development or prototyping of advanced and innovative interactive applications. The MaggLite Post-WIMP toolkit is a new step toward such tools. Relying on a multi-input interaction model and fine-grained scene-graph architecture, MaggLite offers new possibilities in designing user interfaces:

- From the UI design perspective, MIB, a sketch-based interface builder included in MaggLite, allows very-high-fidelity prototyping of post-WIMP user interfaces. Interactions are specified graphically using the ICON data-flow editor [10]. The tools and mechanisms provided by MaggLite (Interface Builder, base components, tools abstrac-

tions and generic interactions) offer a high flexibility that eases the work of the UI designer, whether he is a graphic designer or a programmer. While developers can easily extend the toolkit by refining generic mechanisms, UI designers can rapidly prototype and test advanced user interfaces by applying the MaggLite principle: "draw it, connect it and run it".

▪ From the UI architecture perspective, the *scene-graph* architecture provides mechanisms to easily manage advanced graphical effects (transformations, transparencies, shadows, fading, etc.) and the *interaction-graph* provides an efficient way to specify interactions and connect physical input devices to the application components and tools. This fine-grained description of interaction improves flexibility and extensibility of the toolkit compared to traditional event-based architectures. The MaggLite architecture uses a novel *mixed-graphs* model, where appearance and interactions are intertwined but still described separately. The dynamic combination of the graphs makes UIs fully reconfigurable at runtime.

▪ From the toolkit perspective, MaggLite allows for high input-independence by introducing the concept of *pluggable interaction techniques* as bridges between physical input devices and application objects. MaggLite also provides built-in support for most non-standard input devices (tablets, joysticks, MIDI-devices, webcams, etc.) while remaining extensible. The whole toolkit is designed with interchangeability of input devices and interaction techniques in mind. As a result, many appearances and behaviors can be composed, tested and refined in a very short time when designing advanced GUIs.

The next section introduces the main concepts behind MaggLite by walking through the design of a sample "puzzle" application. Then, we describe the MaggLite architecture and discuss related work before concluding.

UI DESIGN WITH MAGGLITE

MaggLite introduces a straightforward UI design process in three steps:

1. *Draw it*: Using the sketch-based interface builder (MIB for MaggLite Interface Builder), an UI designer draws components on the UI space. These graphical objects are usable immediately after they are created.

2. *Connect it*: With ICON, interactions are graphically described using a dataflow visual language that allows specifying and reusing advanced interaction techniques. Convincing mock-ups can be built without writing one line of code.

3. *Run it*: The designed UI is fully functional during creation; it can be executed, refined and saved. The saved configuration can be reloaded 1) as a mock-up from a generic application, 2) from the specific application is has been created for, or 3) from another application for reusing parts of its behavior or graphical components.

To illustrate it, we will observe Geoff, a UI designer, creating a sample "puzzle" application using MaggLite.

Designing a puzzle UI with MaggLite

As a designer, Geoff can create simple script-based programs but prefers graphical interactive tools to express all his creativity when designing UIs. He is usually asked to create functional applications instead of mere bitmaps or screenshots, in order to test and evaluate the feasibility and usability of the UI. In our scenario, Geoff has to design the UI for a simple and intuitive puzzle game. We show how MaggLite built-in tools help him in carrying-out this task.

Drawing the UI (Figure 1a). To design his puzzle UI graphics, Geoff successively uses the tools available on the MIB toolglass. Each time he draws a stroke starting from inside an area of the toolglass, an object of a specific type is created or a command is applied to the underlying object.

1. *Freehand sketching tool*: Geoff draws each piece of the Puzzle by initiating a drawing through the "basic component" area of the toolglass (Figure 2). This tool is used to draw freeform UI components, and is well-adapted for our puzzle example. The shape of the created object is given by the entire stroke.

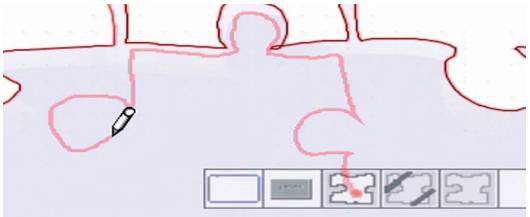


Figure 2: Sketching a component with MIB.

2. *Color changing tool*: Geoff changes the color of the puzzle pieces by moving the "colorizer" area of the toolglass over the object and clicking through it (Figure 3). The colorizer's attributes (current foreground and background colors) can be changed beforehand.

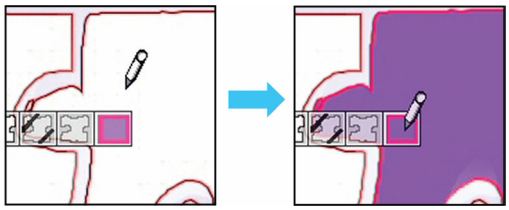


Figure 3: MIB - Changing colors with the toolglass.

3. *Gesture tool*: after he has drawn the puzzle pieces, Geoff adds a button that will trigger the shuffling of the pieces. By drawing the appropriate gesture, he creates a rectangular button (Figure 4). Each time a gesture is recognized, a predefined object is created or an action is performed (deletion, move, etc.). If gestures are not recognized, strokes are kept as annotations (Figure 4c). Finally, Geoff decides to design a fancier button. He draws a flower-shaped button (top right of figure 1a)

with the freehand sketching tool before erasing the rectangular one with a gesture.

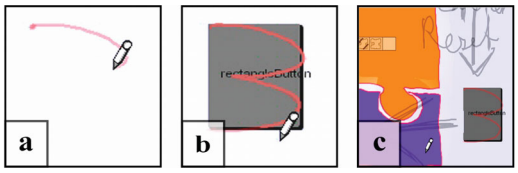


Figure 4: MIB - Gestures. (a) start of a new gesture. (b) recognized (c) unrecognized (annotations).

Advanced post-WIMP interfaces often rely on non-rectangular shapes for their graphics. MaggLite supports freeform shapes with graphical effects such as transparency. Furthermore, MIB has been developed with the MagLite toolkit itself and makes rich use of Post-WIMP techniques: in addition to toolglasses, it exploits freehand sketching and gesture recognition with stylus devices in order to make UI design simple and natural, close to designer habits (as with SILK [23,24]). If some designer prefers other interaction techniques for building UIs, MIB can be applied to itself.

Configuring and Running the UI (Figure 1b). When Geoff adds new components, he can graphically specify how they are controlled with input devices using the ICON data-flow editor. The basic building blocks of ICON are called "devices", which denotes physical input devices but also filters (we come back to ICON in the next section). To describe the behavior of his puzzle UI, Geoff starts by connecting the outputs of the "mouse" device to the inputs of a "cursor" feedback device (Figure 5, step 1). At this point, he is able to move a cursor over the UI graphical components with his mouse. Now, Geoff needs to describe the interaction with the graphical components.

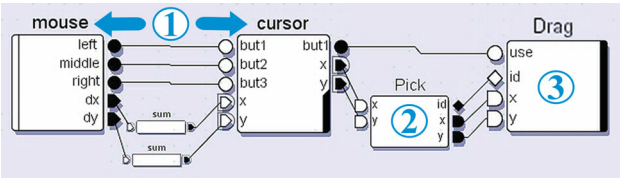


Figure 5: Puzzle interaction configuration in 3 steps.

MaggLite provides specific devices to link interactions and UI components: *Interaction Access Points* devices (IAPs for short). For his puzzle, Geoff uses two kinds of IAPs: *interaction devices* and *manipulators*.

Geoff adds a *picking device* in the dataflow diagram and connects it to the cursor (Figure 5, step 2). Picking is thus enabled: each time it receives a new cursor location, the picking device sends on its output slots the list of picked objects and the coordinates where the pick occurred. Geoff then decides to use a simple drag interaction to move the pieces so he connects the MaggLite *drag device* after the picking device. Because Geoff wants the dragging to start when the mouse button is pressed he connects the button output slot of the cursor to the "use" slot of the drag device. He can now drag picked pieces (Figure 5, step 3).

Geoff soon realizes that a drag interaction is not well adapted, because he is only able to translate the pieces, not rotate them, so without leaving the application, he replaces the drag device with the "paper sheet" device (Figure 1b). The "paper sheet" technique, described in [5], allows translating and rotating an object with a single gesture. Like other interaction techniques, it is available in MaggLite as a simple IAP device.

Now, Geoff needs to specify the behavior of the "shuffle" button. For this purpose, each MaggLite graphical component created using MIB also appear as a device in ICON. Those devices are called *manipulators*. Geoff's button manipulator has an "action" output which sends the Boolean "true" value each time the button is clicked (Figure 6).

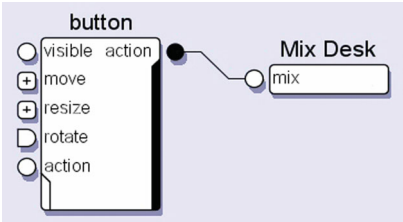


Figure 6: Button manipulator.

During the initial design, the button is not connected to any action but later, Geoff asks a programmer to create a "Mix Desk" device to shuffle the puzzle pieces. Once the programmer is done, Geoff plugs the output slot of the button manipulator to the input slot of the "Mix Desk" device (Figure 6). The shuffling function is the only part of this application that has to be programmed. Simpler actions can sometimes be built graphically using existing ICON devices or programmed directly inside the editor with "*scripting*" devices that interpret JavaScript code.

Using, refining and shipping the designed UI (Figure 1c). Geoff has now made a fully functional puzzle UI he can save into an XML file that can be launched by any user having MaggLite installed. This final puzzle application is illustrated in Figure 1c. The original gestures and annotations are not displayed anymore, though saved in the file. At this stage, the interaction and the appearance are still fully configurable at runtime. Geoff himself can continue to refine the puzzle and test other interaction techniques. For example, he can replace the mouse by a tablet to move the pieces and use the pressure sensitive capabilities of the stylus to adjust the moving speed. He can also directly connect a keyboard key or a speech command to the "Mix Desk" device as a shortcut for shuffling the pieces. He can even duplicate his original dataflow diagram, allowing for multi-user interaction with the puzzle. All these configurations can be saved and proposed as alternatives to different users, depending on their personal taste, abilities or hardware configuration.

The puzzle application was actually built in about ten minutes by one of the authors. This simple scenario aimed at showing how MaggLite and its associated tools can be used for building Post-WIMP interactive applications graphically. In the next section we describe MaggLite's architecture and mechanisms and give more examples of use.

MAGGLITE

MaggLite is a Java toolkit that relies on a *mixed-graphs* model to describe both appearance and behavior of interactive applications. A mixed-graph is made-out of a scene-graph, an interaction-graph and mechanisms to communicate between them. As previously seen, MaggLite also provides two interactive builders: MIB for interactively editing the scene graph and ICON to interactively specify the interactions and connect them to the scene-graph and the application-specific behaviors.

The mixed-graphs model

The MaggLite graphical part builds upon classical scene-graph approaches. This common model for 3D toolkits has already been used by some advanced 2D toolkits [7,3,25]. Scene-graph approaches break the heavy structure of widget-based architectures by using fine-grained graphical objects that can be grouped to create more complex graphical objects. MaggLite goes beyond the scene-graph approach by decomposing not only graphics, but also interactions. Interactions are described by *interaction-graphs* that break the input-event structure into dataflow graphs. Those graphs are made of interconnected filters we call "devices", which can be in turn decomposed into individual channels. The interaction-graph dynamically updates the scene-graph structure and components when needed, i.e. in reaction to user input or system events. The mixed graphs architecture makes a clear distinction between the appearance and the behavior of the interface [22]. Furthermore, the interaction-graphs can be reconfigured graphically in many ways to adapt the UI or to test alternative interaction techniques at runtime.

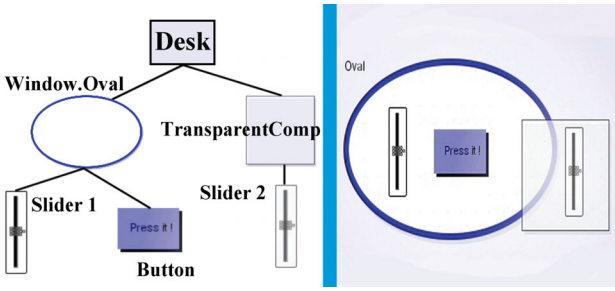


Figure 7: A simple MaggLite scene-graph and the corresponding interface.

Scene-graphs. A MaggLite scene-graph is made up of a root object, the *desk*, and nodes that are graphical objects (Figure 7). The MaggLite toolkit provides several classes of graphical objects but most of them can be instantiated with arbitrary shapes. This shape abstraction allows more expressivity in designing the interface appearance as objects are not limited to rectangular or elliptic shapes. Graphical properties, such as colors, geometrical transformations or opacity, are embedded into objects and propagated to child nodes by graphic contexts.

MaggLite provides a predefined set of *atomic graphical objects* with several possible states (transparent, *stuck*, layers, etc.). The toolkit also includes *composite graphical objects* which encapsulate predefined scene-graphs having a well-defined behavior. For example, the slider of Figure 8 combines a pane and a thumb and constrains the thumb location to the slider's internal model. All regular widgets (buttons, sliders, windows, etc.) can be implemented as composite objects. Predefined atomic and composite objects can be extended by inheritance whereas new ones can be created from abstract classes and interfaces.

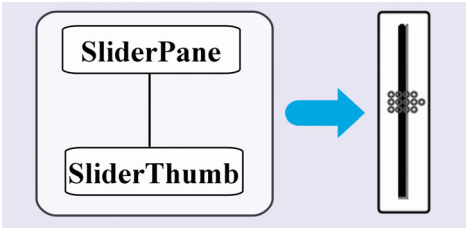


Figure 8: A widget defined by a scene-graph of atomic graphical objects.

One major issue with most classical scene-graph approaches is that interaction is described together with graphics by adding special nodes in the scene-graph, if not simply hiding it inside graphical nodes. To achieve more flexibility and extensibility, our mixed-graphs model describes interactions separately in interaction-graphs. Interaction-graphs rely on ICON [10], a toolkit we previously developed for handling advanced input. So that following sections about interaction-graphs could be understood, we give in the next section a brief overview of the ICON model and graphical language.

The ICON (Input Configurator) Toolkit. ICON is a Java toolkit and interactive editor for creating input-reconfigurable interactive applications, i.e. applications that can be controlled using a wide range of input devices and interaction techniques [10]. ICON introduces a reactive dataflow architecture that describes input methods using interconnected modules.

ICON's model is based on devices, which are a broad generalization of input devices: ICON's devices can produce output values, but can also receive input values. A device contains typed channels called *input slots* and *output slots*, as well as parameters to configure them. Slot types have a distinct graphical representation depending on their type (e.g. circle for Booleans, triangle for integers) and can be hierarchically grouped to form structured types (Figure 9).

There are three main categories of devices: *system devices* describe system resources such as input peripherals (mice, keyboards, tablets, speech input, etc.); *library devices* are

system-independent utility devices that range from simple boolean operators to complex feedback/interaction devices such as cursors, toolglasses and gesture recognizers; *application devices* are devices that control application (domain) objects. System and library devices are part of the toolkit whereas application devices are meant to be implemented by application programmers through a straightforward process.

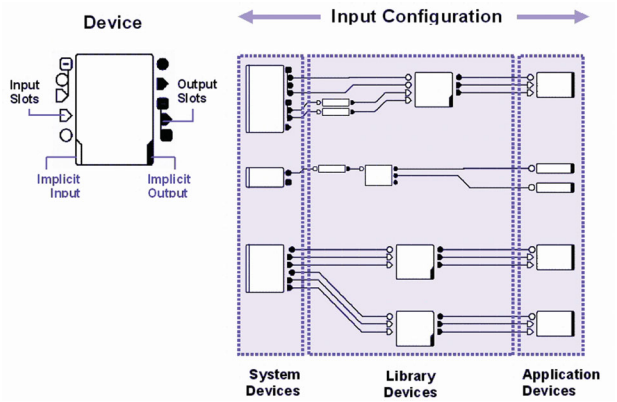


Figure 9: ICON components graphical representation.

An output slot of a device can be linked to one or several compatible input slots of other devices by *connections*, which are represented by wires (Figure 9). A set of connected devices defines an *Input Configuration* that can be executed by ICON's reactive interpreter. ICON's graphical editor allows mapping input devices to applications interactively. Currently plugged input devices and all devices of the ICON library are shown on a container and just have to be dragged towards the editor's pane to be used. The mapping task may involve insertion and connection of devices that encapsulate predefined interaction techniques (e.g., a gesture recognizer is meant to be inserted between a pointing device and a text input component) as well as the description of new interaction techniques by the combination of simpler processing devices.

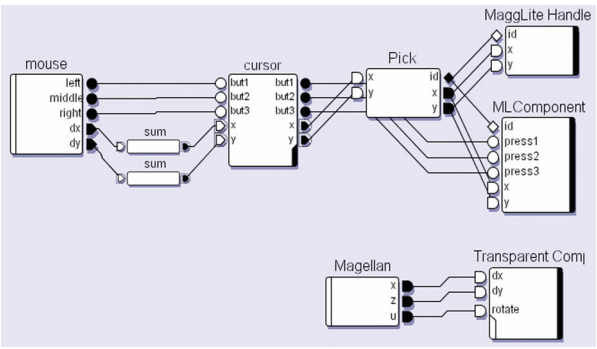


Figure 10: A possible interaction-graph (described by an ICON configuration) for the scene-graph of Figure 7.

Interaction-graphs. A MaggLite interaction-graph is an ICON configuration describing the way a scene-graph can be manipulated (Figure 10). The same way as scene-graphs break heavy graphical structures into primitive graphical components, interaction-graphs break heavy event structures into dataflow processing devices and slots. This dataflow paradigm allows the description of interactions at a finer grain than using standard input events. For example, in event-driven toolkits, there is an event type for each kind of input device. To support a new device type (such as a game pad or a pressure sensitive stylus), a new input event type has to be defined, the event dispatch mechanism has to be adapted for this new type of events and each object has to be extended to handle the new event. With MaggLite, if the new input device is not yet supported, one just needs to implement a new device for it. Once available, it can be used like any other existing device to control graphical objects through existing or new interaction techniques.

Interaction Access Points (IAPs)

As described in the puzzle example, IAP devices provide different ways of linking ICON devices (input devices) to MaggLite graphical components. In our mixed-graphs paradigm, IAPs are dynamic connection points between the interaction-graph and the scene-graph. Each time a new graphical object is created, it also appears as a device into ICON's device pane. IAPs can be considered as "runtime glue" to dynamically compose a mixed-graph. They transmit and receive data from the scene-graph or modify its structure by inserting or removing nodes, changing their properties or triggering special behaviors. We distinguish three kinds of IAPs: *interaction devices* and *manipulators* (mentioned in the puzzle example), as well as *InnerTools*.

Interaction devices. We introduced interaction devices with the *Picking* device in the puzzle example. We describe it further and also describe a pluggable interaction technique called *Responsive Handles*.

The picking device communicates with the desk, root of the scene-graph. Each time it receives new positional values it asks the desk for nodes of the scene-graph under the given location. It then sends references of picked objects through its output slot. Due to this externalization of the picking operation, MaggLite is not limited to one picking technique. Currently, MaggLite supports the conventional *under-cursor* picking (with several strategies) and *proximity-picking* (all objects within a certain distance are picked, which can be materialized as a halo surrounding graphical objects).

Classical UI architectures encapsulate interactions within objects. In MaggLite, interaction techniques are pluggable. Graphical components declare their capabilities by implementing Java interfaces whereas each interaction device is specialized into a specific capability. For example, a graphical object willing to be moved needs to implement the "Moveable" interface, which makes him compatible with interaction devices such as "Drag" and "Paper Sheet". This mechanism avoids heavy coding when implementing new interaction techniques and can be found in other approaches such as instrumental interaction [4]. We will

illustrate this by adding a new manipulation technique called *Responsive Handles*.

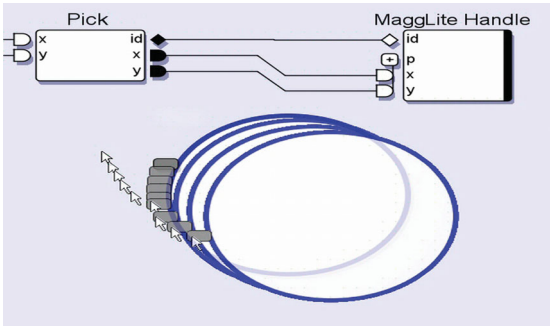


Figure 11: Responsive Handle interaction technique provides a generic interaction with movable objects.

The *Responsive Handles* technique works as follows: when the pointer reaches the proximity of a moveable object, a handle appears right on its shape boundary. This handle follows the pointer as long as it stays in the object proximity. The user can drag the handle to move the object. Figure 11 illustrates this technique and Figure 12 shows the underlying communications in the mixed-graphs model. When a moveable object is proximity-picked a handle is inserted in the scene-graph just before the object. When it is moved, the handle moves the object accordingly. The moving of the handle itself is managed by a conventional drag device. When the object proximity is not picked anymore, the responsive handles device removes the handle from the scene-graph.

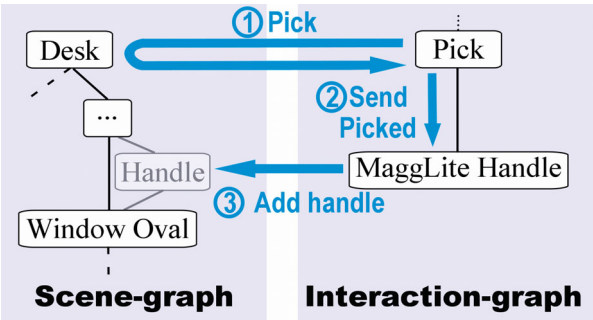


Figure 12: Mixed-graphs – Responsive Handle. Pick receives signals, and queries for picked objects (1). It transmits values to the handles device (2) which inserts a handle object in the scene-graph (3).

Other MaggLite pluggable interaction techniques include:

- All advanced interaction techniques provided by ICON, such as speech and gesture recognition, toolglasses, etc.
- an *event producer* device, which integrates MaggLite to Swing's event-driven architecture,
- *drag* and "*paper sheet*" [5] interactions, that provide moving moveable objects,
- the *Magnifier*, which zooms parts of scene-graphs,
- the *Fisheye*, which apply a fisheye lens deformation on parts of scene-graphs.

Manipulators. Unlike generic interaction devices which can potentially manipulate any scene-graph object of a given class, *manipulators* deal with instances *i.e.* individual scene-graph nodes. As explained in the puzzle example, an object manipulator is an ICON device that externalizes entry and output points to interact with the object. For example, a moveable object externalizes move slots, which expect to receive coordinates. Figure 13 shows a moveable object connected to a joystick. This principle of direct association is a simple and efficient way for performing direct manipulation [4].

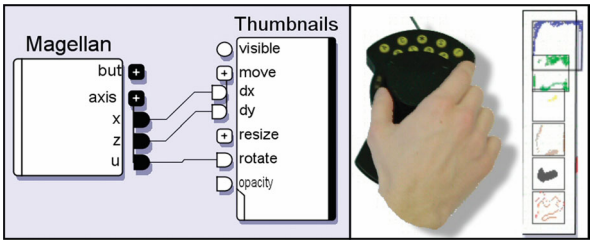


Figure 13: Direct connection between an input device and a manipulator device.

Manipulators can also be useful to describe interactions between UI components (Figure 14). The behavior of the three widgets is configured graphically as follows: the output value of the slider is connected to the input of the text zone by a conversion device. When the slider is moved, the text zone displays the slider's value. When the button is pressed, the "pass" device resets the slider's value to the constant specified by the "intValue" device. Such interaction-graphs are well-adapted to explain UI mechanisms and prototype them in an educational context.

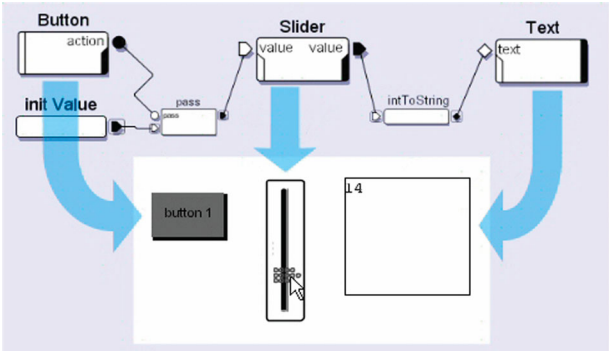


Figure 14: Connection of manipulators devices. The button resets the slider and the slider value is displayed in the text zone.

InnerTools. MaggLite provides a third way to manage interactions that was not used in the puzzle example: *InnerTools*. These tools are IAPs that receives positional values (in screen pixel coordinates). When an InnerTool is added to a compatible object, it is active only within the bounds of the object. It also maintains a cursor if needed (Figure 15).

A *DrawInnerTool*, implementing drawing actions, is included in MaggLite to allow easy creation of drawing inter-

faces. We use it as example to explain how InnerTools are processing coordinates, and therefore their two behaviors:

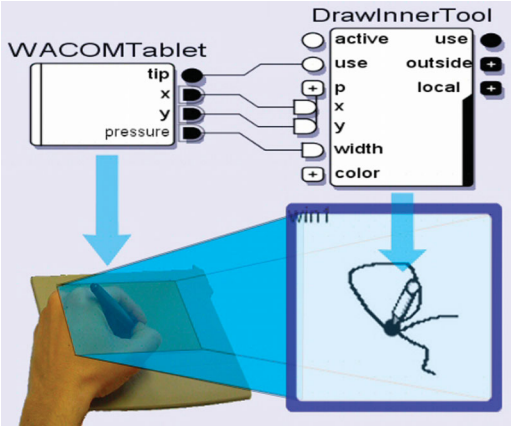


Figure 15: InnerTools – Full mapping mode.

1. *Full mapping*: received positional values are converted into component local coordinates. This mode is well-adapted to absolute pointing input devices (e.g. tablets) because the device is entirely mapped to the component. On Figure 15, the DrawInnerTool is connected to a digitizing tablet device, in full mapping mode.

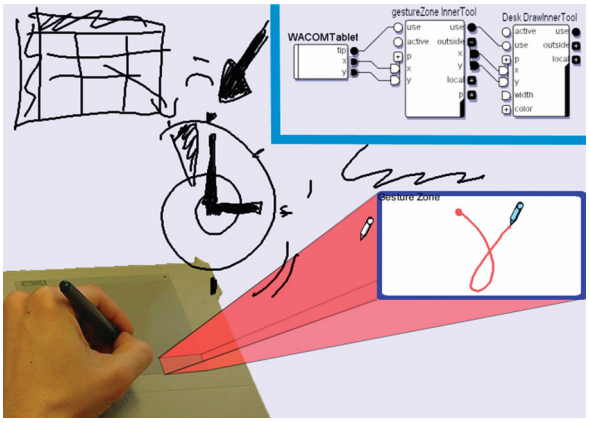


Figure 16: Using InnerTools in global mode to restrict interactions to an input device zone.

2. *Global mode*: when an InnerTool is not used in full mapping mode, received positional values are interpreted in screen coordinates. The tool is active when the coordinates are inside the bounds of the linked component. In other cases, the tool is inactive and coordinates are sent through the InnerTool's output slots. Figure 16 shows two InnerTools in global mode, connected together. The first one, named *gestureZone*, is linked with the blue bordered component and performs gesture recognition. Its input slots are connected to a tablet. The second tool is a *DrawInnerTool* linked with the application desk. Its input slots are connected to the forwarding output slots of the gesture tool. When the tablet send screen coordinates that are in the bounds of the gesture zone component, the linked gesture tool is activated and doesn't forward any values to the desk drawing tool. When coordinates are outside of the gesture zone, the gesture tool is disabled and it forwards coordinates to the desk drawing tool. Therefore, the user can draw anywhere on the desk, except in the gesture zone where his strokes are interpreted as gestures. This mapping of screen areas into input devices is an effective way to specify post-WIMP interactions, especially when working with tablets, screen tablets or touch-screen devices.

When an InnerTools is added in an interaction-graph, it installs its feedback (the cursor) in the scene-graph on top of the component the tool is linked with. This allows the feedback to be painted over the component. Then, when the "use" signal is triggered on the device, its action method is executed to perform the implemented interactions. MaggLite provides three working InnerTools: a drawing tool, an erasing tool and a generic tool used to restrict an existing interaction technique to the bounds of a component. Additionally, developers can easily extend the InnerTools set by subclassing the InnerTools class. They only have to implement the "action" method as other mechanisms (input handling modes, feedback, coordinates conversions) are inherited.

Extending MaggLite

MIB was developed to ease the prototyping of advanced UIs with MaggLite. It is also a good example of an application developed with MaggLite and extending it. It took one of the authors about two hours to implement MIB. MIB has a main IAP named "*MaggLiteBuilder*". It receives a command and a shape, in fact the class and the shape of the object to insert in the application scene-graph. Therefore, it can be connected to any device able to send commands and/or shapes (voice commands, gesture commands, drawing tools, etc.) The MaggLiteBuilder device is also able to save created UI into an XML file.

In its default input configuration, the MIB main tool (sketch, gestures and color changing) is controlled with a tablet and the toolglass is moved with a Magellan (6DOF isometric device). A mouse can be used to move created objects. A MIDI fader box allows specifying the colors of the colors changer (one fader for each RGBA component). Like all MaggLite applications, MIB can be controlled with other types of input devices and interaction techniques embedded in the toolkit (voice recognition for changing colors, mouse for sketching, keyboard for selecting object classes, etc.)

A more complex application and toolkit extension is currently under development: Svalabard [19], a sketch-based 3D modeler. User studies conducted in the early stages of this work as well as the state of the art show that post-WIMP techniques are essential to bring creativity in 3D modeling. This project heavily uses and extends MaggLite by adding domain-specific components and devices such as drawing filters or user behavior analyzers. Those new devices and components are added to the toolkit. Once added however, they can be easily reused in other applica-

tions. Finally, we describe a scatter plot visualization built with MaggLite as an example of use of the toolkit application template.

Building a Scatter Plot Visualization using MaggLite

MaggLite is not a visualization-dedicated toolkit but allows rapid development of advanced visualization techniques. "*MaggLiteVisu*" is an application that allows displaying 6-dimension data using scatter plots [13]. An example dataset can be census data with attributes such as city names, population, area, crime, etc. Each entry of the dataset, loaded from an XML file, is visualized with a rectangular graphical object. Attributes of the dataset are then assigned to graphical properties of the component (width, height, color, position). It took us only ten minutes to develop this well-known visualization technique using MaggLite.

Using the MaggLite application template, only thirty lines of code have to be written for parsing the file (using the Sun Java XML parser) and create MaggLite components. There are only two abstract methods to implement when extending the application template class:

`createMaggLiteComponents`, for new MaggLite graphical objects instantiation. Generally, one will use existing classes and instantiate them with a specific shape.

`createApplicationDevices`, to add newly created ICON devices. These devices are application-specific and will only show in ICON while configuring this application.

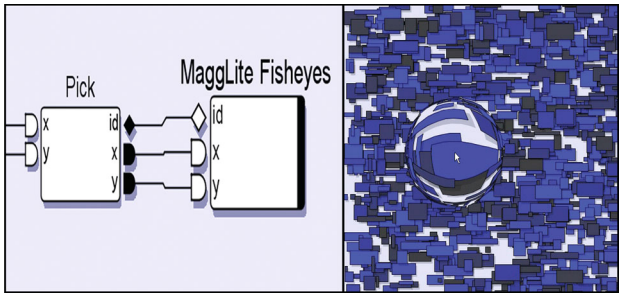


Figure 17: *MaggLiteVisu* application. A large dataset is displayed, with a generic fisheye connected.

More than programming ease, the strength of MaggLite is its ability to provide dynamically configurable interaction and visualization techniques without writing a single line of code. In Figure 17, one can use the toolkit pluggable fisheye lens simply by connecting it in ICON.

RELATED WORK AND DISCUSSION

In this section we compare MaggLite with other approaches. Those include programming and prototyping tools, as well as some interaction techniques close to those used in MaggLite.

Advanced GUI Toolkits

Although mainly used to develop conventional UIs, toolkits such as Garnet/Amulet [26] and Subarctic [18] have introduced several advanced features to facilitate UI programming, such as constraint-based layout management [18]. They also describe input and event handling in a cleaner, more general and extensible way than traditional toolkits [26,15]. Even though they support at some level Post-WIMP techniques such as simple gesture recognition, they are only aware of a limited set of input devices and require significant modifications to handle any new interaction paradigm. In contrast, the palette of physical input devices and interaction techniques supported by MaggLite is very large and easily extensible due to the fine-grained architecture of interaction-graphs.

Post-WIMP Toolkits

Most "truly" post-WIMP toolkits are specialized into specific interaction paradigms. Satin [16] for example, is a Post-WIMP toolkit which extends Swing for handling advanced gesture-based techniques. Unfortunately, it only uses standard mouse events and does not handle stylus pressure and high-resolution strokes as MaggLite does with ICON advanced input handling. We nevertheless reused Satin's gesture interpreters in ICON, which allows us to benefit from advantages of both tools. Jazz [7] is a Java toolkit for developing zoomable interfaces. As with MaggLite, user interfaces are described using scene graphs, at a fine level of granularity compared to monolithic widgets. Ubit [25] also uses scene graphs but with a "molecular" architectural metaphor: basic graphical objects (atoms) are assembled to build widgets (molecules). User interfaces are scripted rather than programmed but prototyping capabilities are limited as no graphical editor is available. Like CPN Tools [3] and MMM [8], Ubit handles multiple pointing devices but is not aware of other devices and techniques. More generally, all previous Post-WIMP graphical toolkits support a limited set of input devices and use them quite efficiently but in ad-hoc ways.

Multi-Device Toolkits

Recent work has involved research on multi-device toolkits, especially in the fields of ubiquitous computing and augmented reality/virtuality [28,14,1]. So far, these approaches focused on providing unified and flexible access to a large number of devices while relying on minimalist interaction models. The Phidgets / WidgetTaps library [14] allows binding widgets to their physical counterparts. The "Patch-Panel" of the iStuff toolkit [1] allows similar device/function bindings, with support for "translations" such as domain transformations. The "dynamic connection" approach used in these tools is similar to MaggLite's. Still, interaction in most GUIs can not be merely described as button/command bindings and scaling functions. In fact, being able to adapt efficiently any device to any task requires using advanced interaction techniques, which in turn requires much more power and flexibility in tools. From this point of view, MaggLite goes one step beyond existing tools and shows how power and flexibility can be achieved by describing input with interaction-graphs which can be connected to scene graphs at a fine-grained level.

Interface Builders

Classical interface builders such as Visual Basic have two important drawbacks: first, they are strongly limited to WIMP interfaces. Second, while the UI appearance can be built graphically, the behaviors (including interaction) must be programmed. A number of more advanced UI building and prototyping tools have been proposed, especially with the Garnet/Amulet toolkit [26]. Most of them, like Lapidary [29], are graphical interface builders which use "programming by demonstration" for specifying behavioral aspects. Although powerful, this paradigm raises a number of issues that have not been addressed yet, especially when non-trivial behaviors have to be specified. Silk [24,23] is a UI prototyping tool also related to Garnet, but which uses gestures. It is a complete UI design environment with a lot of capabilities such as sketch editing, history, annotations and storyboards to specify behaviors. But UIs elements are limited to standard Visual Basic or CommonLisp widgets. Moreover, the interfaces cannot be fully tested while being built, as with MaggLite.

Graphical Behavior Editors

Constraint-based editors such as Thinglab [9] or Fabrik [20] allow specifying some behavioral parts of interactive applications graphically, mainly for describing geometrical layouts behaviors. Control-flow approaches such as ICO/PetShop [2] use Petri Nets or State-Transition Diagrams to describe control-intensive, highly modal parts of interactive applications. Dataflow-based editors have been used in various domains. For example, Max/MSP [27] is a widely-used graphical dataflow programming environment for musical applications. Though well-adapted for midi, audio and image real-time processing with simple standard widgets, it is not aimed at describing advanced interactions. Application of dataflow-based editors to interaction specification has been rarely exploited outside the area of 3D authoring and animation. Virtools Dev [30] uses a dataflow editor for specifying 3D input techniques interactively. Jacob's VRED system [21] uses both a control-flow (state transition diagrams) and a dataflow editor to describe discrete and continuous aspects of 3D interaction. The dataflow approach has proved quite promising for describing techniques making use of multiple input devices, but as far as we know and if we except MaggLite, the only attempt to use it for describing 2D interaction has been Whizz'Ed [12]. This notation has been successfully used to specify animation and some bimanual techniques, though other techniques and input devices have not been investigated before ICON and MaggLite.

Related Interaction Techniques

Local Tools [6] describe an alternative to tool palettes in which "each tool can be picked up (where it replaces the cursor), used, and then put down anywhere on the work surface". The KidPad [11] application uses Local Tools as well as the MID library [17] for handling multiple mice, thus allowing using multiple tools at the same time. Those features are close from MaggLite's, although it was not possible until now to freely associate physical devices with tools, nor to support more advanced devices such as graphical tablets or 3D isometric controllers.

CONCLUSION

We presented a new Post-WIMP user interface toolkit called MaggLite. Based on a novel *mixed-graph* model, MaggLite successfully separates graphical and interactions parts of user interfaces. The same way as scene-graph model break monolithic graphical architectures, interaction-graphs split the heavy structure of events handling in fine-grained data-flow processing devices. This architecture allows MaggLite to improve post-WIMP interfaces design and use in terms of:

- *Extensibility*, as programmers can easily extend the toolkit with new graphical components and add support for new input devices and pluggable interaction techniques without heavy coding.

- *Flexibility*, as UI designers can quickly prototype novel interfaces by using the sketch-based interface builder and the advanced interaction techniques available as pluggable components. As far as we know, MaggLite is the first UI toolkit that brings together such advanced interaction techniques in a fully input-independent and application-independent way. We believe this is a first step toward widespread use of techniques that are sometimes not available for technical reasons more than usability ones.

- *Adaptability*, as interfaces developed with MaggLite are fully configurable at runtime. Users can adapt applications to their abilities and to the input devices they own.

A free distribution of MaggLite and related materials are available at the URL: http://www.emn.fr/x-info/magglite/.

ACKNOWLEDGMENTS

We would like to thank Geoffrey Subileau, Mohammad Ghoniem and Narendra Jussien for their help while writing this article.

REFERENCES

1. Ballagas, R. et al. iStuff: A Physical User Interface Toolkit for Ubiquitous Computing Environments. In *Proc. of CHI '03: Human Factors in Computing Systems* (April 2003, Ft. Lauderdale, FL), ACM/SIGCHI, 2003, pp. 537-544.

2. Bastide, R., Navarre, D. and Palanque, P. A model-based tool for interactive prototyping of highly interactive applications. In *CHI'02 extended abstracts on Human factors in Computing Systems*, ACM/SIGCHI, 2002, pp. 516-517.

3. Beaudouin-Lafon, M. et al. CPN/Tools: A Post-WIMP Interface for Editing and Simulating Coloured Petri Nets. In *Proc. of ICATPN'2001: 22nd International Conference on Application and Theory of Petri Nets* (June 2001, Newcastle upon Tyne, England), Lecture Notes in Computer Science, Springer-Verlag, 2001, pp. 71-80.

4. Beaudouin-Lafon, M. Instrumental Interaction: an Interaction Model for Designing Post-WIMP User Interfaces. In

Proc. of CHI 2000: Human Factors in Computing Systems (2000, La Haye, Netherlands), ACM/SIGCHI, 2000, pp. 446-453.

5. Beaudouin-Lafon, M. Novel Interaction Techniques for Overlapping Windows. In *Proc. of UIST 2001: 14th ACM Symposium on User Interface Software and Technology* (2001, Orlando, FL), ACM/SIGCHI, 2001, pp. 153-154.

6. Bederson, B.B. et al. Local Tools: An Alternative to Tool Palettes. In *Proc. of UIST'96: 9th ACM Symposium on User Interface and Software Technology* (November 1996, Seattle, WA), ACM/SIGCHI, 1996, pp. 169-170.

7. Bederson, B., Meyer, J. and Good, L. Jazz: An Extensible Zoomable User Interface Graphics Toolkit in Java. In *Proc. of UIST 2000: 13th ACM Symposium on User Interface and Software Technology* (November 2000, San Diego, CA), ACM/SIGCHI, 2000, pp. 171-180.

8. Bier, E.A. and Freeman, S. MMM: A User Interface Architecture for Shared Editors on a Single Screen. In *Proc. of UIST'91: 4th ACM Symposium on User Interface Software and Technology* (1991, Hilton Head, SC), ACM/SIGCHI, 1991, pp. 79-86.

9. Borning, A. Thinglab - *A Constraint-Oriented Simulation Laboratory*. PhD thesis, Stanford University, July 1979. Also available as STAN-CS-79-746 Stanford Computer Science Department technical report.

10. Dragicevic, P. and Fekete, J.D. Input Device Selection and Interaction Configuration with ICON. In *Proc. of IHM-HCI 2001, People and Computers XV - Interaction without Frontiers* (September 2001, Lille, France), Springer Verlag, 2001, pp. 543-448.

11. Druin, A. et al. KidPad: A Design Collaboration Between Children, Technologists, and Educators. In *Proc. of CHI'97: Human Factors in Computing Systems* (March 1997, Atlanta, GA), ACM/SIGCHI, 1997, pp. 463-470.

12. Esteban, O., Chatty, S. and Palanque, P. Whizz'ed : a visual environment for building highly interactive software. In *Proc. of INTERACT'95: 5th IFIP International Conference on Human-Computer Interaction* (June 1995, Lillehammer, Norway), IOS Press, 1995, pp. 121-126.

13. Fekete, J.D. The InfoVis Toolkit. In *Proc. of InfoVis'04: 10th Symposium on Information* Visualization (October 2004, Austin, TX), IEEE Press, 2004.

14. Greenberg, S. and Fitchett, C. Phidgets: Easy Development of Physical Interfaces through Physical Widgets. In *Proc. of UIST 2001: 14th ACM Symposium on User Interface Software and Technology* (November 2001, Orlando, FL), ACM/SIGCHI, pp. 209-218.

15. Henry, T.R., Hudson, S.E. and Newell, G.L. Integrating Gesture and Snapping into a User Interface Toolkit. In *Proc. of UIST'90: 3rd ACM SIGGRAPH Symposium on User Interface Software and Technology* (1990, Snowbird, UT), ACM Press, pp. 112-121.

16. Hong, J. and Landay, J. SATIN: A Toolkit for Informal Ink-based Applications. In *Proc. of UIST 2000: 13th ACM Symposium on User Interface Software and Technology* (November 2000, San Diego, CA), ACM/SIGCHI, 2000, pp. 63-72.

17. Hourcade, J.P. and Bederson, B.B. Architecture and Implementation of a Java Package for Multiple Input Devices (MID), Human-Computer Interaction Laboratory, University of Maryland, College Park, MD 20742, USA, 1999.

18. Hudson, S.E. and Smith, I. Ultra-Lightweight Constraints. In *Proc. of UIST'96: 9th ACM Symposium on User Interface and Software Technology* (November 1996, Seattle, WA), ACM/SIGCHI, 1996, pp. 179-187.

19. Huot, S., Dumas, C. and Hégron, G. Svalabard: A Virtual Drawing Table for 3D Modeling. In *Proc. of IHM'04: 16th French-Speaking Conference on Human-Computer Interaction* (August 2004, Namur, Belgium), ACM press, 2004.

20. Ingalls, D. et al. Fabrik: A Visual Programming Environment. In *Proc. of OOPSLA'88: Object-Oriented Programming Systems, Languages and Applications* (1988, San Diego, CA), SIGPLAN Notices 23(11), 1998, pp 176-190.

21. Jacob, R., Deligiannidis, L. and Morrison, S. A Software Model and Specification Language for Non-WIMP User Interfaces. *ACM Transactions on Computer-Human Interaction*, 6(1):1-46, march 1999.

22. Krasner, G.E. and Pope, S.T. A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk 80 System, *J. Object Oriented Programming*, 1(3):26-49 (1988).

23. Landay, J.A. and Myers B.A. Interactive Sketching for the Early Stages of User Interface Design. In *Proc. of CHI '95: Human Factors in Computing Systems* (May 1995, Denver, CO), ACM/SIGCHI, pp. 43-50.

24. Landay, J.A. SILK: Sketching Interfaces Like Krazy. *Technical Video Program of CHI '96* (April 1996).

25. Lecolinet, E. A molecular architecture for creating advanced interfaces. *CHI Letters*. pp. 135-144. ACM Press 2003.

26. Myers, B.A. A New Model for Handling Input. In *ACM Transactions on Information Systems*, 8(3):289-320, 1990.

27. Puckette, M. Combining Event and Signal Processing in the MAX Graphical Programming Environment, *Computer Music Journal*, 15(3):50-57, 1991.

28. Salber, D., Dey, A.K. and Abowd, G.D. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *Proc. of CHI'99: Human Factors in Computing Systems* (1999, Pittsburgh, PA), ACM/SIGCHI, 1999, pp. 434-441.

29. Vander Zanden, B.T. and Myers, B.A. Demonstrational and Constraint-Based Techniques for Pictorially Specifying Application Objects and Behaviors. In *ACM Transactions on Computer Human Interaction*, 2(4):308-356, Dec. 1995.

30. Virtools dev. Virtools SA, 2001. http://www.virtools.com/.

Excentric Labeling: Dynamic Neighborhood Labeling for Data Visualization

Jean-Daniel Fekete
Ecole des Mines de Nantes
4, rue Alfred Kastler, La Chantrerie
44307 Nantes, France
Jean-Daniel.Fekete@emn.fr

Catherine Plaisant
Human-Computer Interaction Laboratory
University of Maryland
College Park, MD 20782, USA
Plaisant@cs.umd.edu

ABSTRACT
The widespread use of information visualization is hampered by the lack of effective labeling techniques. A taxonomy of labeling methods is proposed. We then describe "excentric labeling", a new dynamic technique to label a neighborhood of objects located around the cursor. This technique does not intrude into the existing interaction, it is not computationally intensive, and was easily applied to several visualization applications. A pilot study indicates a strong speed benefit for tasks that involve the rapid exploration of large numbers of objects.

KEYWORDS
Visualization, Label, Dynamic labeling

INTRODUCTION
A major limiting factor to the widespread use of information visualization is the difficulty of labeling information abundant displays. Information visualization uses the powerful human visual abilities to extract meaning from graphical information [Card et al, 1998, Cleveland, 1993]. Color, size, shape position or orientation are mapped to data attributes. This visualization helps users find trends, and spot exceptions or relationships between elements on the display. Experimental studies have been able to show significant task completion time reduction and recall rate improvements when using graphical displays instead of tabular text displays (e.g., [Lindwarm-Alonso et al., 1998.]) However textual information in the form of labels remains critical in identifying elements of the display. Unfortunately, information visualization systems often lack adequate labeling strategies. Often labels are entirely missing and users have to peck at graphical objects one at a time. Sometimes labels overlap each other to the point of obscuring the data and being barely usable; or they are spread out in such a way that the relation between objects and labels becomes ambiguous. The problem becomes acute when the data density increases and the labels are very long.

To address this problem we propose "excentric labeling" as a new dynamic technique to label a neighborhood of objects (Figure 1 to 3). Because it does not interfere with normal interaction and has a low computational overhead, it can easily be applied to a variety of visualization applications.

The labeling problem is not new. It has been extensively studied for cartographic purposes [Christensen et al., 1998] where printing or report generation is the main purpose of the application. But very few solutions have been proposed to automate the labeling process of interactive applications. In this paper we propose a taxonomy of labeling methods, then describe our excentric labeling technique in detail, discuss its benefits and limitations, and illustrate how it can benefit a variety of applications.

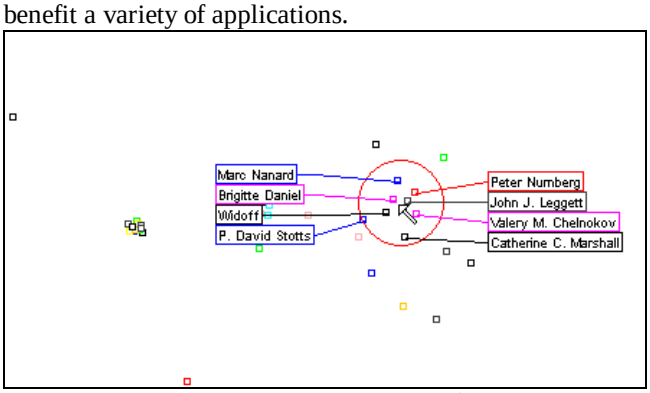


Figure 1: Excentric labeling provides labels for a neighborhood of objects. The focus of the labeling is centered on the cursor position. Labels are updated smoothly as the cursor moves over the display, allowing hundreds of labels to be reviewed in a few seconds. The color of the label border matches the object color.

HCIL Technical Report 99-09 (December 1998) ;
http://www.cs.umd.edu/hcil
CHI'99 Pittsburgh, PA, USA, May 15-20, 1999, 512-519.

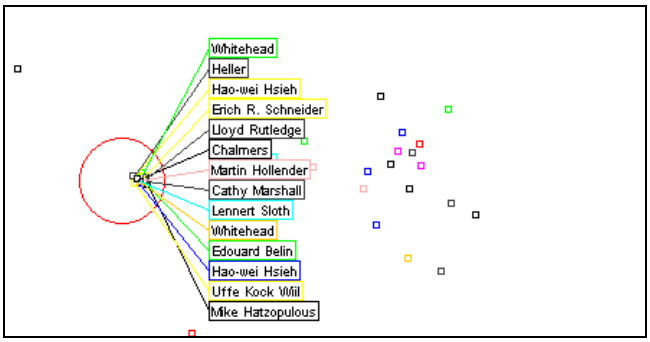


Figure 2: Labels are spread to avoid overlapping, possibly revealing objects clumped together on the display.

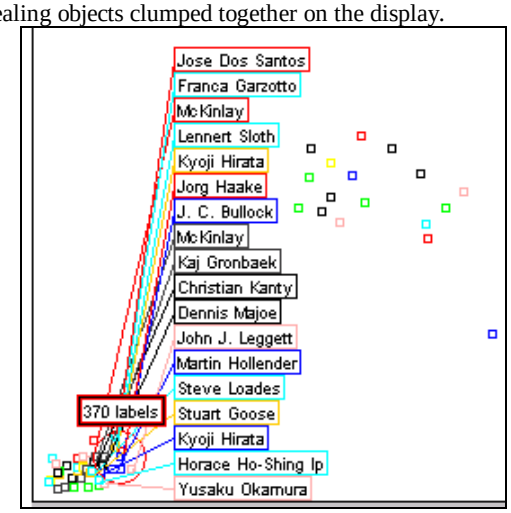


Figure 3: Special algorithms handle border effects (e.g., corners) When objects are too numerous, the total number of objects in the focus area is shown, along with a few sample labels.

TAXONOMY OF LABELING TECHNIQUES

The labeling challenge can be stated as follows: given a set of graphical objects, find a layout to position all names so that each name (label) is:

1. Readable.
2. Non-ambiguously related to its graphical object.
3. Does not hide any other pertinent information.

Completeness (the labeling of all objects) is desired but not always possible.

Labeling techniques can be classified into two categories: static and dynamic. The goal of static labeling is to visually associate labels with a maximum (hopefully all) graphic objects in the best possible manner. But good static technique are usually associated with delays not suitable for interactive exploration. Dynamic labeling began with interactive computer graphics and visualization. Two attributes account for the “dynamic” adjective: the set of objects to be labeled can change dynamically, and the number and layout of displayed labels can also change in real time, according to user actions.

Static Techniques

Static techniques have been used for a long time in cartography. Christensen et al. (to appear) wrote a recent summary of label placement algorithms. Cartography also needs to deal with path labeling and zone labeling, which is less widespread in visualization. We do not address those two issues in this article. But the same algorithms can be used for both cartography and general visualization. Since static techniques have to find "the" best labeling possible, the set of objects has to be carefully chosen to avoid a too high density in objects or labels. In cartography, this is achieved by aggregating some information and forgetting (sampling) others (this process is called "generalization"). This technique could be nicknamed the "label-at-all-cost" technique since one of the constraints is to label all objects of the display.

For data visualization, a similar process of aggregation can be applied to achieve a reasonable result with static techniques (e.g., aggregation is used in the semantic zooming of Pad++ [Bederson, 1994] or LifeLines [Plaisant et al., 1998]), but the logic of aggregation and sampling is mainly application dependent. Label sampling has been used occasionally (e.g., Chalmers et al., 1996).

The most common techniques remain the "No Label" technique, and the "Rapid Label-all" technique which leads to multiple overlaps and data occlusion [e.g., in the hyperbolic browser [Lamping et al, 1995]). Also common is the "Label-What-You-Can" technique in which only labels that fit are displayed; other labels that would overlap or occlude data objects are not shown (e.g., in LifeLines).

Some visualizations avoid the problem completely by making the labels the primary objects. For example, WebTOC [Nation et Al, 1997] uses a textual table of contents and places color and size coded bars next to each label.

Dynamic techniques

Dynamic labeling techniques are more varied (see Table 1). The classic infotip or "cursor sensitive balloon label" consists at showing the label of an objet right next to the object when the cursor passes over it. The label can also be shown on a fixed side window, which is appropriate when labels are very long and structured.

In the "All or Nothing" technique, labels appear when the number of objects on the screen falls below a fixed limit (e.g., 25 for the dynamic query and starfield display of the film finder [Ahlberg et al., 94]). This is acceptable when the data can be easily and meaningfully filtered to such a small subset, which is not always the case. Another strategy is to require zooming until enough space is available to reveal the labels, which requires extensive navigation to see all labels. This technique can be combined elegantly with the static aggregation technique to progressively reveal more and more details - and refined labels - as the zoom ratio increases.

The overview and detail view combination is an alternative zooming solution [Plaisant et al., 1994]. The detail view can also be deformed to spread objects until all labels fit (i.e., in the way of a labeling magic lens). Those last two techniques require either a tool selection or dedicated screen space.

Chalmers et al., proposed dynamic sampling where only one to three labels are displayed, depending on the user's activity. Cleveland describes temporal brushing: labels appear as the cursor passes over the objects (similarly to the infotip), but those labels remain on the screen while new labels are displayed, possibly overlapping older ones.

Type	Technique	Comments/Problems
STATIC	No label	No labels!
	Label-only-when-you-can (i.e. after filtering objects)	Need effective filters. Labels are rarely visible.
	Rapid Label-All	High risk of overlaps or ambiguous linking to objects
	Optimized Label-All	Often slow - may not be possible
	Optimized Label-All with aggre-gation and sampling	Effective but application dependant- may not be possible
DYNAMIC		
One at a time	Cursor sensitive balloon label	Requires series of precise selection to explore space (slow), cannot reach overlapped objects.
	Cursor Sensitive label in side-window	Same as above. Constant eye movement can be a problem, but avoids occlusion of other objects.
	Temporal brushing (Cleveland)	More labels visible at a time, but overlapping problem.
Global display change	Zoom until labels appear	May require extensive navigation to see many labels (can be effectively combined with semantic zooming, e.g., Pad++)
	Filter until labels appear	May require several filtering to see labels (can be effectively combined with Zooming, e.g., starfields)
Focus + context	Overview and detail view without deformation	Effective when objects are separated enough in the detail view to allow labels to fit (not guaranteed.)
	Overview and detail with deformation/ transformation (i.e.fisheye or magic lenses)	Deformation might allow enough room for labels to fit. (not guaranteed). May require tool or mode to be selected.
	Global deformation of space (e.g., Hyperbolic Browser)	Requires intensive navigation and dexterity to rapidly deform the space and reveal all labels (e.g., by fanning the space).
Sampling	Dynamic sampling (Chalmers et al.)	Few labels are visible.
NEW	Excentric labeling	Fast, no tool or special skill needed. Spread overlapping labels, and align them for ease of reading.

Table 1: Taxonomy of labeling techniques

EXCENTRIC LABELING

Excentric labeling is a dynamic technique of neighborhood labeling for data visualization (Figure 1 to 3). When the cursor stays more than one second over an area where objects are available, all labels in the neighborhood of the cursor are shown without overlap, and aligned to facilitate rapid reading. A circle centered on the position of the cursor defines the neighborhood or focus region. A line connects each label to the corresponding object. The style of the lines matches the object attributes (e.g., color). The text of the label always appears in black on a white background for better readability. Once the excentric labels are displayed, users can move the cursor around the window and the excentric labels are updated dynamically. Excentric labeling stops either when an interaction is started (e.g., a mouse click) or the user moves the cursor quickly to leave the focus region. This labeling technique does not require the use of special interface tool. Labels are readable (non overlapping and aligned), they are non-ambiguously related to their graphical objects and they don't hide any information inside the user's focus region.

Algorithm and Variations

To compute the layout of labels, we experimented with several variants of the following algorithm:

1. Extract each label and position for interesting graphic objects in the focus region.
2. Compute an initial position.
3. Compute an ordering.
4. Assign the labels to either a right or left set.
5. Stack the left and right labels according to their order.
6. Minimize the vertical distance of each set from the computed initial position.
7. Add lines to connect the labels to their related graphic object.

So far, we have used three main variations of this algorithm: non-crossing lines labeling, vertically coherent labeling and horizontally coherent labeling (the last two can be combined). Each uses a different method to compute the initial position, the ordering, to assign the labels to the stacks and to join the labels to their related graphic objects.

Non-Crossing Lines Labeling – Radial Labeling

The non-crossing lines labeling layout (Figure 4) does not maintain the vertical or horizontal ordering of labels, but avoids line crossings. This technique facilitates the task of tracing the label back to the corresponding object. It can be used in cartography-like applications where ordering is unimportant. The initial position on the circle (step 2 of

previous section) is computed with a radial projecting onto the circumference of the focus circle. It is always possible to join the object to the circumference without crossing another radial spoke (but two radii - or spokes- may overlap). Then, we order spokes in counter-clockwise order starting at the top (step 3). The left set is filled with labels from the top to the bottom and the right set is filled with the rest.

Labels are left justified and regularly spaced vertically. We maintain a constant margin between the left and right label blocks and the focus circle to draw the connecting lines.

For the left part, three lines are used to connect objects to their label: from the object to the position on the circumference, then to the left margin, and to the right side of the label box. This third segment is kept as small as possible for compactness, therefore barely visible in Figure 4, except for the bottom-left label. For the right labels, only two lines are used from the object to the initial position to the left of the label. The margins contain the lines between the circumference and the labels.

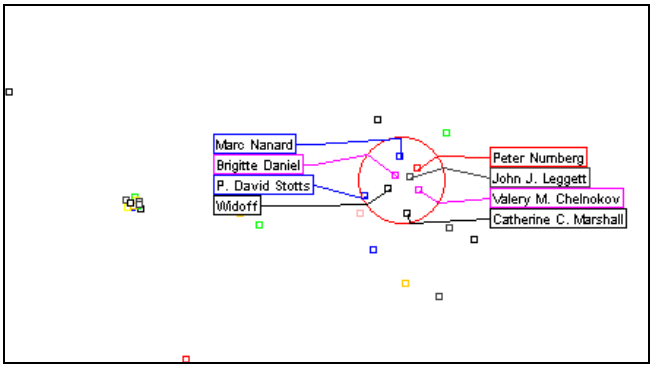


Figure 4: This figure shows the same data as in Figure 1 but using the non-crossing - or radial - algorithm.

Vertically Coherent Labeling

When the vertical ordering of graphic objects has an important meaning we use a variant algorithm that does not avoid line crossing but maintains the relative vertical order of labels. This will be appropriate for most data visualization, for example, in the starfield application FilmFinder [Ahlberg, 1994], films can be sorted by attributes like popularity or length, therefore labels should probably be ordered by the same attribute. Instead of computing the initial position in step 2 by projecting the labels radially to the circumference, we start at the actual Y position of the object. The rest of the algorithm is exactly the same. Figure 1 and 2 shows examples using the vertically coherent algorithm, which is probably the best default algorithm. Crossing can occur but we found that moving slightly the cursor position animates the label connecting lines and helps find the correspondence between objets and their labels.

Horizontally Coherent Labeling

When the horizontal ordering of graphic objects has a special meaning, we further modify the algorithm in step 5. Instead of left justifying the labels, we move them horizontally, so that they follow the same ordering as the graphic objects in Figure 5.

Dealing with window boundaries

When the focus region is near the window boundaries, chances are that the label positions computed by the previous algorithms will fall outside of the window and the labels appear truncated (e.g., the first characters of the left stack labels would not be visible when the cursor is on the left side of the window).

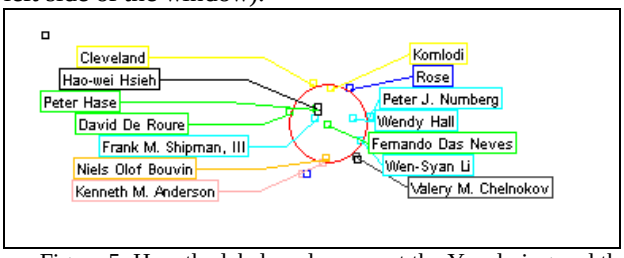


Figure 5: Here the labels order respect the Y ordering and the indentation of the labels reflects the X ordering of the objects.

To deal with window boundaries the following rules are applied. If some labels are cut on the left stack, then move them to the right stack (symmetric for the right side.) When labels become hidden on the upper part of the stack (i.e., near the upper boundary), move them down (symmetric for the bottom). Combining those rules takes care of the corners of the window (Figure 6).

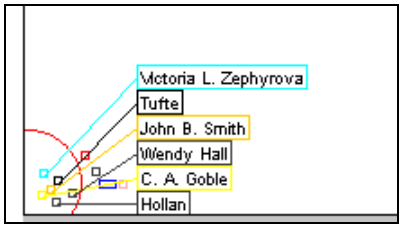


Figure 6: When the focus is close to the window boundaries, labels are moved so that they always fall inside the window.

DISCUSSION

Excentric labeling fills a gap in information visualization techniques by allowing the exploration of hundreds of labels in dense visualization screens in a matter of seconds. Many labels can be shown at once (optimally about 20 at a time.) They are quite readable and can be ordered in a meaningful way. Links between objects and labels remain apparent. The technique is simple and computationally inexpensive enough to allow for smooth exploration while labels are continuously updated. Of course these algorithms don't solve all the problems that may occur when labeling. Three important challenges remain, and we propose partial solutions for them:

Dealing with too many labels

We estimate that about 20 excentric labels can reasonably be displayed at a time. When more objects fall in the focus region, the screen becomes filled by labels and there is often no way to avoid that some labels fall outside the window. We implemented two "fallback" strategies: (1) showing the number of items in the focus region, and (2) showing a

sample of those labels in addition to the number of objects (see Figure 3). The sample could be chosen randomly or by using the closest objects to the focus point. Although not entirely satisfactory, this method is a major improvement over the usual method of showing no labels at all, or a pile of overlapping labels.

The dynamic update of this object counts allows a rapid exploration of the data density on the screen. Of course (this is data visualization after all) the number of objets could also been shown graphically by changing the font or box size to reflect its level of magnitude.

Dealing with long labels

Labels can be so long that they just don't fit on either side of the focus point. There is no generic way to deal with this problem but truncation is likely to be the most useful method. Depending on the application, labels may be truncated on the right, or on the left (e.g., when the labels are web addresses), or they may be truncated following special algorithms. Some applications may provide a long and a short label to use as a substitute when needed (e.g., Acronyms). Using smaller fonts for long labels might help in some cases. If long lines occur infrequently, breaking long labels in multiple lines is also possible.

Limiting discontinuities

One of the drawbacks of the dymamic aspect of excentric labeling is that the placement of an object's label will vary while the cursor is moving around the object. This is needed to allow new labels to be added when the focus area covers more objects, but can lead to discontinuities in the placement of labels. For example when the cursor moves from the left side of an object to its right side, the label will move from the right to the left stack. This effect is actually useful to confirm the exact position of a label but might be found confusing by first time users. We found that discontinuties were more common with the non-crossing algorithm than the Y coherent algorithm which we favor despite the risk of lines crossing.

POSSIBLE IMPROVEMENTS

Depending on the application, several improvements might be considered :

- Changing the size and shape of the focus area can be allowed, either at the user's initiative, or dynamically as a function of the label density;
- When too many items are in the focus area, excentric labels can show not only the number of objects but also a glyph or bar chart summarizing the contents of the area (e.g., showing the color distribution of the points in the focus).
- Labels can inherit more graphic attributes from the objects they reference, as is often done in cartography. We currently map the color of the label border to the object's color. But text font size, style or color can also be used if clear coding conventions exist and if adequate readability is preserved.
- Excentric labels can easily be used as selection menus. For example pressing a control key can temporarily "freeze" the excentric labeling, free the cursor, allowing users to select any of the labels.

USE WITHIN EXISTING VISUALIZATION APPLICATIONS

We have implemented excentric labels within three different applications: a java version of starfield display/dynamic query visualization [Ahlberg et al, 1994] (Figure 7), a Java implementation of LifeLines (Figure 9), and a map applet to be used for searching people in a building. The addition of excentric labeling to the first two applications was done in a few hours. The last program was built from scratch as an evaluation tool.

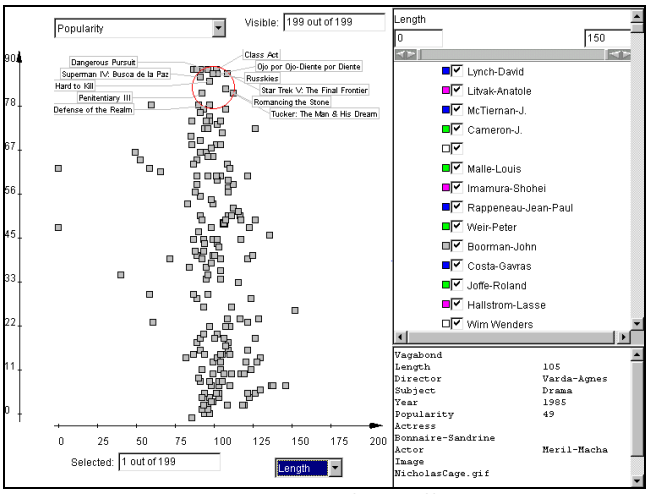


Figure 7: Excentric labeling was found effective in a java implementation of a starfield/dynamic query environment. It provides a rapid way to review the names of the data objects and to fathom the density of the overlapping areas.

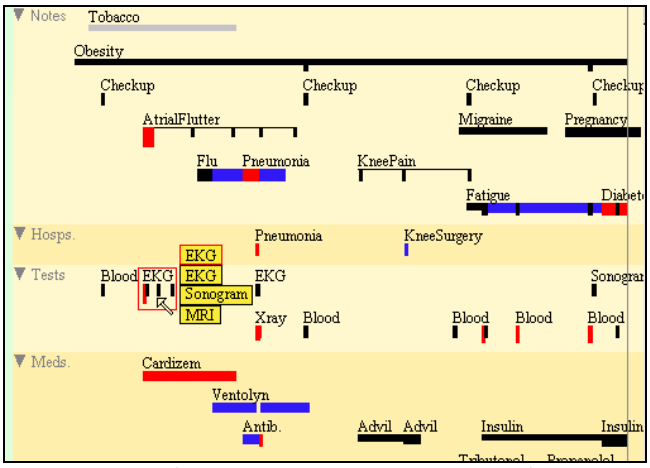


Figure 8: In LifeLines excentric labeling can be useful as it guarantees that all events in the focus are labeled, even if events overlap. Chronological order is best for ordering labels. In this example the focus area is rectangular (i.e., a time range) and no connecting lines are used. The label background is yellow to make them more visible.

EARLY EVALUATION

We are in the process of comparing excentric labeling with a purely zoomable interface. The map of a building is displayed with workers names assigned randomly to offices. Subjects have to figure out if a given person is assigned to a room close to one of three red dots shown on the map (symbolizing three area of interest that a visualization would have revealed, e.g., areas close to both vending machines and printers). Each subject has to repeat the task ten times with new office assignments and red dot locations. The questions asked are of the form: "is <the person> in the neighborhood of one of the red dots?" Subjects reply by selecting "yes" or "no". The time to perform each task and the number of errors are recorded. Subjects using excentric labels (Figure 9) have to move the cursor over and around each highlighted point and read the labels. Subjects using the zooming interface have to move the cursor over each highlighted point, left click to zoom until they can read the labels (one or two zoom operation), right click to zoom back out or pan to the next point.

Our initial test of the experiment highlighted how speed and smoothness of zooming is crucial for zooming interfaces. In our test application a zoom or pan takes about 3/4 seconds to redraw. This is representative of many zooming interfaces, but in order to avoid any bias in favor of the excentric labeling we chose to ignore the redisplay time (the clock is stopped during redraws in the zooming interface version).

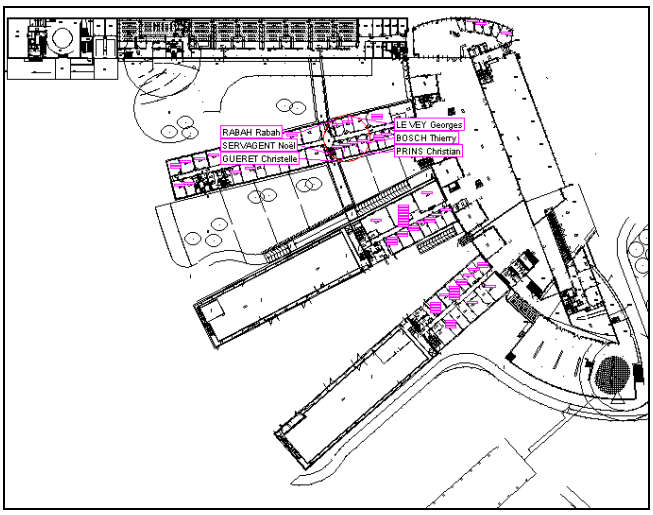


Figure 9: Map with a section of the building dynamically labeled. This application is being used to compare excentric labeling with a plain zooming interface when performing tasks that require the review of many labels.

Initial results of the pilot test (with 6 subjects repeating the task 3 times with each interface) shows that users performed the task 4 times faster when using the excentric labeling than with the zooming interface. Any delay in zooming and panning would further increase the effect in favor of excentric labeling. Our informal observations suggest that users sometime get lost using the zoom and pan, which does not happen when using the excentric labeling. On the other hand some subjects commented on the discontinuity problem.

CONCLUSION

Despite the numerous techniques found in visualization systems to label the numerous graphical objects of the display, labeling remains a challenging problem for information visualization. We believe that excentric labeling provides a novel way for users to rapidly explore objects descriptions once patterns have been found in the display and effectively extract meaning from information visualization. Early evaluation results are promising, and we have demonstrated that the technique can easily be combined with a variety of information visualization applications.

ACKNOWLEDGEMENT

This work was mainly conducted while Jean-Daniel Fekete visited Maryland during the summer 1998. We thank all members of the HCIL lab for their constructive feedback, especially Julia Li for her initial research of the labeling problem, and Ben Shneiderman for suggesting the main-axis projection. This work was supported in part by IBM through the Shared University Research (SUR) program and by NASA (NAG 52895).

REFERENCES

1. Ahlberg, Christopher and Shneiderman, Ben, Visual information seeking: Tight coupling of dynamic query filters with starfield displays, *Proc. CHI'94 Conference: Human Factors in Computing Systems*, ACM, New York, NY (1994), 313-321 + color plates.
2. Bederson, Ben B. and Hollan, James D., PAD++: A zooming graphical user interface for exploring alternate interface physics, *Proc. User Interfaces Software and Technology '94* (1994), 17-27.
3. Chalmers M., Ingram R. & Pfranger C., Adding imageability features to information displays. UIST'96, Seattle Washington USA, ACM.
4. Christensen J., Marks J., Shieber S. Labeling Point Features on Map and Diagrams, to appear in *Transactions of Graphics.*
5. Cleveland, William, *Visualizing Data*, Hobart Press, Summit, NJ (1993).
6. Card, S, Mackinlay, J., and Shneiderman, Ben, *Readings in Information Visualization: Using Vision to Think*, Morgan Kauffman Publishers, to appear.
7. Cleveland, William, *Visualizing Data*, Hobart Press, Summit, NJ (1993).
8. Lamping, John, Rao, Ramana, and Pirolli, Peter, A focus + context technique based on hyperbolic geometry for visualizing large hierarchies, *Proc. of ACM CHI'95 Conference: Human Factors in Computing Systems*, ACM, New York, NY (1995), 401-408
9. Lindwarm D., Rose, A., Plaisant, C., and Norman, K., Viewing personal history records: A comparison of tabular format and graphical presentation using LifeLines, *Behaviour & Information Technology* (to appear, 1998).

10. Nation, D. A., Plaisant, C., Marchionini, G., Komlodi, A., Visualizing websites using a hierarchical table of contents browser: WebTOC, *Proc. 3rd Conference on Human Factors and the Web*, Denver, CO (June 1997).
11. Plaisant, C., Carr, D., and Shneiderman, B., Image-browser taxonomy and guidelines for designers, *IEEE Software 12*, 2 (March 1995), 21-32.
12. Plaisant, Catherine, Rose, Anne, Milash, Brett, Widoff, Seth, and Shneiderman, Ben, LifeLines: Visualizing personal histories, *Proc. of ACM CHI96 Conference: Human Factors in Computing Systems*, ACM, New York, NY (1996), 221-227, 518.
13. Plaisant, C., Mushlin, R., Snyder, A., Li, J., Heller, D., and Shneiderman, B.(1998), LifeLines: Using visualization to enhance navigation and analysis of patient records, to appear *in Proc. of American Medical Informatics Association Conference*, Nov. 1998, AMIA, Bethesda, MD.

DEMONSTRATION

Excentric labeling is implemented in Java. A demo program has been placed in a neutral site for demonstration.

URL: http://www-ihm.lri.fr/excentric/

Readability of Graphs Using Node-Link and Matrix-Based Representations: Controlled Experiment and Statistical Analysis

Mohammad Ghoniem

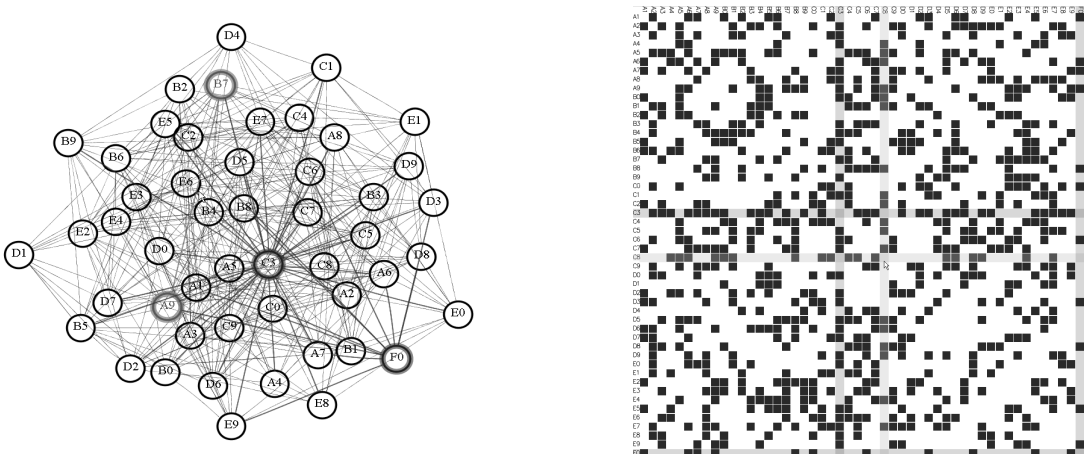
Ecole des Mines de Nantes
4 rue Alfred Kastler. B.P.20722
44307 NANTES Cedex 3
Mohammad.Ghoniem@emn.fr

Jean-Daniel Fekete

INRIA Futurs/LRI
Bât 490, Université Paris-Sud
91405 Orsay Cedex
Jean-Daniel.Fekete@inria.fr

Philippe Castagliola

IRCCyN/IUT de Nantes
Rue Christian Pauc - La Chantrerie
BP 50609 - 44306 Nantes Cedex 3
philippe.castagliola@univ-nantes.fr



a ***b***

Figure 1: Two visualizations of the same undirected graph containing 50 vertices and 400 edges. The node-link diagram a) is computed using the "neato" program and the matrix representation b) is computed using our VisAdj program.

ABSTRACT

In this article, we describe a taxonomy of generic graph related tasks along with a computer-based evaluation designed to assess the readability of two representations of graphs: matrix-based representations and node-link diagrams. This evaluation encompasses seven generic tasks and leads to insightful recommendations for the representation of graphs according to their size and density. Typically, we show that when graphs are bigger than twenty vertices, the matrix-based visualization outperforms node-link diagrams on most tasks. Only path finding is consistently in favor of node-link diagrams throughout the evaluation.

KEYWORDS

Visualization of graphs, adjacency matrices, node-link representation, readability, evaluation.

1 INTRODUCTION

Node-link diagrams have often been used to represent graphs. In the graph drawing community, many publications deal with layout techniques complying with aesthetic rules such as minimizing the number of edge-crossings, minimizing the ratio between the longest edge and the shortest edge, and revealing symmetries [2]. Most works strive to optimize algorithms complying with such rules, but they rarely try and validate them from a cognitive point of view. Recently, Purchase et al. tackled this problem through on-paper [13] and online [11, 12] experiments. These works involved small handcrafted graphs (graphs with 20 vertices and 20 to 30 edges), five aesthetic criteria and eight graph layout algorithms. The authors point out that while some aesthetic criteria taken separately may improve the perception of the graph at hand, it is difficult to claim that an algorithm brings about such an improvement. Moreover, Ware and Purchase set up a study aimed at the validation of some aesthetic properties put forth in the graph drawing community, such as the influence of good continuity on the perception of paths [13]. In the Information Visualization (Infovis) community, many node-link variants have been experimented, both in 2D and 3D [5, 8]. However, as soon as the size of the graph or the link density increases, all these techniques face occlusion problems due to links overlapping (Figure 1a). Thus, it becomes difficult for users to visually explore the graph or interact with its elements.

Conversely, matrix-based visualizations of graphs eliminate altogether occlusion problems and provide an outstanding potential, despite their lack of familiarity to most users. In this paper, we present an evaluation comparing the two representations in order to show their respective advantages with regard to a set of generic analysis tasks.

TO APPEAR IN "JOURNAL OF INFORMATION VISUALIZATON" 2005.

2 THE MATRIX-BASED VISUALIZATION OF GRAPHS

The matrix-based visualization of graphs (Figure 1b) relies from a formal standpoint on the potential representation of a graph via its boolean-valued connectivity matrix where rows and columns represent the vertices of the graph. Although conventions may vary for directed graphs, columns and rows generally represent the origin of edges and their endpoint vertices, respectively. When two vertices are connected, the cell at the intersection of the corresponding line and column contains the value "true". Otherwise, it takes on the value "false". Boolean values may be replaced with valued attributes associated with the edges that can provide a more informative visualization.

The matrix-based representation of graphs offers an interesting alternative to the traditional node-link diagrams. In [4], Bertin shows that it is possible to reveal the underlying structure of a network represented by a matrix through successive permutations of its rows and columns. In [3], the authors visualize the load distribution of a telecommunication network by using a matrix but their effort aims mostly at improving the display of a node-link representation such as displaying half-links or postponing the display of important links to minimize occlusion problems. More recently, in [7], the authors implemented a multi-scale matrix-based visualization that represents the call graph between software components in a large medical imagery application. In [6], we have shown that a matrix-based representation can be used to effectively grasp the structure of a co-activity graph and to assess the activity taking place across time, whereas the equivalent node-link representation was unusable. This work was specifically applied to monitoring constraint-oriented programs.

3 COMPARISON OF REPRESENTATIONS

The comparison of two visualization techniques can only be carried out for a set of tasks and a set of graphs. The list of tasks that are useful or important with regard to graph exploration seems potentially endless. For instance, this fact can be easily illustrated by constructing a Website related graph and the associated list of tasks that one can carry out or wish to carry out on such a graph. However, for tractability reasons, we tackled the problem by focusing on a set of most generic tasks of information visualization, and we adapted them to the visualization of graphs. We believe that the readability of a representation must be related to the ability of the user to answer some relevant questions about the overview. As far as graphs are concerned, some questions may bear on topological considerations or topology-related attributes. The most generic questions related to the topology of a graph – i.e. the ones independent of the semantics of data – are related to its vertices, links, paths and sub-graphs. We extracted from the "Glossary of graph theory" entry of the Wikipedia Free Encyclopedia a set of important concepts and organized basic tasks related to them as follows:

Basic characteristics of vertices: One may be interested in determining the number of vertices (their cardinality), outliers, a given vertex (by its label), and the most connected or least connected vertices.

Basic characteristics of paths: They include the number of links, the existence of a common neighbor, the existence of a path between two nodes, the shortest path, the number of neighbors of a given node, loops and critical paths.

Basic characteristics of subgraphs: One may be interested in a given subgraph, all the vertices reachable from one or several vertices (connected sets) or a group of vertices strongly connected (clusters).

Therefore, comparing the readability of graph representations should, in principle, take all these characteristics into account in order to determine the tasks that are more easily performed with a matrix-based representation and the ones for which it is more appropriate or more reasonable to use a node-link representation. This article presents a comparative evaluation of readability performed on a subset of these generic tasks.

3.1 Readability of a graph representation

The readability of a graphic representation can be defined as the relative ease with which the user finds the information he is looking for. Put differently, the more readable a representation, the faster the user executes the task at hand and the less he makes mistakes. If the user answers quickly and correctly, the representation is considered very readable for the task. If, however, the user needs a lot of time or if the answer he provides is wrong, then the representation is not well-suited for that task.

In our evaluation, we selected the following generic tasks:

- Task 1: approximate estimation of the number of nodes in the graph, referred to as "nodeCount".
- Task 2: approximate estimation of the number of links in the graph, referred to as "edgeCount".
- Task 3: finding the most connected node, referred to as "mostConnected".
- Task 4: finding a node given its label, referred to as "findNode".
- Task 5: finding a link between two specified nodes, referred to as "findLink".
- Task 6: finding a common neighbor between two specified nodes, referred to as "findNeighbor".
- Task 7: finding a path between two nodes, referred to as "findPath".

Readability also depends on the specific graph instances at hand, their familiarity to users, their meaning, and the layout used to visualize them. In our evaluation, we only compare random graphs which are meaningless and equally unfamiliar to users and hence, we only focus on abstract characteristics of graphs. We choose a popular graph layout program called "neato", part of the GraphViz [1] package to compute the node-link diagrams. It could be argued that an alternative layout program might provide a more readable layout according to our tasks. This is certainly true for actual figures but we believe that the overall trends would be similar when the size and density of graphs increase.

3.2 Preliminary Hypotheses

The traditional node-link representation suffers from link overlapping – interfering with neighborhood finding and link counting – and link length – interfering with neighborhood finding. Moreover, some tasks involving sequential search of

graph elements, such as node finding by name, are increasingly difficult when the number of nodes becomes large since, in general, nodes are not laid out in a predictive order. Hence, we expect the number of nodes and the link density to greatly influence the readability of this representation. We define the link density d in a graph as $d = \sqrt{\frac{l}{n^2}}$,

where l is the number of links and n the number of nodes in the graph. This value varies between 0 for a graph without any edge to 1 for a fully connected graph. In graph theory, the density of a graph is usually taken as the ratio of the number of edges by the number of vertices. Although topologically meaningful, this definition is not scale invariant since the number of potential edges increases in the square of the number of vertices.

3.3 Predictions

The matrix-based representation has two main advantages: it exhibits no overlapping and is orderable. We therefore expect tasks involving node finding and link finding to be carried out more easily. Counting nodes should be equally difficult on both representations, unless nodes become cluttered on node-link diagrams. Counting links should be easier on matrices since no occlusion interferes with the task. Also, finding the most connected node should perform better on matrices for dense graphs. In fact, in the context of node-link diagrams, links starting or ending at a node are hard to discriminate from links crossing the node.

On the other hand, when it comes to building a path between two nodes, node-link diagrams should perform better; matrix-based representations are more complex to apprehend because nodes are represented twice (once on both axes of the matrix), which forces the eye to move from the row representing a vertex to its associated column back and forth, unless a more appropriate interaction is provided. Lastly, we believe that node-link diagrams are suitable, and therefore preferable to the less intuitive matrix representation for small-sized graphs.

3.4 Experimental setup

3.4.1 The data

In order to test our hypotheses, we experimented with graphs of three different sizes (20 vertices, 50 vertices and 100 vertices) with three different link densities (0.2, 0.4 and 0.6) for each size, that is to say, a total of nine different graphs (Table 1). In order to avoid any bias introduced by some peculiarity of the chosen data, we opted for random undirected graphs generated by the random graph server located at the ISPT [9]. Moreover, in order to eliminate any ambiguity with regard to task 3, which consists in finding the most connected node, we added an extra 10% of links to the most connected node in these graphs. When several nodes had initially the highest degree, one of them was chosen at random and received an additional 10% of links. The distribution of additional links was also done at random.

size\density	0.2	0.4	0.6
20	graph 1	graph 2	graph 3
50	graph 4	graph 5	graph 6
100	graph 7	graph 8	graph 9

Table 1. The nine types of graphs used for our experiment

The random graph generator we used labels the nodes numerically according to the order of their creation which, as such, makes task 1 amount to finding the greatest numeric label. Consequently, we decided to make this task more relevant by renaming the nodes alphabetically (from A to T on the twenty-node graphs, from A1 to F0 on the fifty-node graphs, and from A1 to K0 on the one-hundred-node graphs).

3.4.2 The population

The population that performed the evaluation consisted of post-graduate students and confirmed researchers in the fields of computer science. All the subjects knew what a graph was. No further knowledge of graph theory was required. The population consisted of 36 subjects, all of whom had previously seen a node-link representation of graphs. All the subjects participated voluntarily to the evaluation.

3.4.3 The evaluation program

We developed an evaluation program that represents the selected graphs according to both representation techniques. It then asks the user to perform the tasks and records the time to answer.

In terms of interaction, our program provides picking and selection mechanisms. On both visualizations, when the mouse goes over a node, it is highlighted in green as well as its incident links; nodes can also be selected through direct pointing, in which case they are highlighted in red as well as their incident links. Likewise, when the mouse goes over a link, it is highlighted in green as well as its endpoints. (Figure 1) These interactive enhancements were added to help users focus on graph elements after an initial testing showing a high level of frustration from users losing focus.

A demonstration made on a set of two graphs exposed the user to the proper reading of representations and the various tasks to be performed. First, the instructor manipulated the system and provided guidelines. Then the user manipulated the system in order to make sure that the representations, the tasks and the interactions were well understood. At the end of the demonstration, we made sure that the user was ready to start the evaluation per se. The instructor proposed to repeat the demonstration as necessary. At the end of this introductory demonstration, three instructions were given:

1. The user has to answer as quickly as possible.
2. The user has to answer correctly.
3. The user is allowed to move to the next question without answering before the answer time elapses in case he feels he is not able to answer.

To avoid memorization biases, the system selects a representation technique at random – matrix or node-link – and represents sequentially all nine graphs, asking the user to execute the seven tasks for each graph. Then, the system moves to the second technique and proceeds in the same fashion. By interchanging the representation techniques, we make sure that the subjects had the same probability to start the evaluation with a series of visualizations belonging to either technique. Each series was divided into two parts: the first included three simple graphs (graphs 1, 2 and 4) and allowed the user to get familiar with the system; the second included the six remaining graphs.

Furthermore, a learning effect was observed when a user was confronted to the matrix representation. We were able to measure such an undesirable effect in a series of ten preliminary tests where the system selected the graphs from the smallest to the largest and from the sparsest to the most connected. In spite of the increasing complexity of the displayed graphs, users would tend to answer more quickly as their understanding of matrix-based representations increased throughout the experiment. To level this effect, during the evaluation, our system selects the graphs at random within each half-series. In this way, the graphs have an equal probability to appear in the beginning, in the middle, or at the end of their respective half-series.

For tasks involving two nodes, (findLink, findNeighbor and findPath), the system selects both nodes beforehand in order to avoid spending time trying to locate them. Therefore, the time we measure corresponds exactly to the time for executing those tasks. Since each evaluation session contains a total of 126 questions (9 graphs x 2 visualization techniques x 7 tasks), we programmed three pauses: a ten-minute pause between the two series and a five-minute pause between the two halves of each series. Moreover, since the sessions are rather long, (a full hour of manipulation per user), we chose to limit the answer time to 45 seconds per question. When the time runs out, the system moves automatically to the next question, and produces an audio feedback in order to notify the user. In this case, we consider that the representation is not effective for that task since the user was not able to provide the answer in the allotted time. The audio feedback also incites the user to hurry up for next questions.

3.4.4 Implementation and tuning

Node-link diagrams were laid out using AT&T's [1] open source graph layout program neato and the java drawing library grappa. We made our best effort to tune both representations in order to make the best use of them. We made the same interaction techniques available on both visualizations. We paid attention to the size of nodes and the readability of their labels on node-link diagrams, however large or dense they got. We superimposed the labels of picked or selected nodes on a semi-transparent background, which eliminates the occlusion problems due to links overlapping over these nodes. Given that we are dealing with undirected graphs, we did not display any arrows at the endpoints of the links, which significantly improves the node-link representation of dense graphs. Dealing with undirected graphs would also simplify the path lookup task on the matrix-based representation since links appear twice. We stored the parameters of the tuned node-link diagrams in the dot format and used those settings along the evaluation. We thus guarantee that the subjects are confronted with exactly the same representation for respectively all nine graphs. Likewise, we exploited the intrinsic orderability of the matrix representation and sorted its rows and columns alphabetically. The matrix being sorted instantaneously, the matrix-based visualizations did not require any preliminary tuning.

The evaluation was carried out on a Dell workstation having an NVIDIA GeForce2 accelerated video card, a dual Pentium III 1Ghz processor, 512 Mbytes of RAM, under Windows 2000. The display was performed in full screen mode on a 21" monitor. Subjects were seated at sixty centimeters from the monitor and executed all tasks using the mouse.

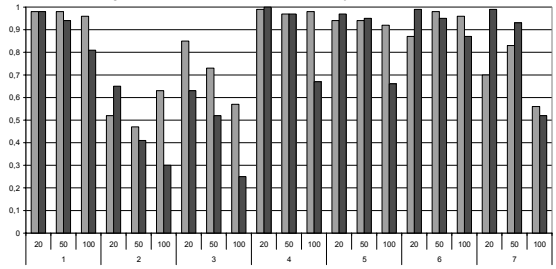


Figure 2 Percentage of correct answers by task and by size. Matrices appear in light gray, node-links in dark gray.

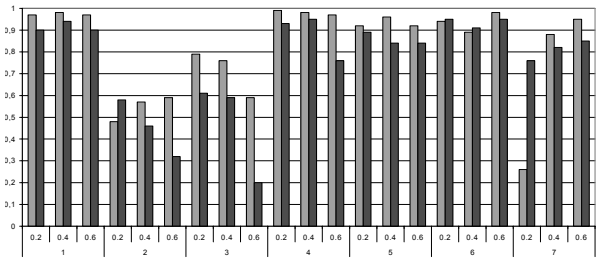


Figure 3 Percentage of correct answers by task and by density. Matrices appear in light gray; node-links in dark gray.

3.5 Results

Throughout this paper, we plotted in light gray the statistics corresponding to matrices and in dark gray those related to node-link diagrams. The measurements were analyzed with a graphic qualitative method (Box-Plot) and a quantitative method (non parametric test of Wilcoxon). The latter makes it possible to compare the central position of two samples without prior knowledge of their distribution (normality for example). This test provides a p-value which is a probability. When the p-value is less than 5%, we conclude that the two samples have the same median value; otherwise, we conclude that the two samples have different central values.

On the box-plot diagrams displayed subsequently, we represent the answer time on the y-axis. For each task, we display the evolution of time for the three graph sizes on diagrams labeled (a), and on the ones labeled (b) we display the evolution of time for the three chosen densities.

In order to fully understand the effect of both variables – size and density – and reveal any possible interaction between them, we ran a multiple regression analysis [10] on the collected measurements. We assume that the response y of a system is supposed to depend on $k = 2$ controllable input variables (ξ_1, ξ_2) called natural variables. In our case, ξ_1 and ξ_2 are respectively the density and the size of the graphs. The true response function $y = f(\xi_1, \xi_2)$ is supposedly unknown. The main goal of the *Multiple Regression Analysis* is to approximate the unknown true response function with a "simple" known function like:

$$y = a_0 + a_1\xi_1 + a_2\xi_2 + a_{12}\xi_1\xi_2 \qquad (1)$$

where a_0, a_1, a_2, a_{12} are the $p = k + 2 = 4$ regression coefficients. This model is called a linear model plus interactions model. In order to estimate the regression coefficients, we have to perform $n > p$ experiments where the natural variables (ξ_1, ξ_2) take different predefined values. In our case, the predefined values are {0.2, 0.4, 0.6} for ξ_1 and {20, 50, 100} for ξ_2. Let **X** and **y** the matrix and vector defined as:

$$\mathbf{X} = \begin{pmatrix} 1 & \xi_{11} & \xi_{12} & \xi_{11}\xi_{12} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \xi_{n1} & \xi_{n2} & \xi_{n1}\xi_{n2} \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \qquad (2)$$

where (ξ_{i1}, ξ_{i2}) are the predefined values chosen during the ith experiment and where y_i is the corresponding response. An estimation of the regression vector $\mathbf{a} = (a_0, a_1, a_2, a_{12})^{\mathrm{T}}$ can be obtained using the following relation:

$$\hat{\mathbf{a}} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y} \qquad (3)$$

The significance of each regression coefficients can be evaluated with the computation of a p-value (not presented here). If the p-value is less than 0.05 then the coefficient is deemed non null, and the corresponding variable (density or size) is deemed influent on the response. On the other side, if the p-value is larger than 0.05 then the coefficient is considered to be null and the corresponding variable is supposed to have no influence on the response.

Finally, we obtain a model represented by a surface in a 3D space whose axes are respectively the two variables (size and density in our case) and the response (answer time in our case). It may also be mentioned that the 3D graphs take into account statistically insignificant terms that were left out for the sake of readability in the corresponding equations. Moreover, in order to help figure out the shape of the 3D model, we have represented in green the projection of the model on plan XY; the green lines correspond to the projection of isovalue contours in the 3D surface. Lastly, we may highlight once again that with regards to tasks involving two nodes such as findLink, findNeighbor and findPath, we made sure that the user did not spend any time finding the nodes in the question by automatically highlighting them. Hence, the measurements faithfully reflected the time spent at these tasks.

In the sequel, we provide a detailed account of the results we obtained. In brief, one may say that, except for findPath, matrix-based visualizations outperform node-link diagrams with medium to large graphs and with dense graphs.

3.5.1 Estimation of the number of nodes (nodeCount)

On Figure 4a, on the matrix-based representation (the light gray boxes), median answer time and time distribution vary a little when size increases, whereas they grow notably on the node-link representation (the dark gray boxes). We therefore conclude that with regard to this task, the readability of node-link diagrams deteriorates significantly when the size of the graph increases whereas the matrix-based representation is less affected by size. On Figure 4b, on the matrix-based representation, median answer time and time distribution increase a little when the density increases; they increase slightly on the node-link representation.

Moreover, in Figure 2, we note that, with regard to large graphs, 96% of the users have answered correctly using the matrix-based representation, against 81% using the node-link representation, that is to say a difference of 15%, deemed statistically significant according to Wilcoxon's test. On the matrix-based representation, the percentage of correct answers remains stable, around 97%, when density varies (Figure 3), which corresponds to a statistically significant improvement of 7% compared to the node-link representation for low and high densities.

In Figure 5, we display the regression models of answer time with regard to the "nodeCount" task. On Figure 4a, answer time using a matrix (T_{MX}) is modeled by the following regression equation:

$$T_{MX} = 18.8999 - 15.3938 \times d + 0.157116 \times d \times s \qquad (4)$$

where d stands for density and s for size. On Figure 4b, answer time using a node-link diagram (T_{NL}) for the same task is modeled by the following regression equation:

$$T_{NL} = 13.7151 - 10.6864 \times d + 0.302776 \times d \times s \qquad (5)$$

Equation 4 shows some interaction between size and density of graphs (see the last coefficient in the equation) and a favorable effect of density on matrices (the coefficient related to density is negative). Likewise, equation 5 shows strong interaction between size and density with node-links (see the highest value at (0.6, 100)) and a favorable effect of density on answer time. Size did not seem to play a statistically significant role in the estimation of node count in either case. This can be explained by the fact that users really estimated the size of graphs without resorting to an exact count; the complexity of many node-link diagrams would deter them if they tried! This is also a likely explanation of the favorable effect of density.

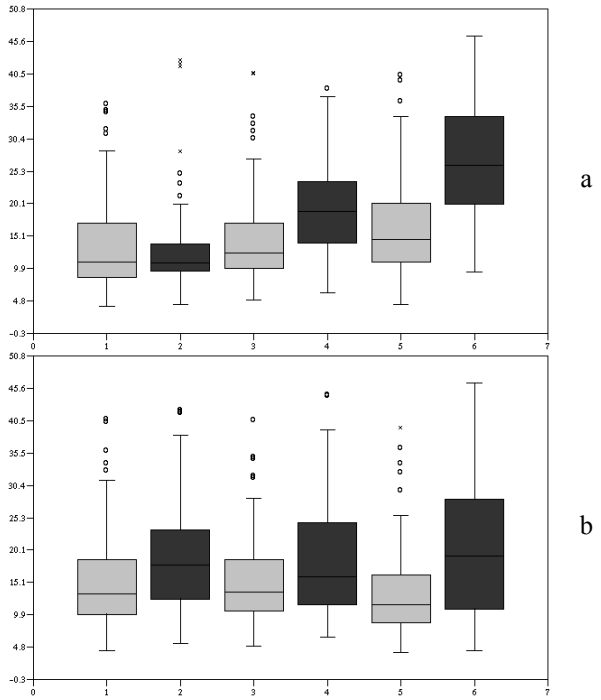


Figure 4 Distribution of answer time for "nodeCount" (a) split by size, (b) split by density

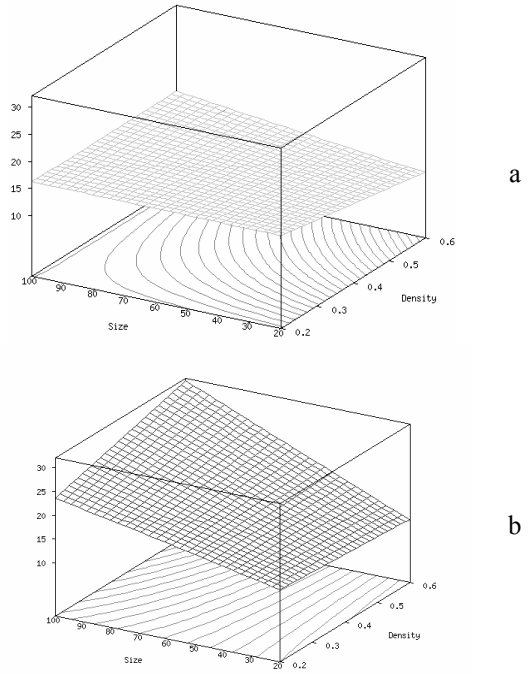
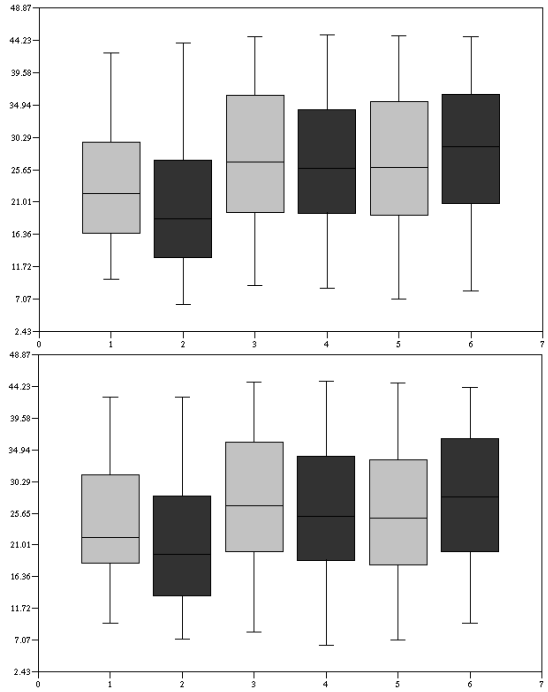


Figure 5: Model of response time with regard to nodeCount for (a) matrices and (b) node-links.

Figure 5 and the related equations suggest that the readability of the node-link diagrams is strongly affected by an interaction between density and size, while size has no significant independent effect. The readability of matrices is less influenced by the interaction between size and density – although such interaction is observed – and is not sensitive to size variation. On top of that, matrices allow carrying out this task more quickly when size and density are medium or large. Indeed, the dark gray surface takes off for these values while the light gray surface remains unchanged.

3.5.2 Estimation of the number of links (linkCount)

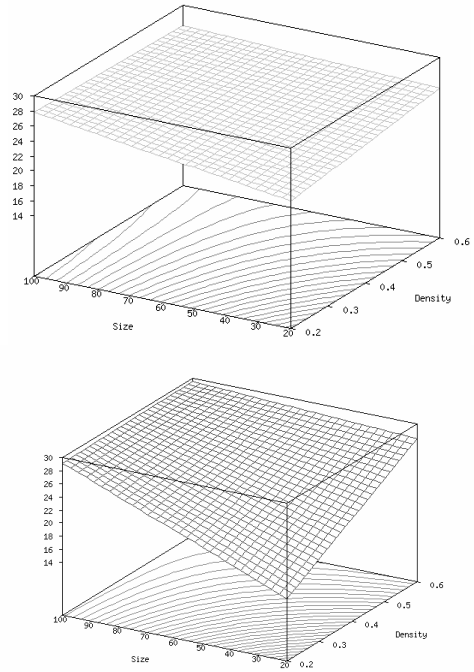
Based on Figure 6, the estimation of the number of links in the graph seems relatively independent of size or link density when these variables take medium or large values. On Figure 6a (x-coordinates 2 and 4), there is a gap in answer time between small and medium-sized graphs and, on Figure 6b (x-coordinates 2 and 4), between sparse and moderately dense graphs. However, there seems to be no difference between the two techniques for any given size or density. On Figure 2, the matrix-based representation records 57% of correct answers on large graphs. This figure goes as low as 25% using the node-link representation, that is to say a significant discrepancy of 27 % compared to matrices. The difference recorded with regard to small and medium-sized graphs respectively in favor of the node-link representation and the matrix-based representation is statistically insignificant. Similar conclusions can be drawn for link density (Figure 3).



a

b

Figure 6 Distribution of answer time for "linkCount" (a) split by size, (b) split by density



a

b

Figure 7: Model of response time with regard to linkCount for (a) matrices and (b) node-links.

The regression equations related to Figure 7 are respectively as follows:

$$T_{MX} = 19.6512 + 0.0844522 \times s \quad (6)$$

$$T_{NL} = 38.3796 \times d + 0.246053 \times s - 0.374642 \times d \times s \quad (7)$$

It appears that answer time using matrices is completely independent of density, whereas answer time using node-link diagrams depends on both size and density and is favorably impacted by an interaction effect between both variables. While node-links stand out clearly when it comes to small graphs (Figure 6 and Figure 7), they take off quickly and provide answer times quite similar to those obtained with matrices. Only the ratio of correct answers makes a difference between the two representations in favor of matrices.

3.5.3 Finding the most connected node (mostConnected)

When achieving this task, we note that, with regard to answer time, both techniques are sensitive when size increases (Figure 8a), whereas they are slightly affected when link density increases (Figure 8b). We cannot differentiate these methods with regard to this task based on answer time only.

Nevertheless, according to Figure 2, 85% of the users execute this task correctly on small graphs using a matrix-based representation against 63% of correct answers with the node-link representation. On medium-sized graphs, we have 73% of correct answers using a matrix against 52% using node-links. Lastly, on large graphs, we record 57% of correct answers using a matrix against only 25% of correct answers with node-link diagrams. These differences are deemed statistically significant using Wilcoxon's test. Similar conclusions can be reached with regard to density on Figure 3.

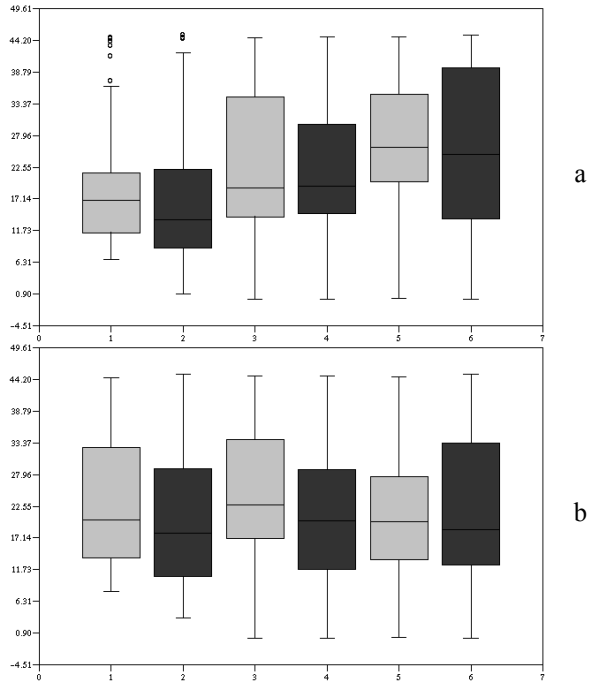


Figure 8 Distribution of answer time for "mostConnected" (a) split by size, (b) split by density

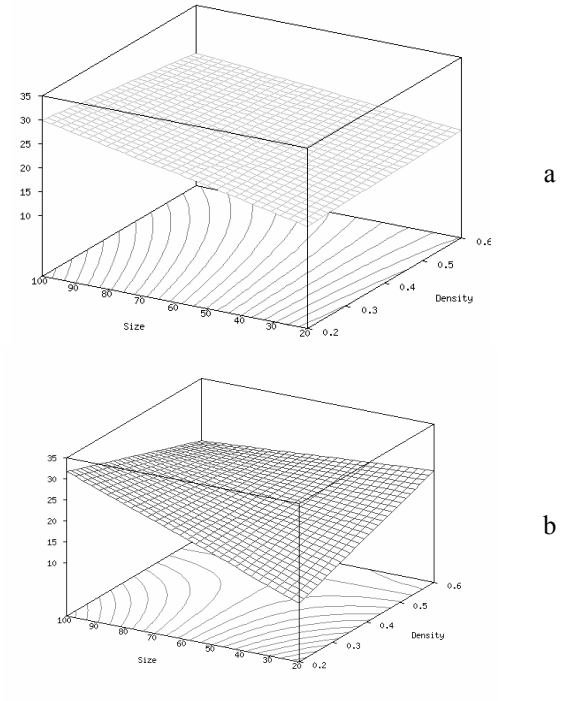


Figure 9: Model of response time with regard to mostConnected for (a) matrices and (b) node-links.

The regression equations related to Figure 9 are:

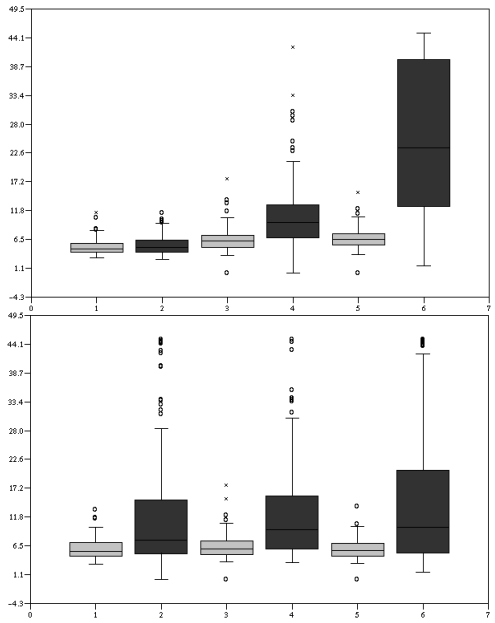
$$T_{NL} = 47.0504 \times d + 0.406933 \times s - 0.757307 \times d \times s \quad (8)$$

$$T_{MX} = 14.3773 + 0.179544 \times s \quad (9)$$

Matrices prove to be completely independent of density; the answer time required for finding the most connected vertex depends on size only with a slope of 18%. As for node-links, answer time depends very much on both variables, despite a strong interaction between both variables reduces the measured values of answer time. (See how the surface bends at the middle on Figure 9b.) Bearing the large ratio of incorrect answers in mind (Figure 2), it seems that some users would randomly select any answer to this question when the graphs became too complex, a behavior that we had the opportunity to see while watching the users.

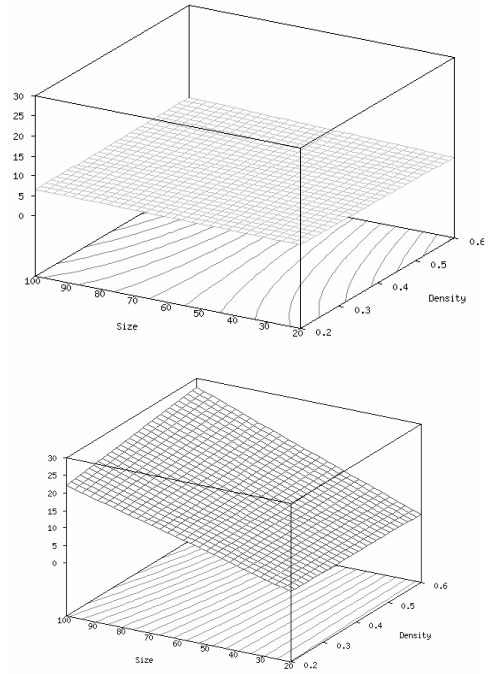
3.5.4 Finding a specified node (findNode)

When considering answer time, we can see that the readability of node-link diagrams deteriorates quickly when the size of the graph increases (Figure 10a) and are moderately affected by link density (Figure 10b). In contrast, the answer time on the matrix-based representation deviates a little when the size increases and does not seem to be affected at all by link density. The dispersion of answer time is very small using matrix-based representation in both cases. When dealing with small graphs, both representations perform equally well. The percentage of correct answers (Figure 2) is high, almost 98%, using the matrix-based representation, irrespective of the size of the graph. The percentage of correct answers is equally good using node-link diagrams, except for large graphs whose score falls as low as 67%, with a discrepancy of 31% compared to matrix-based representations. Similar conclusions can be drawn when link density varies (Figure 3).



a

b



a

b

Figure 10 Distribution of answer time for "findNode" (a) split by size, (b) split by density

Figure 11: Model of response time with regard to findNode for (a) matrices and (b) node-links.

The regression equations related to Figure 11 are as follows:

$$T_{MX} = 6.02597 \qquad (10)$$

$$T_{NL} = 0.179424 \times s + 0.179913 \times d \times s \qquad (11)$$

Equation 10 confirms what we already know, more precisely the fact that looking up a vertex in an ordered representation (the matrix) does not depend on size or density of graphs. Node-link diagrams seem independent of density; this representation suffers however from the size of graphs (i.e. the number of nodes therein) and from an interaction between size and density (see the dark gray surface takes off until it reaches its maximum value at the far top corner). This may be accounted for by the complexity induced by the unpredictable layout of nodes in node-link diagrams, whatever the density may be. Size proves to be more preponderant in this case since it increases substantially the list of nodes to be scanned through.

3.5.5 Finding a link between two nodes (findLink)

Using the node-link representation, the larger the graph the longer it takes to look up a link in it (Figure 12a); the answer time does not vary significantly when link density increases (Figure 12b). Using matrices, this task is insensitive to size and density variation. For large graphs, and for medium and high link density, a significant gap is measured in favor of matrix-based representations. A significant difference is measured in favor of node-link diagrams for small graphs. Both representations record excellent percentages of correct answers, about 95%, for small and medium-sized graphs (Figure 2). For large graphs, the matrix-based representation records 92% of correct answers against 66% with the node-link diagrams, that is a discrepancy of 26%.

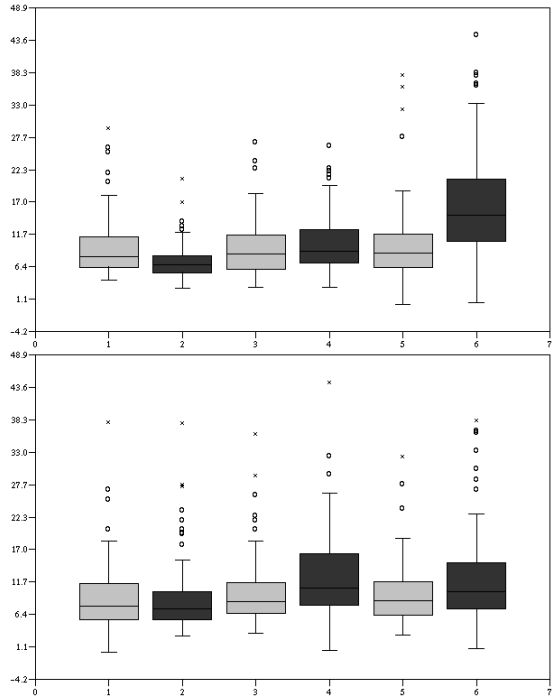


Figure 12 Distribution of answer time for "findLink" (a) split by size, (b) split by density

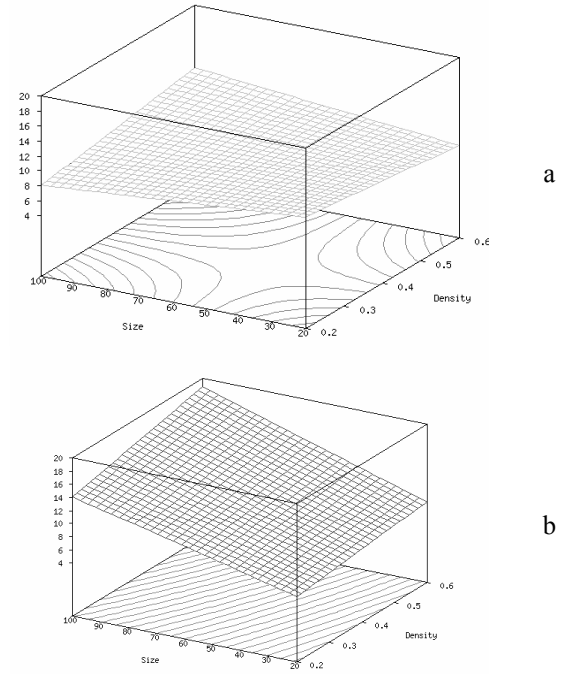


Figure 13: Model of response time with regard to findLink for (a) matrices and (b) node-links.

The regression equations related to Figure 13 are:

$$T_{MX} = 13.3781 - 9.7559 \times d - 0.0714139 \times s + 0.188828 \times d \times s \qquad (12)$$

$$T_{NL} = 0.088413 \times s \qquad (13)$$

Equation 13 confirms what we had already observed (Figure 12) i.e. answer time using node-link diagrams depends significantly on size only. We could also observe how the answer time using node-links takes off for large values of size and density. As for matrices, an interaction between size and density influences answer time. We also note an improvement in answer time when size or density increase. This may be accounted for by a learning effect relevant for the graphs appearing in phase 1 of the evaluation.

3.5.6 Finding of a common neighbor (findNeighbor)

Using the node-link representation, the larger the graph the longer it takes to say whether a common neighbor exists (Figure 14a); the median answer time is marginally affected when link density increases (Figure 14b). On the matrix-based representation, size variation has no impact on answer time, while median answer time and value dispersion improve slightly when link density is large.

When dealing with small graphs, node-link diagrams record 99% of correct answers with a lead of 12% over matrices. Matrix-based representations take an equivalent lead when dealing with large graphs, towering at 96% of correct answers (i.e. node-links recorded 84% of correct answers). Both techniques record a similar percentage of correct answers on medium-sized graphs (Figure 2). When link density varies (Figure 3), both representations score about 90% of correct answers.

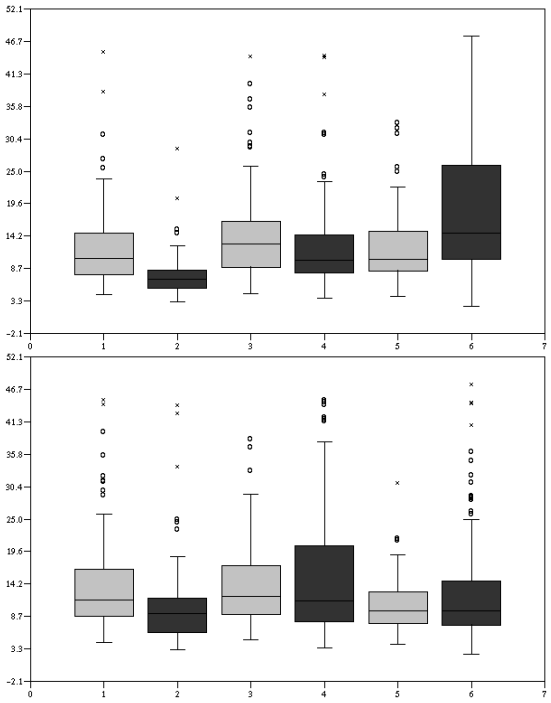


Figure 14 Distribution of answer time for "findNeighbor" (a) split by size, (b) split by density

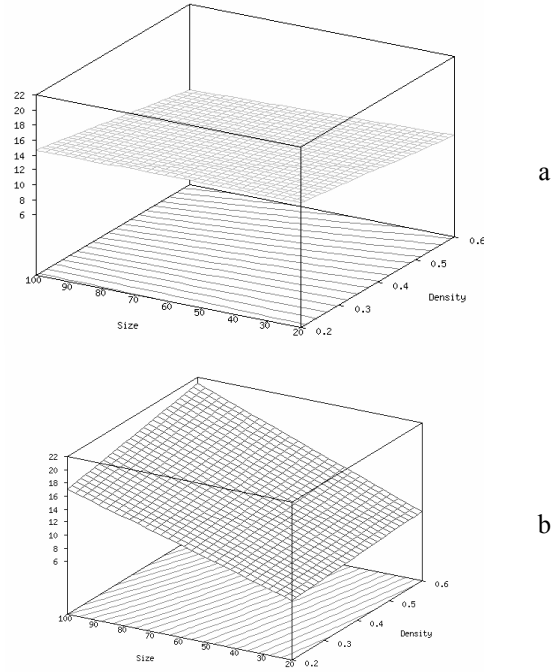


Figure 15: Model of response time with regard to find-Neighbor for (a) matrices and (b) node-links.

The regression equations plotted in Figure 15 are:

$$T_{MX} = 15.777 \quad (14)$$

$$T_{NL} = 0.109657 \times s \quad (15)$$

Equation 14 suggests that answer time using matrices is independent of the variables of the problem. At first, this may come as a surprise, but a sensible explanation can be found along the following observations. During the evaluation, the vertices in the questions are automatically highlighted, i.e. the related columns in the matrix are highlighted, finding a common neighbor amounts to finding a row intersecting both columns in a pair of black cells, one on each column. Therefore, most users would carry out this task by rolling the mouse over the rows sequentially until they hit one such row or hit the end of the matrix (We may mention once again that when the mouse flies over a row it is highlighted). Very often users would skip the first few rows and then backtrack over them if they fail to find a common neighbor at the first trial. Not only may this schema explain why answer time appears to be independent of size and density, but also why answer time is rather long (around 15 seconds) compared to other tasks like findNode where answers fall twice as quickly (around 6 seconds). Using node-link diagrams, answer time depends significantly on graph size, which confirms our previous observations (Figure 14). Node-link diagrams appear to perform better on small sparse graphs, but their performance worsens dramatically with large and dense graphs (see the peak reached on the 3D plot at the high end of size and density values).

3.5.7 Finding a path between two nodes (findPath)

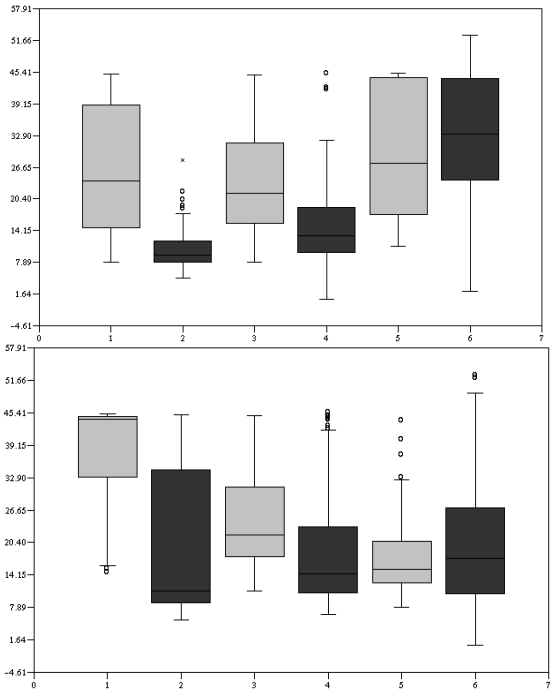


Figure 16 Distribution of answer time for "findPath" (a) split by size, (b) split by density

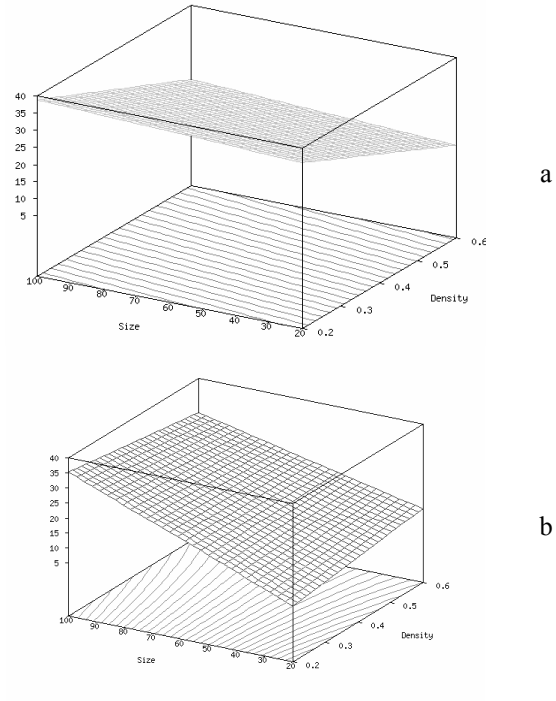


Figure 17: Model of response time with regard to findPath for (a) matrices and (b) node-links.

Finding a path between two nodes proves to be increasingly difficult using node-link diagrams when the size increases (Figure 16a), whereas the median answer time increases slightly when link density increases (Figure 16b). The matrix-based representation performs very poorly for this task except for very dense graphs where it outperforms the node-link representation (Figure 16b). This fact is confirmed by the percentage of correct answers which towers at 95% for the matrix-based representation against 85% for node-links when dealing with large link density (Figure 3). On small graphs, 99% of the users answer correctly using node-link diagrams against 70% only using matrices. On medium-sized graphs, node-link diagrams record 93% of correct answers with a lead of 10% over matrices. For large graphs, every other user answers correctly using both representations with a statistically insignificant lead in favor of matrices. Lastly, this task is clearly in favor of node-link diagrams when visualizing sparse graphs.

The regression equations plotted in Figure 17 are:

$$T_{MX} = 45.7229 - 53.507 \times d \quad (16)$$

$$T_{NL} = - 6.05372 + 23.2979 \times d + 0.445508 \times s - 0.397442 \times d \times s \quad (17)$$

The performance of matrices improves significantly when it comes to finding a path as graphs become increasingly dense. This does not come as a surprise since more and more short paths are being available between any pair of nodes. This task would then be quite similar to finding a common neighbor or even finding a direct link between the given endpoints. The answer time expected with node-link diagrams depends on both size and density, while an interaction between size and density may prove helpful. As seen earlier, node-links perform far better than matrices on this task with regard to small sparse graphs. However, matrices outperform them about large and dense graphs.

4 DISCUSSION

We expected the readability of node-link diagrams to deteriorate when the size of the graph and its link density increase. This hypothesis was confirmed for the seven tasks we selected. Only for "findPath" task did node-link diagrams prove superior to matrix-based representations, although their performance deteriorates on large and dense graphs. This conclusion must however be qualified since this task is difficult to carry out visually when the distance between the endpoints is greater than two or three arcs, as shown in [13].

Another hypothesis was related to the significant impact of orderability of matrices on node and link finding tasks. "findNode" and "findLink" tasks validate this hypothesis for large graphs and for dense graphs.

As far as "linkCount" task is concerned, both visualizations record a large share of erroneous answers. We account for – but this has yet to be proven through experimentation – that the number of links is intrinsically difficult to estimate on node-link diagrams and that users failed to compute it correctly using matrices. Indeed, links are displayed twice because we considered undirected graphs in our study.

We may also question the extensibility of the results obtained in this evaluation to other node-link layout programs than the one we chose in our experimentation. However, based on earlier works [11], we can safely assert that, with regard to small

graphs, the layout program has very little impact on the readability of the displayed output and would not change the trends we observed.

We may further highlight that all the users who took part in the experiment were familiar with node-link diagrams whereas none had previously heard about the matrix-based visualization of graphs. Since they were given little training through a short demonstration (first, users would watch the instructor perform the tasks on a graph, and then they would train on one similar graph), we expect users familiar with both representations to perform even better with matrices.

As a first approach we have split the data by size and by density in order to measure the effect of these variables on the readability of graph representations. To this end, we have done our best effort to isolate those factors and make sure that no other considerations would interfere with the tasks. Moreover, a multiple regression analysis has been used to investigate the interaction between size and density of graphs and the impact of such interaction on the readability of their representations. In most cases, no interaction was found for the matrix-based representation, while some interaction was found between size and density with regard to node-link diagrams. A negative interaction was observed indeed on node-link diagrams with regard to the linkCount, mostConected, and findPath tasks, which means that answer time would improve while the complexity of graphs increase. This can be explained by the fact that node-link diagrams become so cluttered that users would be tempted to give any answer. This is confirmed by the high rate of erroneous answers observed for linkCount and mostConnected. As far as findPath is concerned, many users answered this task correctly. Therefore, there must be another explanation to account for that. For instance, finding a path on a node-link diagram representing a sparse graph proves all the more difficult if the shortest path between the endpoints is long (which is even more difficult with matrices). In this case, the density of the graph may not be the most relevant indicator; the length of the shortest path may be a better choice and should be taken into account. Conversely, in dense graphs, the shortest path is likely to be one or two links long, but the visual clutter produced by links on node-link diagrams renders this task infeasible, while matrices perform very well. Only one task, findLink, related answer time using matrices to both variables of the problem as well as an interaction between them. Otherwise, answer time using matrices would depend on either variable alone, while it occurred twice – for tasks findNode and findNeighbor – that answer time was deemed independent of size and density of graphs. This was expected with regard to findNode and came as a surprise with regard to findNeighbor. For the latter, the explanation may be related to the search schema applied by most users (a cursory sequential scan over the rows of the matrix).

In this evaluation, we compare two representations of graphs, a matrix-based representation and a node-link representation produced by a force directed algorithm, against nine random graphs and a set of seven exploration tasks. In this context, various recommendations and insights are derived from our study. For small graphs, node-link diagrams are always more readable and more familiar than matrices. For larger graphs, the performance of node-link diagrams deteriorates quickly while matrices remain readable with a lead of 30% of correct answers, with comparable if not better answer time. For more complex tasks such as "findPath", we are convinced that an appropriate interaction is always preferable, for example by selecting a node and displaying all the possible paths starting from it and ending at a pointed node. On the matrix-based representation, this path can be displayed using curves connecting adjacent links, i.e. connecting the cells representing those links.

5 CONCLUSION

In this paper, we have listed generic tasks for the visualization of graphs and have compared two representations of graphs on a subset of these tasks. These techniques proved to be complementary: node-link diagrams are well suited for small graphs, and matrices are suitable for large or dense graphs. Path related tasks remain difficult on both representations and require an appropriate interaction that helps perform them. We are investigating with interaction techniques to overcome this pitfall.

The matrix-based representation seems therefore under exploited nowadays, despite its quick layout and its superior readability with regard to many tasks. We think that a wider use of this representation will result in a greater familiarity and will consequently improve its readability. We currently use the matrix-based representation for the real-time monitoring of constraint-oriented programs where graphs evolve dynamically, both in size and activity. The results we are obtaining are quite encouraging.

We are investigating clustering and aggregation techniques on matrices for the visualization of very large graphs having about tens of thousands vertices.

6 ACKNOWLEDGEMENTS

This work has partially been funded by the French RNTL OADYMPPAC project. We would like to thank Pierre Dragicevic, Véronique Libérati and Vanessa Tico for their time and advice. We are grateful to our colleagues at Ecole des Mines de Nantes who volunteered and took part in this evaluation.

REFERENCES

[1] AT&T Labs Research. Graphviz - open source graph drawing software, 2004. http://www.research.att.com/sw/tools/graphviz/

[2] Battista, G.D., Eades, P., Tamassia, R. and Tollis, I.G. Graph Drawing. Prentice Hall, 1999.

[3] Becker, R.A., Eick, S.G. and Wills, G.J. Visualizing network data. IEEE Transaction on Visualizations and Graphics, 1 (1). 16-28.

[4] Bertin, J. Sémiologie graphique : Les diagrammes - Les réseaux - Les cartes. Editions de l'Ecole des Hautes Etudes en Sciences, Paris, France, 1967.

[5] Cohen, R.F., Eades, P., Lin, T. and Ruskey, F., Volume upper bounds for 3D graph drawing. in Proceedings of the 1994 conference of the Centre for Advanced Studies on Collaborative research, (Toronto, Ontario, Canada, 1994), IBM Press.

[6] Ghoniem, M., Jussien, N. and Fekete, J.-D., VISEXP: visualizing constraint solver dynamics using explanations. in FLAIRS'04: Seventeenth international Florida Artificial Intelligence Research Society conference, (Miami Beach, FL, 2004), AAAI press.

[7] Ham, F.v., Using Multilevel Call Matrices in Large Software Projects. in Proc. IEEE Symp. Information Visualization 2003, (Seattle, WA, USA, 2003), IEEE Press, 227-232.
[8] Herman, I., Melançon, G. and Marshall, M.S. Graph Visualization and Navigation in Information Visualization: a Survey. IEEE Transactions on Visualization and Computer Graphics, 6 (1). 24-43.
[9] ISPT, Waseda University. Random Graph Server. http://www.ispt.waseda.ac.jp/rgs/index.html
[10] Montgomery, D.C. and Runger, G.C. Applied Statistics and Probability for Engineers, Wiley, 1999.
[11] Purchase, H.C., The Effects of Graph Layout. in Proceedings of the Australasian Conference on Computer Human Interaction, 1998, p. 80.
[12] Purchase, H.C., Carrington, D.A. and Allder, J.-A. Empirical Evaluation of Aesthetics-based Graph Layout. Empirical Software Engineering, 7 (3). 233-255.
[13] Purchase, H.C., Cohen, R.F. and James, M.I. An Experimental Study of the Basis for Graph Drawing Algorithms. The ACM Journal of Experimental Algorithmic, Volume 2, Article 4, 1997.
[14] Ware, C., Purchase, H.C., Colpoys, L. and McGill, M. Cognitive measurements of graph aesthetics. Information Visualization, 1 (2). 103-110.

Les leçons tirées des deux compétitions de visualisation d'information

Jean-Daniel Fekete

INRIA Futurs/LRI
Bât 490, Université Paris-Sud

F91405 Orsay Cedex, France
Jean-Daniel.Fekete@inria.fr

Catherine Plaisant

Human-Computer Interaction Laboratory
University of Maryland
A.V. Williams Building, College Park,
MD 20742, USA
plaisant@cs.umd.edu

RESUME

La visualisation d'information a besoin de benchmarks pour progresser. Un benchmark est destiné à comparer des techniques de visualisation d'information ou des systèmes entre eux. Un benchmark consiste en un ensemble de données, une liste de tâches, généralement des recherches à accomplir et une liste de résultats de recherches connus sur ces données (les *pépites* à trouver). Depuis deux ans, nous organisons une compétition de visualisation d'information destinée avant tout à obtenir des résultats pour des benchmarks. Nous décrivons ici les leçons les plus importantes que nous en avons tirées.

MOTS CLES : Visualisation d'information, benchmark, compétition, évaluation.

ABSTRACT

Information visualization needs benchmarks to carry on. A benchmark is aimed at comparing information visualization techniques or systems. A benchmark is made of a dataset, a list of tasks mostly based on finding facts about the dataset, and a list of interesting or important findings about the datasets (the *nuggets* to find). For the second year, we are organizing the InfoVis Contest aimed at collecting results for benchmarks. We describe here the main lessons we learned.

CATEGORIES AND SUBJECT DESCRIPTORS: H.5.2 [Information interfaces and presentation] User Interfaces - Evaluation/methodology; Graphical user interfaces (GUI)

GENERAL TERMS: Visualization, Measurement, Performance, Human Factors.

KEYWORDS: Information Visualization, Benchmark, Contest, Evaluation.

INTRODUCTION

Le domaine de la visualisation d'information mûrit et les outils conçus par notre communauté commencent à toucher des utilisateurs. La visualisation d'information sort des laboratoires de recherche et passe dans des produits commerciaux de plus en plus nombreux (Spotfire, HumanIT, ILOG, InXight), en addition de produits statistiques (SPSS/SigmaPlot, SAS/GRAPH et Datadesk) ainsi que dans des environnements de développements comme ILOG JViews. Le public est aussi exposé directement à des visualisations utiles comme la carte du marché (MarketMap de SmartMoney), les statistiques du bureau de recensement américain en ligne ou la visualisation de l'état de la circulation automobile.

Que pouvons-nous faire pour améliorer l'utilité de la visualisation et donner des informations utiles aux professionnels désireux de mettre en œuvre des techniques de visualisation dans leurs systèmes ? Améliorer les techniques d'évaluation et les théories prédictives sont deux chemins évidents mais difficiles et qui prendront plusieurs années avant de donner des résultats généralisables [1,2,3].

Dans des délais plus court, une voie intéressante est de construire des benchmarks permettant de comparer des techniques de visualisation et des systèmes pour des jeux de données relativement génériques et pour des tâches pouvant autant que possible être transférées dans plusieurs domaines d'applications.

La constitution de benchmarks est une méthode qui tend à se diffuser en informatique. Plusieurs autres domaines ont développés des benchmarks qui deviennent incontournables : TREC[1] pour la fouille de texte, KDD Cup[2] pour la fouille de données, PETS[3] pour la surveillance vidéo etc.

Traditionnellement, les benchmarks sont composés de deux parties : un jeu de données et des tâches à accomplir sur ces données. Par exemple, si les données sont des pairs d'arbres, une tâche simple peut être de trouver quel arbre a le plus de nœuds.

Bien entendu, pour évaluer la qualité des réponses, il convient de connaître les réponses aux questions posées. Une difficulté de la visualisation d'information vient du fait qu'elle prétend favoriser l'exploration et les découvertes et qu'il est difficile de définir des métriques précises sur les découvertes. Soit nous spécifions de tâches simples et nous maîtrisons les résultats, soit nous demandons des tâches exploratoires et nous contrôlons

Copyright © 2004 by the Association for Computing Machinery, Inc. permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Publications Dept., ACM, Inc., fax +1 (212) 869-0481, or permissions@acm.org.
IHM 2004 Aug 30 – Sep 3, 2004, Namur, Belgium
Copyright 2004 ACM 1-58113-926-8 ……. $5.00

[1] http://trec.nist.gov/
[2] http://www.kdnuggets.com/datasets/kddcup.html
[3] http://vspets.visualsurveillance.org/

beaucoup moins les résultats. Malgré tout, on peut penser qu'un système qui ne permet pas d'accomplir des tâches simples a peu de chance de permettre d'accomplir les tâches plus complexes ou exploratoires. La première année, nous avons panaché des tâches simples et des tâches de haut niveau. La seconde année, les tâches étaient moins guidées. Les tâches simples peuvent être évaluées avec les techniques classiques telles les expériences contrôlées mais les tâches exploratoires sont plus complexes à évaluer.

2003 : COMPARAISON DE PAIRES D'ARBRES

Déjà en 1997, la conférence CHI a organisé une compétition de visualisation sur les arbres : le CHI Browseoff [13]. Il s'agissait d'une session ludique où plusieurs chercheurs devaient manipuler leurs systèmes devant une audience pour réaliser des tâches en temps réel. Le jeu de données était une taxonomie générale des objets du monde et les tâches étaient essentiellement des recherches plus ou moins guidées.

Les arbres sont des structures de données extrêmement fréquentes dans des domaines d'application très différents. La visualisation d'information dispose de techniques et de systèmes spécifiques pour visualiser des arbres comme les Treemaps ou les diagrammes nœud-lien.

Lors de cette première compétition, nous avons voulu collecter une vue d'ensemble sur les techniques de visualisation et de navigation dans les arbres. Nous avons tenté de croiser plusieurs caractéristiques d'arbres : la taille, l'arité et le nombre d'attributs associés aux branches et aux feuilles. En fixant comme objectif la comparaison d'arbres, la compétition incluait donc la visualisation, la navigation et la comparaison, ce qui représentait un croisement riche.

Jeux de données

Trois jeux de données étaient proposés :

1. une large taxonomie des êtres vivants (environ 200 000 nœuds d'arité variable avec des attributs multiples) en deux versions dont il fallait trouver et caractériser les différences ;
2. les méta-données d'un site Web, capturé toutes les semaines cinq fois consécutivement (environ 70 000 nœuds par semaine, d'arité variable et avec moins de dix attributs par nœud) ;
3. deux arbres phylogénétiques de petite taille (60 nœuds, deux attributs, arité fixe) de deux protéines censées avoir co-évolué données par Elie Dassa, chercheur à l'institut Pasteur.

Dans le premier jeu de données, nous savions la nature des différences entre les deux versions car elles avaient été introduites par Cynthia Parr, une biologiste de l'université du Maryland. Pour l'évolution du site Web, nous n'avions aucune idée de ce qui allait changer. Pour les arbres phylogénétiques, il s'agissait de comparaison approximative car les arbres eux-mêmes étaient issus d'un processus de construction statistique dont le résultat pouvait énormément changer suivant les algorithmes de construction et les réglages des seuils de ces algorithmes. Ces difficultés, ainsi que la nécessité de travailler avec des biologistes pour maîtriser la nature du problème, ont limité l'intérêt des participants pour ce jeu de données par rapport aux deux autres.

Tâches

Nous avons décrit deux type de tâches : génériques et spécifiques. La généricité des tâches est un problème difficile en visualisation d'information : on voudrait de la généricité autant que possible mais dans les systèmes réels, les utilisateurs finissent toujours par s'intéresser à des tâches concrètes et dépendantes du domaine. Un des problèmes importants de la constitution des benchmarks consiste à trouver le bon niveau de généricité et de spécificité pour permettre une certaine généralisation tout en permettant de résoudre des problèmes concrets.

Déroulement de la compétition

Organiser une compétition pose des problèmes d'organisation un peu différents et plus compliqués que l'organisation d'un colloque scientifique. Nous avons établi le calendrier suivant :

- Octobre : annonce de la compétition et de son sujet (comparaison de paires d'arbres) sans dévoiler les données
- Février : publication des jeux de données et des tâches et démarrage de la période de questions/réponses pour de clarifications sur le tâches ou le format des données ainsi que la correction d'erreurs dans les données.
- Août : soumissions
- Septembre : résultats
- Octobre : réception des résultats mis sous une forme standardisée afin d'en faciliter la lecture et comparaison pour leur publication sur le site de benchmarks d'InfoVis [12].

Nous avons choisi ces dates pour coïncider avec le second trimestre de cours aux USA afin que des groupes d'étudiants puissent participer. Deux enseignants ont joué le jeu et plusieurs groupes ont échangé des messages pendant la période de questions/réponses et certains ont finalement soumis.

Résultats

Nous avons reçu huit soumissions à cette première compétition. C'est un nombre faible mais peu surprenant pour une première année, et qui nous arrangeait, n'ayant aucune idée de la manière dont nous pouvions évaluer ces soumissions.

Chaque soumission devait contenir un document PDF de deux pages, une vidéo montrant le système en action – s'agissant de visualisation interactive – et un document HTML joint expliquant comment les tâches étaient accomplies à l'aide du système.

Nous avons dépouillé tous les résultats à quatre : les deux responsables de la compétition, Cynthia Parr, et Anita Komlodi de l'UMBC comme relectrice externe. La consigne étant de juger comment l'outil permettait aux utilisateurs d'accomplir les tâches demandées. Les soumissions devaient être triées en trois catégories : premier prix (plusieurs prix possibles), second prix et rejetés. Les premiers prix obtenaient une récompense symbolique de

Figure 1: Images d'écran des trois premiers prix en 2003 : TreeJuxtaposer, Zoomology et InfoZoom

15 minutes de présentation lors du symposium. Les seconds prix pouvaient présenter leurs outils pendant les sessions de posters. Les soumissions non rejetées sont toutes disponibles sur le site des benchmarks [12].

Les premiers prix ont été les suivants (Figure 1) :

- James Slack et al. avec TreeJuxtaposer (premier toute catégorie) de l'université de British Columbia, Canada [4]. Ils ont soumis la description la plus convaincante de la manière dont les tâches étaient accomplies et les résultats interprétés.
- Jin Young Hong et al. avec Zoomologie (premier projet étudiant) de Georgia Tech., USA [5]. Ils ont présenté un système original et ont montré qu'il était efficace pour plusieurs jeux de données.
- Michael Spenke et al. avec InfoZoom (premier en originalité) de Fraunhofer Institute, Allemagne [6]. InfoZoom a été une surprise car, à l'origine, ce système a été conçu pour manipuler des tables et non des arbres. Cependant, les auteurs ont impressionnés les relecteurs en montrant que leur système pouvait accomplir toutes les tâches, faire des découvertes et même trouver des erreurs qui n'avaient jamais été trouvées dans des jeux de données censés être validés depuis longtemps.

Les trois deuxièmes prix ont montrés des techniques prometteuses mais n'ont pas su communiquer aux relecteurs les découvertes qu'ils avaient fait ou la façon dont ils les avaient fait.

- David Auber et al. avec EVAT, du LaBRi à Bordeaux [7]. EVAT a montré des outils analytiques qui amélioraient sensiblement la visualisation pour accomplir les tâches de comparaison demandées.
- Nihar Sheth et al. de l'université d'Indiana [8]. Cette soumission a montré les bénéfices offerts par une infrastructure logicielle pour rapidement construire et assembler des outils et faire des analyses.
- David R. Morse et al. avec Taxonote, de l'Open University, Angleterre et université de Tsukuba, Japon [9]. Taxonote a été réalisé par des professionnels des taxonomies et a montré que la gestion des labels était très importante.

Les soumissions reçues se répartissaient selon la figure 2, ce qui nous a donné un critère simple de classement.

Il s'est avéré que lors de la présentation au symposium, InfoZoom a le mieux démontré sa grande flexibilité pour la manipulation des données de la compétition, ce qui lui aurait valu un « prix du public » tacite.

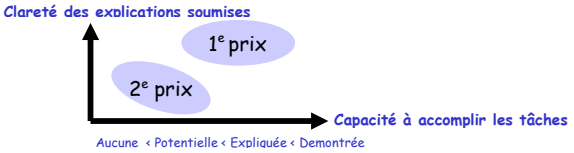


Figure 2 : distribution des soumissions selon la clarté des explications soumises et les capacités des outils.

Les participants n'ont accomplis qu'une partie des tâches, et des tâches différentes, rendant les comparaisons quasi impossibles. Les auteurs, qui sont accoutumés à décrire leurs systèmes ont continué à le faire sans toutefois fournir les réponses aux questions. Néanmoins les résultats étaient intéressants et ont démontrés la diversité des approches possibles et mis à jour les avantages des techniques présentées. La compétition a encouragé le développement de techniques nouvelles, et la présentation des résultats de cette première compétition fut très appréciée lors de la conférence, ce qui a encouragé les organisateurs du symposium à pérenniser la formule.

2004 : VISUALISER L'HISTOIRE DE LA VISUALISATION D'INFORMATION

La seconde compétition, dont les résultats ont été annoncés un mois avant IHM et seront présentés en octobre, coïncide avec les 10 ans du Symposium InfoVis. Le sujet était donc naturellement de visualiser l'histoire de la visualisation d'information. Georges Grinstein, de l'université du Massachusetts à Lowell, s'est joint aux deux précédents organisateurs, pour qu'ils puissent être relevés l'année suivante.

Visualiser l'histoire d'un domaine de recherche est un problème intéressant en tant que tel et fait partie (ou devrait faire partie) du travail des organismes de supervision et planification de la recherche. Un avantage du thème est qu'il est familier aux participants. Un inconvénient est qu'il n'existe aucune ressource unifiée pour recueillir les informations croisées sur les publications scientifiques internationales en informatique.

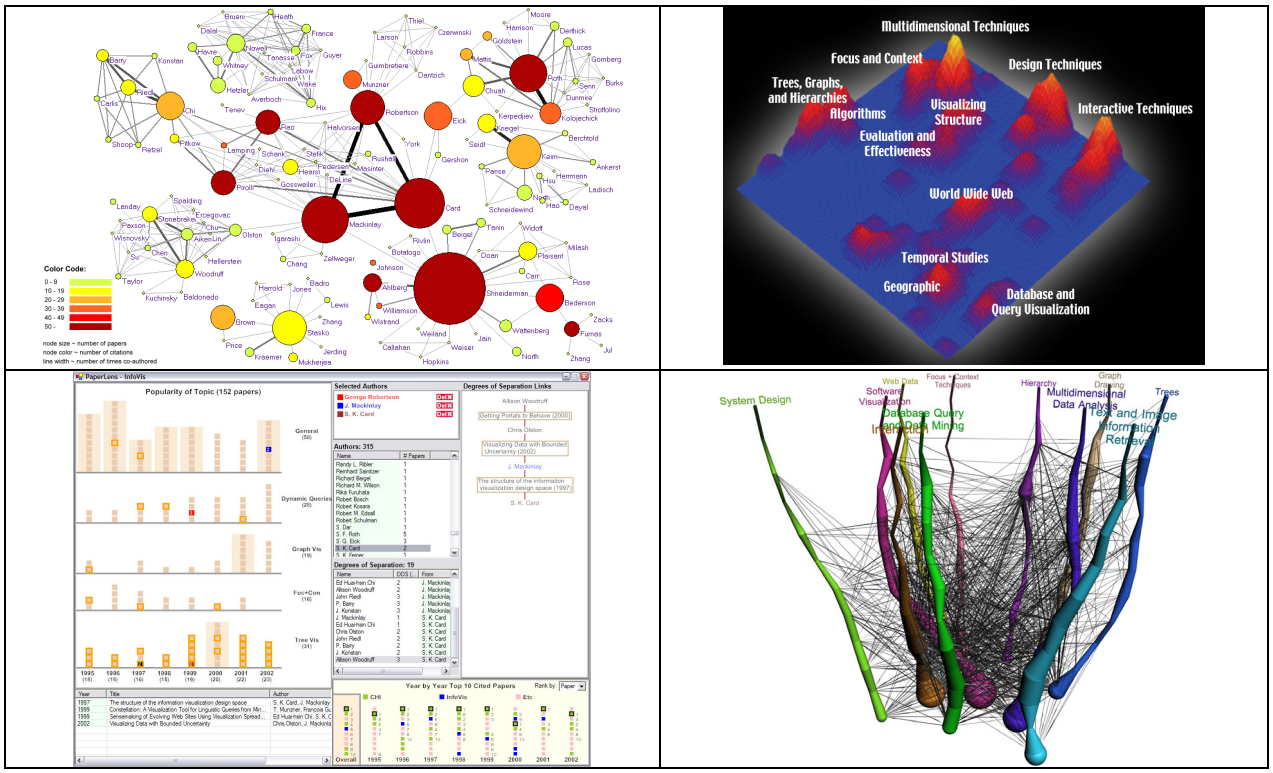


Figure 3 Images d'écran des quatre premiers prix en 2004, université d'Indiana en haut a gauche, PNNL en haut à droite, Microsoft Research en bas à gauche et l'université de Sydney en bas à droite.

Un de nos buts était de simplifier le benchmark et n'inclure qu'un seul jeu de données et un nombre restreint de tâches, afin de faciliter les comparaisons.

Jeux de données

La constitution du jeu de données a été un défi bien plus important que nous ne l'avions imaginé. Nous avons fait l'hypothèse que les articles et les auteurs les plus importants en visualisation d'information devaient être référencés par les articles publiés au symposium InfoVis. L'étude des citations à partir des articles publiés à InfoVis nous semblait à la fois focalisée dans le domaine et complet. Que signifierait une publication importante en visualisation d'information qui n'aurait été que très peu ou pas citée par les articles d'InfoVis ?

Pour recueillir l'ensemble des publications référencées par les articles publiés à InfoVis, nous nous sommes appuyés sur les bibliothèques numériques de IEEE et d'ACM. Les publications d'ACM sont disponibles avec une grande quantité de métadonnées comme les noms des auteurs associés à un numéro unique permettant de les identifier de façon sûre, ainsi que la liste des citations liées à leur entrée dans la bibliothèque numérique grâce à un identificateur unique d'article.

La réalité s'est révélée beaucoup moins simple. Les métadonnées disponibles sur les articles des symposiums InfoVis sont gérées par IEEE et sont beaucoup moins fournies que celles d'ACM. De plus, les données disponibles sur le site d'ACM sont très bruitées. Par exemple, il existe cinq références « uniques » pour Jock Mackinlay. IEEE quand à elle ne donne pas la liste des citations par article donc cette liste n'apparaît pas sur le site d'ACM. Enfin, ACM utilise une méthode semi-automatique pour résoudre les références entre les citations (qui sont des chaînes de caractères) et les articles stockés dans la bibliothèque, et cette résolution fonctionne relativement mal.

Nous avons tenté d'extraire automatiquement les références et de les nettoyer. C'est un travail très compliqué et nous n'avons trouvé aucun système automatique pour le réaliser convenablement. Nous avons donc extrait manuellement les données des articles PDF des 8 années du symposium disponibles sur la bibliothèque numérique. Nous avons ensuite cherché semi-automatiquement les articles cités dans la bibliothèque numérique d'ACM et nous avons extrait ces articles lorsqu'ils existaient (et qu'on les trouvait). Nous avons aussi unifiés manuellement les publications non incluses dans la bibliothèque numérique d'ACM.

Au total, le jeu de données contient 614 descriptions d'articles publiés entre 1974 et 2004 par 1036 auteurs, citant 8502 publications. Il aura fallu plus de 1000 heures homme pour le constituer.

Tâches

Pour ce genre de données, nous avons proposés des tâches de haut niveau, laissant la plus grande latitude aux participants :

1. créer une représentation statique montrant une vue d'ensemble des 10 ans d'InfoVis
2. caractériser les domaines de recherches et leur évolution

3. Quelle est la position d'un auteur particulier dans les domaines trouvés dans la tâche 2 ?
4. Quelles sont les relations entre deux chercheurs ou plus ?

Déroulement de la compétition

Le déroulement a été très semblable à la première compétition. Seul le problème de qualité du jeu de données a provoqué plus de questions avec les participants et beaucoup plus de travail pour les organisateurs. Environ trente personnes ont participé à la constitution et au nettoyage des données.

Durant le dépouillement, l'évaluation de la qualité et quantité des découvertes est devenue plus importante cette deuxième année. En effets les auteurs ont commencé a en fournir au lieu de simplement décrire leurs systèmes. La limitation du nombre de jeux de données et tâches a rendu les comparaisons plus faciles que la première année mais néanmoins les taches pouvaient être interprétées de manière différente, entraînant des réponses encore très diverses et rendant les comparaisons souvent délicates.

Résultats

Nous avons reçu 18 soumissions venant de 6 pays (USA, Canada, Allemagne, France, Australie, Pays-Bas), 7 soumissions étudiantes. Nous considérons que c'est une nette progression de la compétition, tant du point de vue de l'audience que de la qualité des soumissions.

Nous avons finalement retenu 12 soumissions dont 4 ont eu un premier prix (Figure 3) :

1. Weimao Ke et al. de l'université d'Indiana
2. Pak Chung Wong et al. du Pacific Northwest National Laboratory
3. Bongshin Lee et al. de Microsoft Research et de l'université du Maryland
4. Adel Ahmed et al. de l'université de Sydney.

Douze équipes ont été aussi retenues, chacune montrant des éléments intéressants mais aucune ne répondant à toutes les questions de manière convaincante.

Nous avons été agréablement surpris par la grande diversité des solutions proposées. Deux des trois premiers prix (Indiana et PNNL) ont une longue expérience dans le domaine de l'analyse des données. L'équipe de Microsoft a une grande expérience dans l'interaction et leur système est très interactif pour afficher les multiples facettes du problème pendant l'exploration des données. Il est possible que pour des données aussi complexes, une seule visualisation ne permette pas de répondre à toutes les questions et que les réponses ne puissent venir que d'analyses supplémentaire ou d'interactions plus sophistiqués.

Plusieurs systèmes utilisaient des placements de graphes par nœuds et liens. Il est frappant de voir la grande diversité des résultats en terme de placement et de coloriage, ainsi que la compétence que peuvent acquérir des experts à la lecture de graphes si denses qu'ils nous paraissent opaques. Très peu de ces graphes sont vraiment lisibles pour les profanes, mais les graphes animés sont souvent plus expressifs que les graphes statiques.

LEÇONS TIREES

Voici les quelques leçons que nous avons tirées et qui nous semblent transférables à d'autres compétitions :

- Ne pas sous-estimer la difficulté et le temps nécessaire à créer un jeu de données de bonne qualité. Le temps de constitution du jeu de données pour 2004 a probablement dépassé les 1000 heures homme.
- Participer à la compétition de visualisation d'information prends autant de temps que la rédaction d'un article long. Il faut que les bénéfices soient perçus comme comparables (ce n'est pas encore le cas).
- D'un point de vue logistique, nous avons été débordés par les problèmes de vidéo, la plupart des groupes n'étant pas familiers avec la création de vidéos à partir de leurs systèmes et produisant des fichiers très volumineux.
- Il y a des manières très différentes d'accomplir les tâches demandées. Par exemple, l'utilisation d'InfoZoom, initialement destiné à visualiser des tables, pour comparer des arbres nous a surpris et s'est avéré extrêmement efficace. Il convient donc d'exprimer les tâches de la manière la plus abstraite possible et de suggérer des tâches plus précises ensuite pour autoriser les surprises.
- La comparaison des résultats de la compétition reste difficile. Nous avons explicitement limité les résultats à trois catégories : refusé, accepté et vainqueur d'un prix parmi les trois ou quatre.
- Tous les résultats sont intéressants (en principe). La compétition n'est qu'une excuse à la construction d'un ensemble de benchmarks. Les résultats partiels sont donc encouragés car une technique ou un outils très adapté à une tâche particulière est utile à connaître, voir [13] pour s'en convaincre.
- La synthèse des résultats est aussi importante que le résultat de chacun des participants. Nous demandons un résumé de deux pages en PDF et surtout un formulaire HTML relativement structuré qui fournit des réponses précises aux tâches à accomplir.
- Sans un formulaire précis et des exemples de réponses, beaucoup de soumissions ne répondent pas aux questions posées. Par exemple, il y a plusieurs niveaux de réponses possibles à la question « à quelle heure part le premier train rapide de Paris à Nantes le jeudi matin ? » Une réponse est : « mon système permet de répondre a la question facilement ». Une autre est : « regardez le premier chiffre du formulaire affiché par mon système». Une troisième est : «5h50». La réponse attendue est : « 5h50, c'est le premier chiffre bleu affiché dans la première colonne du formulaire produit par mon système en spécifiant la date par telle interaction etc. »

CONCLUSION

Après ces deux premières compétitions, le symposium d'IEEE sur la visualisation d'information a décidé de continuer en gardant deux responsables par an, chacun restant deux ans en alternance de manière à pérenniser

les savoir-faire. Georges Grinstein continue donc l'année prochaine et tentera de simplifier les données et normaliser davantage les tâches, tout en préservant l'aspect exploratoire nécessaire à la compétition. Vous pouvez donc participer à cette compétition, enrichir les résultats des benchmarks et montrer les avantages des systèmes que vous concevez et réalisez. Vous pouvez toujours utiliser les benchmarks en dehors de la compétition et envoyer vos résultats sous une forme normalisée constituée d'une page HTML structurée. Ces résultats seront publiés sur le site de benchmark d'InfoVis et pourront servir de référence aux travaux à venir en visualisation d'information. Nous vous invitons aussi à utiliser les benchmarks déjà existants si vous devez présenter des résultats de visualisation afin de faciliter la comparaison avec des systèmes existants.

Parallèlement à la compétition de visualisation d'information, nous allons tenter de transférer notre expérience dans le domaine plus général de l'interaction homme-machine en organisant la compétition d'interaction à UIST 2005. Cette compétition aura une composante plus ludique car une partie de la compétition aura lieu en direct lors de la conférence. Encore une fois, notre objectif déguisé mais avoué est de constituer des benchmarks utiles pour permettre le choix de techniques d'interactions basé sur des éléments tangibles.

Notre objectif à plus long terme est d'améliorer les techniques d'évaluation utilisées en visualisation d'information et plus généralement en interaction homme-machine afin d'améliorer la qualité des systèmes à venir, d'éclairer les concepteurs et éventuellement de faciliter la mise au point et la vérification de modèles plus théoriques.

REMERCIEMENTS

L'organisation des compétitions de visualisation d'information a été soutenue activement par les organisateurs du symposium d'IEEE sur la visualisation d'information (InfoVis). Les réflexions sur la compétition ont été nourries d'innombrables discussions avec des membres de la communauté de visualisation d'information que nous voudrions remercier collectivement ici. Merci à Georges Grinstein de nous avoir rejoint en 2004, de son travail et de ses réflexions, en lui souhaitant bonne chance pour l'année prochaine. Enfin, la constitution de données de bonne qualité pour la compétition de 2004 a demandé une quantité de travail insoupçonné qui a mis a contribution plusieurs équipes d'étudiants. Nous voudrions les remercier ici : Caroline Appert (Univ. Paris-Sud, France), Urska Cvek, Alexander Gee, Howie Goodell, Vivek Gupta, Christine Lawrence, Hongli Li, Mary Beth Smrtic, Min Yu and Jianping Zhou (Univ de Mass. à Lowell) pour leur aide pour l'extraction manuelle des références bibliographiques. Pour la compétition 2003, nous voudrions remercier Cynthia Parr, Bongshin Lee (Univ. du Maryland) et Elie Dasa (Institut Pasteur).

Nous voudrions aussi remercier les sponsors qui nous ont permis de récompenser les premiers prix : le Hive Group, ILOG et Stephen North à titre personnel.

BIBLIOGRAPHIE

[1] Chen, C., Czerwinski, M. (Eds.) Introduction to the Special Issue on Empirical evaluation of information visualizations, International Journal of Human-Computer Studies, 53, 5, (2000), 631-635.

[2] Komlodi, A., Sears, A., Stanziola, E., InformationVisualization Evaluation Review, ISRC Tech. Report, Dept. of Information Systems, UMBC. UMBC-ISRC-2004-1 http://www.research.umbc.edu/~komlodi/IV_eval (2004).

[3] Plaisant, C. The Challenge of Information Visualization Evaluation, *in Proceedings of the working conference on Advanced visual interfaces (AVI 2004)*, pp. 109—116, Gallipoli, Italy, ACM Press, 2004.

[4] Munzner, T., Guimbretière, F., Tasiran, S., Zhang, L. and Zhou, Y., TreeJuxtaposer: Scalable tree comparison using Focus+Context with guaranteed visibility. *ACM Transactions on Graphics, SIGGRAPH 03* (2003) 453-462

[5] Hong, J. Y., D'Andries, J., Richman, M., Westfall, M., Zoomology: Comparing Two Large Hierarchical Trees, in *Poster Compendium of IEEE Information Visualization* (2003)

[6] Spenke, M., Beilken, C., InfoZoom - Analysing Formula One racing results with an interactive data mining and visualization tool, Ebecken, N. *Data mining II,* (2000), 455–464

[7] Auber, D., Delest, M., Domenger, J-P., Ferraro, P., Strandh, R., EVAT - Environment for Visualization and Analysis of Trees, in *Poster Compendium of IEEE Information Visualization* (2003)

[8] Sheth, N., Börner, K., Baumgartner, J., Mane, K., Wernert, E., Treemap, Radial Tree, and 3D Tree Visualizations, in *Poster Compendium of IEEE Information Visualization (*2003)

[9] Morse, D. R., Ytow, N., Roberts, D. McL., Sato, A., Comparison of Multiple Taxonomic Hierarchies Using TaxoNote, in *Poster Compendium of IEEE Information Visualization* (2003)

[10] InfoVis Contest 2003: www.cs.umd.edu/hcil/iv03contest

[11] InfoVis Contest 2004 : www.cs.umd.edu/hcil/iv04contest

[12] Repository : www.cs.umd.edu/hcil/InfovisRepository

[13] Mullet, K., Fry, C., Schiano, D. (1997) "On your marks, get set, browse! (the great CHI'97 Browse Off)", Panel description in ACM CHI'97 extended abstracts, ACM, New York, 113-114

Compus
Visualization and Analysis of Structured Documents For Understanding Social Life in the 16th Century

Jean-Daniel Fekete
Ecole des Mines de Nantes
4, rue Alfred Kastler, La Chantrerie
44307 Nantes Cedex, France
+33 2 51 85 82 08
Jean-Daniel.Fekete@emn.fr

Nicole Dufournaud
Université de Nantes, Faculté des Lettres
Chemin de la Censive du Tertre
44312 Nantes Cedex 3, France

Nicole.Dufournaud@humana.univ-nantes.fr

ABSTRACT
This article describes the Compus visualization system that assists in the exploration and analysis of structured document corpora encoded in XML. Compus has been developed for and applied to a corpus of 100 French manuscript letters of the 16th century, transcribed and encoded for scholarly analysis using the recommendations of the Text Encoding Initiative. By providing a synoptic visualization of a corpus and allowing for dynamic queries and structural transformations, Compus assists researchers in finding regularities or discrepancies, leading to a higher level analysis of historic source. Compus can be used with other richly-encoded text corpora as well.

Keywords
Information Visualization, Structured Documents, SGML, TEI, XML, XSL, Computers and the Humanities, History.

INTRODUCTION
Exploration and analysis of historical textual corpora is difficult. Researchers are faced with large numbers of documents that have never been studied or even read, seeking to "interesting" historic facts that, once structured will confirm or contradict existing hypothesis. Researchers currently rely on their notes, files and their own memories. Notes and files are hard to search, update and share among researchers. Memory is not thorough and prone to error. Will it change in the 21st century?

The field of "computers and the humanities" has been very active in the last decade, recently summarized in [13]. Most of the work has focused on data formats and conventions for interchange. Several digital libraries containing textual archives for research in humanities are now accessible from the Internet, such as [17, 5]. XML [4, 7] promises to be the next archival format for textual material in digital libraries[10], however, very few tools exist to support exploration and analysis using XML encoded documents [19, 15, 16].

This article describes *Compus*, a system designed to support and improve the process of finding regularities and discrepancies out of a homogeneous corpus of encoded textual documents. Developed in close collaboration with French Social History researchers, Compus visualizes a corpus of documents encoded in XML and supports refinement through dynamic queries [1] and structural transformations.

The next section describes our motivations for designing Compus and some details about the encoding of texts using the recommendations from the Text Encoding Initiative [21]. The third section presents Compus using examples from our corpus. The fourth compares Compus with related work. We conclude with a discussion of the use of Compus for research in other fields of humanities such as literature or linguistics before concluding.

MOTIVATION
This work started as an experiment in using structured documents and the recommendations of the Text Encoding Initiative for conducting research in early modern history [8]. We had to explore a corpus of letters of Clemency (*Lettres de Rémission*) kept in the regional archives of Nantes. These letters are manuscripts from the 16th century, administrative handwritten copies of letters of Clemency usually granting pardon from the King or the Queen. These early modern manuscripts have not been thoroughly studied and have not been transcribed. The historical work consisted of transcribing 100 letters, as in Figure 1, issued between 1531-1532 and studying "interesting" details of the social life at that time, just before Brittany was linked with France.

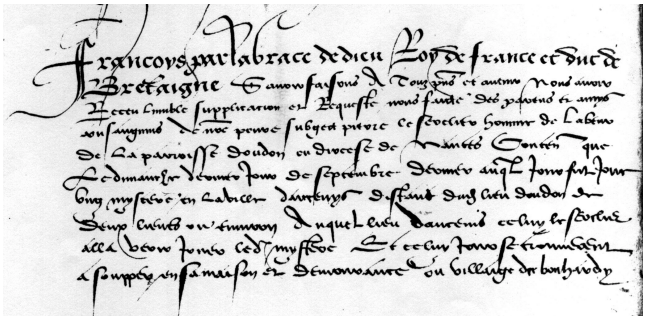


Figure 1: Beginning of a "Lettre de Rémission" of 1520

Historians are interested in these letters because they describe details of everyday life as well as habits, customs and social relations. The topics they address vary from the architecture of the city of Rennes' jail, to the vocabulary of love and hate. Their extensive analysis is almost impossible but, by reading a large sample, historians are able to extract details that match their research domains and use the letters to correlate their theories or find counter examples of established theories.

In general, given a corpus, a historian will first search for a list of topics of interest. A topic is of interest if the historian is familiar with it and either several documents address it or a document presents new or unexpected facts about it. The former case needs further analysis whereas the latter is already an "interesting historical finding" on its own. In our corpus, criminality is the main recurrent topic, but the social role of women also appears in several letters, as well as the use of weapons. To further analyze a recurrent topic, the historian uses different methods such as counting and statistics, correlation with other topics, linguistic analysis (i.e. what words were used to qualify crimes depending on the social position of the criminal), factual observations (i.e. women were running taverns).

Once encoded, our corpus can be considered as a semi-structured database of historical facts. Research was done in two steps: transcription and low-level encoding for building the database, and analytical encoding and exploration for Social History research.

To create the database, the transcribed letters have been encoded using the XML/TEI format that offers a high expressive power for describing paratextual phenomena such as abbreviations, insertions, corrections and unreadable portions of text.

TEI is a set of rules, conventions and procedures for encoding textual documents for interchange among different fields of the humanities. It relies on SGML [11] and has been designed to gather and formalize the practices of various fields to avoid the proliferation of incompatible formats. TEI is now turning to XML with few changes for users thanks to automatic document converters.

Once encoded in XML/TEI, the beginning of the letter in Figure 1 looks like this:

```
<lb n="1"/><name reg="Francois Ier">Francoys
</name>, par la grace de Dieu, roy de France et
duc de
<lb n="2"/>Bretaigne, savoir faisons a tous
<abbr>presens</abbr> et a venir,  nous avoir
<lb n="3"/>receu l'umble supplicacion et requeste
nous faicte des <s ana="structuresociale-parente">
parens et amys
<lb n="4"/>consanguins</s> de <abbr>notre</abbr>
povre subgect <name key="PL" ana="suppliant
masculin">Pierre Leserclier</name>, <s ana="crime-
statut-social">homme de labeur</s>
<lb n="5"/>de la paroisse d'<rs type="toponyme">
Oudon</rs> ou diocese de <rs type="toponyme">
Nantes</rs>, <abbr>contenant</abbr> que
<lb n="6"/>le <date value="30/9/1520" ana="crime-
date">dimanche, dernier jour de septembre dernier
</date>, <abbr>auquel</abbr> jour fut joué …
```

The encoding expresses three levels of phenomena: typographic or lexical, semantic and analytic.

1. Lexical and typographic phenomena: lines and pages are numbered using **lb** and **pb** tags, errors or omissions are noted (**sic**), as well as deletions (**del**), insertions (**ins**), unclear parts (**unclear**) and abbreviations (**abbr**);
2. Semantic encoding: names, dates and places are tagged as such and normalized using XML attributes. At this stage, the corpus is a database ready for interchange.
3. Analytical encoding: topics of interests are chosen by the historian and described in a sub-document. All segments of text in the corpus related to these topics are then linked to the topic of interest using the **ana** attribute. In the example above, the date of line 6 is marked as the date of the crime using the **ana** attribute.

This step requires the Historian to select a set of topics, describe them in a separate file. For example, the analytical topic "**crime-date**" is described in the sub-document by the following line:

```
<interp id="crime-date" type="date" value="Date of
the crime">
```

Topics are usually grouped using **interpGrp** (interpretation group) tags. For example, the "masculine" and "feminine" topics are grouped as "sex" like this:

```
<interpGrp id="sex">
 <interp id="masculine" value="male person">
 <interp id="feminine" value="female person">
</interpGrp>
```

Each portion of text related to each topic is then linked to the topic using the **ana** tag attribute, as in line 4 of the manuscript where the person is linked to both **masculin** and **suppliant**. When the region is already lexically or semantically tagged, like **name** in the previous example, the **ana** attribute is added in the element. Otherwise, an "**s**" element is created to delimit the region of interest as in line 4 and the **ana** attribute is set there. These mechanisms and their rationales are described at length in [21].

These encoding rules may seem very tedious but they are closely similar to the traditional practices in history, adding formal structure to the process of describing and analyzing manuscripts. Analytic encoding is a mechanism similar to maintaining manual files and notes for each document of the corpus. The traditional historical practice is not changed but adapted for the digital medium.

From there, traditional analysis of historic corpora is a well-established process whereby a historian starts from the sources and, through several levels of analysis, tries to find regularities or discrepancies and to structure the human history or find exceptions to supposed structures.

For this higher-level analysis, the historian is on her own and relies on her memories, files and notes. The Compus

system has been designed to help and support the historian in this task of search and discovery.

THE COMPUS SYSTEM

Figure 2 shows the Compus system. At startup, it contains three panes on a tab group. The main pane displays one vertical colored bar per XML document. Each XML element is assigned a color and a starting and ending index, corresponding to the character offset where the element starts and ends. A document is visualized as an ordered collection of colored segments using a space filling approach comparable to [14]. Since XML elements are nested, inner elements overlap outer ones.

For example, the sample XML document in boldface:

```
0         1         2         3         4
0123456789012345678901234567890123456789012345678
<A>abcd<B>efgh</B><C>ijkl<D>mnop</D></C>qrst</A>
```

is first converted into the following list of intervals:

A=[0,48[, B=[7,18[, C=[18,40[, D=[25,36[

A color is then associated with each element name and the document is displayed. Each document is given the same amount of space, usually a thin vertical rectangle. The rectangle is considered as a long line that wraps. In this case, the line would have the following colors:

A: [0,7[, B: [7,18[, C: [18,25[, D: [25,36[, C: [36,40[, A: [40,48[

Wrapping the line every 5 pixels produces the following display:

Index	Color				
0	A	A	A	A	A
5	A	A	B	B	B
10	B	B	B	B	B
15	B	B	B	C	C
20	C	C	C	C	C
25	D	D	D	D	D
30	D	D	D	D	D
35	D	C	C	C	C
40	A	A	A	A	A
45	A	A	A		

For displaying a corpus, each document is displayed in a given order (date in our corpus) and is given the same rectangle. A scale factor is applied to the original indexes so that the longest document fits in the whole rectangle and all documents are comparable in length. Corpora displayable by Compus are limited by the screen size minus the width of the list box on the right, the scrollbar and a separator line, leaving room for about 500 documents on 1280x1024 screens. Our corpus contains 100 documents so each rectangle is 5 pixels wide, as shown in figure 3. More can be visualized either by scrolling –hiding some documents – or by splitting the screen vertically - thus trading more documents for less space per document.

By applying this visualization to a corpus, users can at once compare document sizes and overall structure. As Figure 2 clearly shows, four parts are visible on our corpus, due to a change in texture. They correspond to the TEI header, the body of the text, a glossary and the description of the analytical categories. The 27th letter is much longer than the others and the first letter exhibits a gray zone at the end of the header or at the beginning of the body. The length of the 27th letter has even surprised the historian who encoded it. A judge has been found guilty of prevarication and asks for a special grace (something very specific to 1531). The particular gray zone is a **revisionDesc** element containing the description of changes made to the document. The first document has been subject to many more modifications than the others to fix rules we found suitable for the whole corpus. The mixed color of each part comes from the dominant encoding. The text part is full of abbreviations in maroon and dates in dark red. With just this first overview visualization, we can glean important clues about the contents of the corpus.

In the next subsections, we describe color allocation, how to interact with this representation, how to perform structural transformations and how to sort and export data, along with examples of uses.

Color Allocation

Each element type (i.e., tag name) is automatically assigned a color. Structured documents usually contain elements of various lengths, ranging from the full document to zero for empty elements. We compute the average size of each element and use three different color sets for large elements, middle sized elements and small elements. Large elements (over 100 characters long in average) use gray values, from black to white. Middle sized elements (from 10 to 100 characters) use the color scheme described by Healey [12]. Small elements use saturated red, green and blue colors. Threshold for color sets can be configured, but we found the 10/100 values well-suited to actual documents.

With this allocation scheme, larger elements are less visible than middle sized elements and small elements are vivid and can be distinguished. When more colors are required, the automatic color allocation cycles through the color sets. Interaction, as described in the next section, is useful to limit the number of visualized elements to improve search efficiency and focus on specific phenomena.

Interaction

ToolTips identify the document and the structural nesting of the pointed position. Clicking on the position displays the selected document in the pane called “Work”.

Color control and selection

Healey [12] has shown that 7 colors can be searched very quickly thanks to properties of our perceptual system.

When a user starts Compus, her screen visualizes all the documents with more than 7 colors. Large scale structures are visible through dominant colors, but users are usually searching for more precise phenomenon such as correlation or distribution of events. Moreover, when a color is assigned to each element name, nested elements disappear. For example, in `<name><unclear>Axe</unclear></name>`, the color of the **name** element will disappear under the color of the **unclear** element.

To control which elements are colored, the right list box displays the name of all the elements used in the corpus with their type and assigned color. Selecting an item hides the elements of that name on the visualization panel, revealing any element hidden behind it. Several items can be selected at the same time to focus on a smaller set of elements. Furthermore, users can control how colors are assigned to elements, either by right-clicking on the element in the list to popup a color selector or by triggering the color optimization menu that re-applies the color allocation to the currently visible set of elements.

With the list, users start by hiding unwanted elements and optimize colors to highlight specific phenomena. For example, selecting all elements but **unclear** in the scrollable list reveals documents harder to read because their density of colored segments is higher than the others. It turns out that our documents are all less readable at their end than at their beginning, due to the fact that they are handwritten copies of the original letters sent directly from the King to the requester and that the copyist was probably tired at the end. This distribution is visible from the variation of density of color on each document rectangle. By reducing the number of visible up to 7, users can improve their search time by relying on their perception system.

Visualization types

Because the hierarchical structure of elements creates a highly fragmented representation – i.e. a distribution – users might chose to see an aggregated view of the elements in the documents. With this visualization type, the surface covered by each selected element is visualized with bar graphs. In the example of Figure 4 the corpus was filtered to show only the lexical elements (unclear, sic, deletions and additions), aggregated to reveal their relative importance in each document.

Sorting

Documents can be sorted according to a variety of attribute orders: e.g., number of visible elements, surface of visible elements and lexicographically according to visible elements. Number of elements is self-explanatory; surface is simply the sum of all visible elements sizes. Sorting by surface is useful for the **unclear** tag and shows documents from the hardest to read to the easiest. Lexicographic sort is more meaningful for transformed documents so we address it after the next section.

Once sorted, documents remain in that order until another sort is asked for. New regularities can be searched, like displaying a color for each copyist when the corpus is sorted in readability order. Poor copyists appear on the left and good ones on the right.

Structural Transformations

At this stage, very few things can be said about social history because TEI elements describe lexical and syntactic phenomena. To access the analytic markup, attribute values are needed.

Compus allows for structural transformations of the documents to either focus on specific elements, transform XML attributes into elements or perform more sophisticated transformations allowed by the expressive power of the XSLT [6] transformation language.

XSLT has originally been designed for expressing Style Sheets to format an XML document for printing or online display. XSLT applies rules to a source XML document to transform it into another XML document. We have integrated an XSLT processor into Compus to filter and refine the visualization of corpora.

XSLT rules have two parts: a match part and an action part. Conceptually, the XSLT processor does a traversal of the source document, searches for the most specific match at each node and applies the related action. This action returns a new structure or nothing. It may continue the traversal on sub elements of the source to build its new structure.

For example, only to process the body of a TEI document, the XSLT rule would be:

```
<xsl:template rule="/">
 <xsl:apply-templates select="//body"/>
</xsl:template>
```

This simple transformation extracts only the sub-tree starting at the **body** element of the document.

XSLT has two implicit rules saying that when no specific rule is defined, the text is copied and elements are ignored.

Since Compus only visualizes elements and not attributes, XSLT is used to change the documents structure so that encoded events of interest appear as elements. Some elements can also be discarded or added to adapt the visualization. For each set of interrelated phenomena we want to visualize, we build an XSLT rule set that selects the useful elements and translates some attributes into elements that will be displayed. When focusing on analytical matters, we translate any TEI analytical attributes to elements of the same name. A document source containing:

```
<name ana="criminal">Jehan Mace</name>
```

becomes

```
<criminal>Jehan Mace</criminal>
```

by applying the following rule:

```
<xsl:template rule="*[@ana]">
 <xsl:element name="{@ana}">
```

```
  <xsl:apply-templates/>
 </xsl:element>
</xsl:template>
```

Where the rule reads: any element name with the **ana** attribute defined produces an element named by the contents of the attribute. The real rule is more complicated since **ana** attributes may contain a list of analytical names. We convert the list into nested elements or duplicated elements depending on the kind of visualization type. To visualize analytical phenomena, we also apply the following transformations: dates are normalized by replacing their contents by the contents of the **value** attribute. This regularization is required because the syntax of dates used in 1531 are hard to parse automatically (see line 6) and the calendar has changed from Julian to Gregorian. In a similar fashion, the transformation use regularized versions of the text when it exists, encoded in the **reg** attribute. Finally, we use the **n** attribute of elements analytically linked to ages to have plain numbers in the transformed version of the document.

New XSLT rules can be typed directly into Compus in a pane called "Stylesheet" or loaded from a regular text file. Since XSLT is a language, it is able to import modules so rule sets are usually small, about 30 lines long. When Compus applies the rule set to a list of source documents, it adds a new visualization pane similar to the original one, as shown in Figure 5. Each pane has the name of its rule set so users may have more than one transformed pane if desired.

Once the rule set to visualize analytically the corpus is applied, we may want to focus on the life of women in the 16th century. We select only male and female elements in the element list, revealing the distribution of each. By selecting both criminal and feminine, we can see how many women are criminals (only two). The same can be done for victims (only two also). Relying on the corpus, crimes were a masculine matter.

Compus also infers the type of elements from their contents. Currently, Compus recognizes numeric types and dates. It also keeps track of empty elements, textual only elements and mixed elements. These types are displayed on the scrollable list next to the color square, using one letter per type: I for integer, R for real, D for date, T for text, / for empty elements and a space for mixed contents.

From there, users can search for new phenomena or export the new documents into files readable by spreadsheets or traditional information visualization tools such as Spotfire [2].

Sorting and Exporting

Compus can use element types inferred during structural transformation for two purposes: sorting and exporting.

When sorting with only one element visible, Compus uses the element type to perform the sort. When several elements are visible and all are typed, Compus asks for an order and sorts the documents using a lexicographic order.

The initial order of our corpus is by administrative date of decision, but the date of the crime is also marked using an analytical link. Therefore, we can sort crimes according to their type first and then to their date.

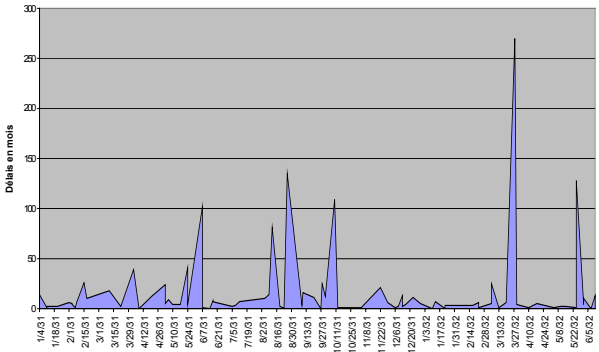


Figure 6 delay in months between the crime and its grace visualized from a Compus export.

When a sort is applied, all the views are sorted so one view can be used to express an order that may reveal a phenomenon visible in another view.

Element types are also used to export filtered documents. A traditional table is then produced containing one row per document and one column per selected element name. When a document contains several instances of a selected element, they are concatenated for text elements and an error is signaled for the other types.

We have exported views to perform calculations using a spreadsheet calculator and for other types of visualizations like showing the delay between the crime and the decision as shown in Figure 6. This delay decreased probably because the King wanted to keep Brittany happy just before it joined the French Kingdom. However, some crimes were old with a peak of 270 months when the criminal escaped, returned and got caught more than 22 years later.

Implementation and performance

Compus is implemented in about 5000 lines of Java 1.2 and relies on the XP Java library for parsing XML and the XT system for applying XSLT transforms[1]. It has been used on 300Mhz PCs with at least 128Mb of memory required to load the documents for fast redisplay during dynamic queries.

Applying the transformation of Figure 4 to the 100 documents takes about 15 seconds. Users never complained about performance, probably because the exploration and analysis are much longer.

[1] XP and XT are two free Java libraries and programs developed by James Clark and available at the following URL: http://www.jclark.com/xml/

RELATED WORK

Compus is related to Seesoft [9], Spotfire [2], the Document Lens [20] and the system described by Lecolinet et al. in [15].

Seesoft visualizes documents using vertical bars and colors. However, SeeSoft and Compus are quite different in their goals and details. Seesoft displays statistics associated with lines of code and extracted from an external database. Most problems with structured documents come from their dual aspects as databases and text. Compus uses a space filling approach for displaying a corpus whereas Seesoft clearly separates each document, providing a lower information density. Nested elements are not addressed by Seesoft, neither is color allocation and optimization.

Spotfire is a commercial system that implements the concept of starfield display with dynamic queries. It displays a database table using several types of visualizations (scatter plots, pie charts, histograms and bar charts) and provides interactive controls to filter visualized items. Large databases can be explored through these visualizations. Dynamic filtering helps in finding regularities and discrepancies by visual inspection, relying on our perceptual system. Compus is a specialization of the concept of starfield display with dynamic queries: it displays a corpus of structured documents instead of a flat database table, using a particular type of visualization not provided by Spotfire. The dynamic selection of visible elements is available in Spotfire but not the color allocation scheme. Spotfire can extract views from a database by SQL queries whereas Compus extracts views from an XML corpus by applying XSLT transforms.

The Xerox Document Lens is part of the WebBook environment and can display several pages using a perspective visualization. However, it is intended for displaying formatted Web pages and not structured documents: it is page oriented and not corpus oriented and does not contain functions to compare or sort documents since it relies on the natural page order.

Lecolinet et al. have designed a system to display a corpus for comparing text variants. A perspective wall is used to display each page of a document next to the other on a line and the documents stacked in columns. This representation simplifies the reading of aligned portions of text but is not adapted to corpus on unrelated documents like ours. Compus does not provide a focus+context view but a space filling view of the corpus. For interaction, Lecolinet's system relies on navigation in a static representation whereas Compus uses dynamic queries and structural transformations. The level of granularity is also different: Lecolinet focuses on aligned pages of a variant document whereas Compus focuses on XML elements contained in several documents.

DISCUSSION

Compus has been used by historians to initially perceive the relative importance of each phenomenon, then to quickly check hypothesis about correlation of phenomena or spot discrepancies, as described in the second section. Quickly checking for hypothesis is an important property of Compus since it is difficult for researchers to rely entirely on their memory to judge the frequency of events. Important events are often judged frequent and conversely frequent unimportant events are usually left out.

For example, our initial hypothesis was that just before 1532, more nobles would receive clemency (i.e., be pardoned) for political reasons. Compus helped reveal that the density of pardon given to nobles did not vary across the two years of the corpus. However, two important nobles did receive a pardon in the last letters. We have probably been influenced by these events to produce our hypothesis.

We will now describe the interaction of the various tools and some other experiments we have done on other corpora: the plays of Shakespeare and an Old English language corpus. Finally we will address some limitations of Compus.

Tools for working on structured documents

Researchers need to access several levels of granularity when working on a document corpus: the corpus, the document and various fragments of documents. Compus is suited to visualize a whole corpus but other tools are needed to read documents and their parts. One very effective view of the corpus is through a Web browser and indexes generated from the TEI documents. Figure 7 shows such a configuration.

We also mentioned spreadsheet calculators and information visualization systems to assist in the analysis of valued information, such as dates, age, time, frequency, etc.

Synoptic visualization has been found effective to explore a new corpus. This will become a more and more common situation with the development of digital libraries where users will need to quickly evaluate the scope of collections.

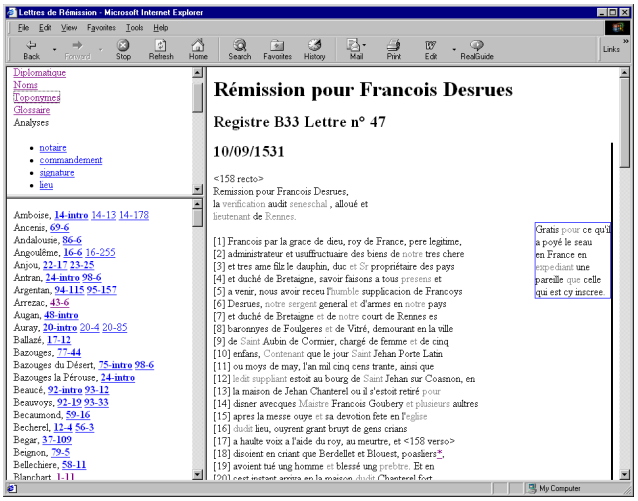


Figure 7: Hypertext configuration with various indexes to read and explore the corpus.

Exploring Shakespeare

We tested Compus on a corpus of 37 plays of Shakespeare encoded in XML [3]. Variation in sizes is obvious. "Henry the Eighth" exhibits larger **STAGEDIR** (stage direction) elements than the others, meaning that scenes happen in different places or that lots of events happen during the scenes. Even if the level of encoding is much lower than ours, Compus is still useful.

The Lampeter Corpus

The Lampeter corpus [18] "is an unusual historical corpus, consisting of 120 unique English pamphlets from the period 1640 to 1740." We received it as an answer to a request on the Internet for testing Compus. Without any knowledge of the corpus, we have been able to see several classes of documents. The corpus is organized into 6 topics: religion, politics, economy and trade, science, law and miscellaneous. We found that some structures were topic specific, like tables used mostly for science, but also for economy. Since the corpus has been selected for its use of the language, we applied a transformation to exhibit language use per topic. Figure 8 shows its visual form when applied a suited analytical filter. This visualization reveals that Latin was by far the most used language (not a surprise). That Greek was mostly used for religion and science whereas Latin was used everywhere but mostly for science, religion and law. French language was most used in political texts, revealing the tensions.

Limitations

One of the current limitations of Corpus is that the transformation language XSLT requires either extensive learning or technical assistance. It will not remain such a problem if technologies related to XML reach a level of popularity comparable to HTML but providing a simpler interface to perform the transformations would certainly help users.

Color allocation should be improved. We observed that users often need to allocate colors using the popup chooser. More investigations are required here.

It would be useful to manage larger corpora from Compus, although most homogeneous corpora analyzed by current historians are usually under 1000 documents, due to the time required to fill analytical forms manually. As explained in section 2, Visualizing more than 1000 documents could be done either by scrolling or splitting the screen vertically. We are interested in investigating both solutions but have not yet found a homogeneous corpus analytically encoded larger than 1000 documents to experiment with.

CONCLUSION AND FUTURE WORK

We have described Compus, a new visualization system suited to exploring corpora of structured documents. It has been mostly experimented on a corpus of historic documents but has also been tested on Shakespeare's plays and a literary corpus.

Compus displays an XML corpus by assigning colors to each element names and applying a space filling representation that visualizes the position and size of each element in each document. Interaction is used to focus on elements of interest. Since only XML elements are visualized, Compus integrates an XSLT processor that transforms structurally XML documents in other XML documents. Phenomena expressed through several encoding mechanisms can be translated into elements and visualized. Transformations are also useful to filter documents content and focus on specific parts of documents or on a set of tags. From there, users can notice global correlation and discrepancies. Compus can also sort document representations according to several orders to reveal other types of correlations.

We have shown how Compus has been successfully used by researchers in early modern history to explore a corpus made of 100 documents encoded in XML/TEI. We have also shown how Compus could be used as a visual interface for accessing databases of homogeneous structured documents. We are now exploring a new application domain with a corpus on biology and will study how practitioners use Compus.

Several library projects are working on such corpora or databases and their access is currently through forms and lists, comparable to FTP access before the Web existed (except for [16]). Compus is an effective tool for exploring the specific class of homogeneous corpora where it provides a global view and several insights to the contents.

We believe Compus is an effective complement to already existing tools like indexers, style sheet processors, spreadsheet calculators and information visualization programs. Rather than building an integrated environment to work on structured document corpora, Compus can be used as a visual explorer, and delegate specific tasks to the other tools.

As for Social life in the 16th century in Brittany described by the Letters of Clemency, alcohol was the main cause of murders. Women were often managing the houses, sometimes even taverns where both men and women were drinking. Comments are left to the patient reader.

ACKNOWLEGMENTS

Thanks to Catherine Plaisant, Wendy Mackay and Cédric Dumas for their advice and help in writing the article. Thanks to Stephanie Evans for cross proofing it.

REFERENCES

1. Christopher Ahlberg and Ben Shneiderman. Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays. In Human Factors in Computing Systems. Conference Proceedings CHI'94, pages 313--317, 1994. ACM.
2. Christopher Ahlberg Spotfire: An Information Exploration Environment SIGMOD REPORT, 25(4), pp. 25-29, December 1996.

3. Bosak J. Shakespeare 2.00. Available at http://metalab.unc.edu/bosak/xml/eg/shaks200.zip

4. Bray, T. Paoli, J. Sperberg-McQueen C. M. Eds. Extensible Markup Language (XML) 1.0, Recommendation of the W3 Consortium. Feb. 1998. Available at http://www.w3.org/TR/REC-xml

5. Caton, Paul. "Putting Renaissance Women Online," New Models and Opportunities, ICCC/IFIP Working Conference on Electronic Publishing '97, April 1997. See also at http://www.wwp.brown.edu/

6. Clark, J. XSL Transformations (XSLT) Version 1.0 W3C Working Draft. Available at http://www.w3.org/TR/WD-xslt

7. Cover, R. The SGML/XML Web Page, Available at http://www.oasis-open.org/cover/.

8. Dufournaud, N. Comportements et relations sociales en Bretagne vers 1530, d'après les lettres de rémission, Mémoire de Maitrise, Univ. De Nantes. Available at http://palissy.humana.univ-nantes.fr/cete/txt/remission/Memoire.pdf

9. Stephen G. Eick and Joseph L. Steffen and Eric E. Sumner Jr. Seesoft --- A Tool For Visualizing Line Oriented Software Statistics IEEE Transactions on Software Engineering, pp. 957-68, November 1992.

10. Friedland, L. E. and Price-Wilkin J. TEI and XML in Digital Libraries, Workshop July 1998, Library of Congress. Available at http://www.hti.umich.edu/misc/ssp/workshops/teidlf/

11. Charles F. Goldfarb and Yuri Rubinsky The SGML handbook, Clarendon Press, 1990.

12. Christopher G. Healey Choosing Effective Colours for Data Visualization Proceedings of the Conference on Visualization, pp. 263-270, IEEE, October 27- Nov 1 1996.

13. Nancy Ide and Dan Greenstein, Eds. Tenth Anniversary of the Text Encoding Initiative, Computer and the Humanities, 33(1-2), 1999.

14. Keim D. A.: Pixel-oriented Database Visualizations , Sigmod Record, Special Issue on Information Visualization, Dec. 1996.

15. E. Lecolinet and L. Likforman-Sulem and L. Robert and F. Role and J-L. Lebrave An Integrated Reading and Editing Environment for Scholarly Research on Literary Works and their Handwritten Sources DL'98: Proceedings of the 3rd ACM International Conference on Digital Libraries, pp. 144-151, 1998.

16. Marchionini, G., Plaisant, C., Komlodi, A. Interfaces and Tools for the Library of Congress National Digital Library Program Information Processing & Management, 34, 5, pp. 535-555, 1998.

17. The Oxford Text Archives, available at http://ota.ahds.ac.uk/.

18. Siemund, Rainer, and Claudia Claridge. 1997. "The Lampeter Corpus of Early Modern English Tracts." ICAME Journal 21, 61-70., Norwegian Computing Centre for the Humanities.

19. Randall M. Rohrer and David S. Ebert and John L. Sibert The Shape Of Shakespeare: Visualizing Text Using Implicit Surfaces Proceedings IEEE Symposium on Information Visualization 1998, pp. 121-129, 1998.

20. George G. Robertson and Jock D. Mackinlay The Document Lens Proceedings of the ACM Symposium on User Interface Software and Technology, Visualizing Information, pp. 101-108, 1993.

21. C. M. Sperberg-McQueen and Lou Burnard (eds.) Guidelines for Electronic Text Encoding and Interchange (TEI P3), Volumes 1 and 2, The Association for Computers and the Humanities, the Association for Computational Linguistics, and the Association for Literary and Linguistic Computing, 1994.

Color Plate

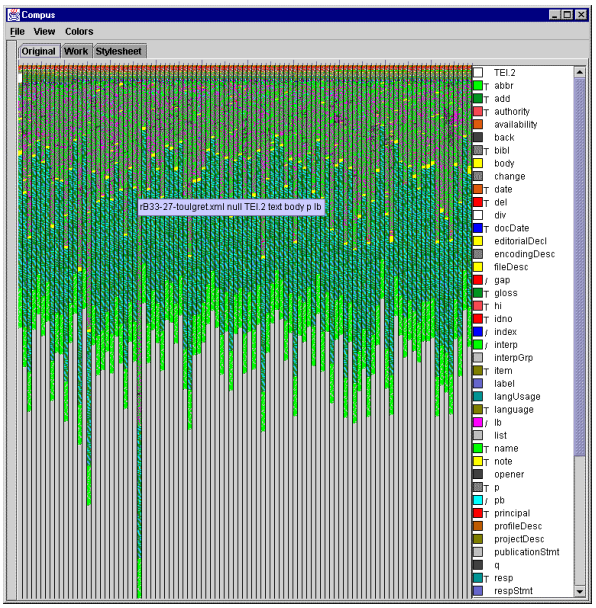


Figure 2: The Compus System showing a corpus of 100 structured documents.

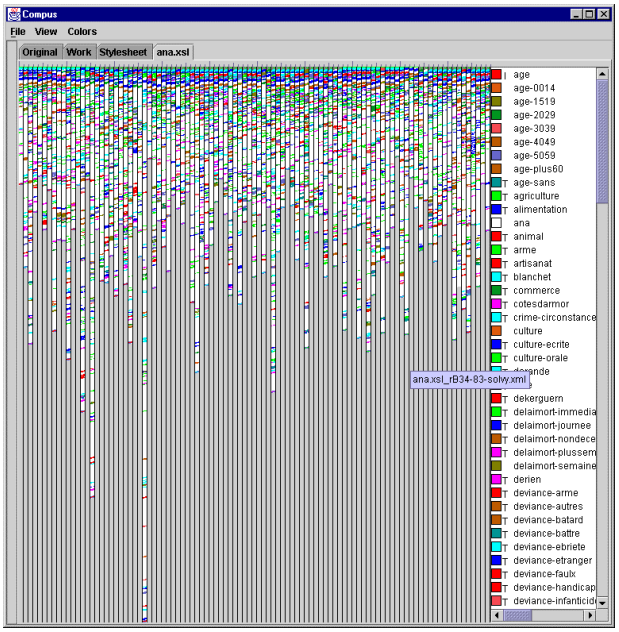


Figure 5: Visualization of the transformed corpus, i.e. analytical view of figure 2.

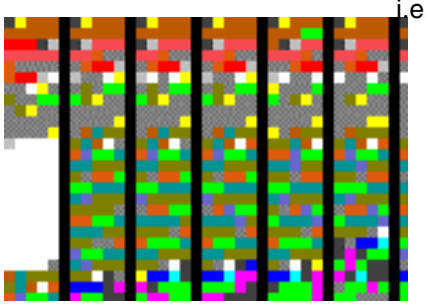


Figure 3: Close up view of the NW corner of figure 2.

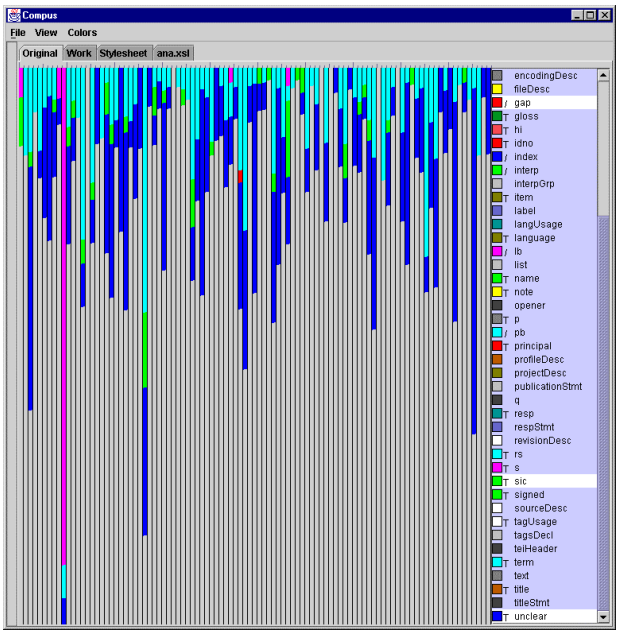


Figure 4: Aggregated elements view of lexical problems (add, del, sic, gap, unclear)

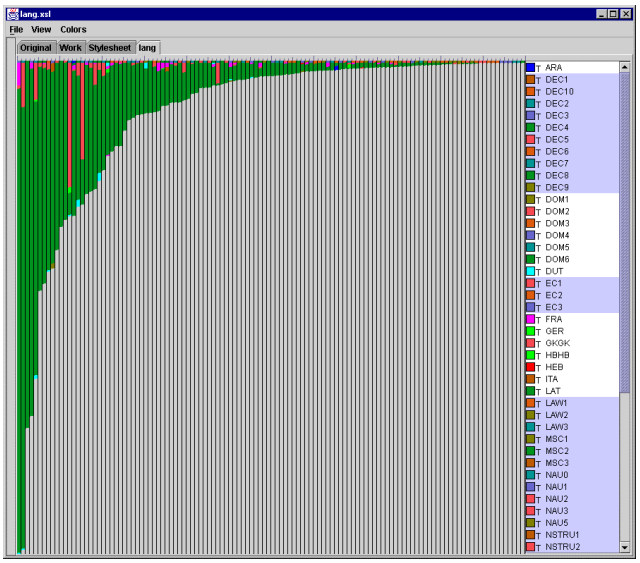


Figure 8: The Lampeter Corpus visualized by element surface showing the various amounts of non-English languages used in descending order. Latin is by far the most popular followed by Greek.

Peeking in Solver Strategies Using Explanations Visualization of Dynamic Graphs for Constraint Programming

Mohammad Ghoniem* École des Mines de Nantes
Hadrien Cambazard† École des Mines de Nantes
Jean-Daniel Fekete‡ INRIA Futurs/LRI
Narendra Jussien** École des Mines de Nantes

ABSTRACT
In this paper, we describe the use of visualization tools in the context of constraint programming. Specifically, we focus on the visualization of dynamic graphs using adjacency matrices, and show that visualization tools can provide valuable insights into the behavior of constraint-based solvers. This contributes to better understanding solver dynamics for teaching purposes, and can enhance the analysis and the optimization of constraint-based programs in connection with known and difficult research problems.

Categories and Subject Descriptors
I.3.3 Picture/Image Generation; Display algorithms, I.2.8 Problem Solving, Control Methods, and Search; Heuristic methods, D.2.6 Programming Environments, Graphical environments.

General Terms
Algorithms, Performance, Experimentation, Human Factors, Languages.

Keywords
Visualization of algorithms, Graph drawing algorithms for software visualization, Visual debugging, Constraint Programming, Finite Domain Solver.

Introduction
Solving a problem within the constraint-based programming (CP) paradigm consists in providing a user-defined model of the problem at hand through the declaration of its constitutive variables, and the constraints that relate them. Once the problem is modeled, a specialized solver – a sort of black box – takes care of the resolution process. CP languages have been successfully used in various application fields in both academic and industrial contexts. These fruitful applications include machine scheduling, sports scheduling, and routing problems - to name but a few.

However, CP programmers and end-users have very often been left without effective tools allowing them to reach a good understanding of the dynamic behavior of solvers, and further analyze, debug, and tune the programs and algorithms they implement.

In this paper, we start with a description of CP needs in terms of visualization along with a quick overview of works related to the visual analysis of constraint-based programs. This is followed by a description of the visualization tools we implemented to help understand, analyze and tune constraint-based programs. We elaborate on the use of advanced visualization techniques such as matrix-based visualizations of graphs and time filtering in order to handle large dynamic graphs. Further, we provide several use cases that describe and discuss the use of our tools. Finally, we highlight future works that can extend the present effort.

1 Visualization Needs of Constraint-Programming
A finite domain constraint problem is specified by the following elements:

- a finite set ν of finite domain variables;
- a finite set D that contains all possible values for variables in ν; in finite domain solvers, D is a set of non negative integers ranging over 0 to *maxint*;
- a function which associates to each variable v its current domain (a subset of D), denoted by D_v;
- a finite set of constraints C. Each constraint c in C defines a relation between the set of variables $var(c)$, the variables of c – i.e. a subset of ν.

A solution of the constraint problem is an assignment of the variables of ν to values in D (a *tuple*), such that all constraints in C are satisfied. All the constraints are thus solved, and the problem is said to have a solution.

When no such tuple can be found, the problem is said to be "over-constrained", and the user needs to relax some constraints to achieve feasibility. Very often, a problem may admit several solutions of various quality or cost. Explicitly, the solver always starts by propagating the immediate effects of the constraint set; this often results in the reduction of the variable domains, through the withdrawal of inconsistent values.

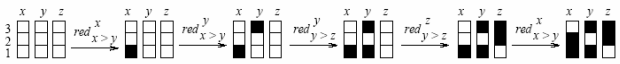


Figure 1: Applications of reductions to the system
$\{ x > y; y > z \}$ $(x,y,z) \in [1..3]^3$

Figure 1 shows the reduction of three variables x, y and z and two constraints $x > y$ and $y > z$. At the beginning, the initial variable domains, $D_x=D_y=D_z=\{1,2,3\}$, are represented by three columns of white squares. Considering the first constraint, it appears that x cannot take the value 1 because otherwise there would be no value for y such that $x>y$; $red^x_{x>y}$ eliminates this inconsistent value from D_x. The deleted value is marked with a black square.

* email: Mohammad.Ghoniem@emn.fr
† email: Hadrien.Cambazard@emn.fr
‡ email: Jean-Daniel.Fekete@inria.fr
** email: Narendra.Jussien@emn.fr

In a similar fashion, $\mathrm{red}^{y}_{x>y}$ filters the value 3 from the domain of D_y. Then, considering the constraint $y>z$, $\mathrm{red}^{y}_{y>z}$ and $\mathrm{red}^{z}_{y>z}$ withdraw respectively the sets {1} and {2,3} from D_y and D_z. Finally, a second application of $\mathrm{red}^{x}_{x>y}$ reduces D_x to the singleton {3}. The final solution is $\{x=3, y=2, z=1\}$.

If, after this stage, some variable domains are not reduced to singletons, the solver takes one of these variables and tries to assign it each of the possible values in turn. This enumeration stage triggers more reductions, which possibly leads to solutions, or failures, or additional enumerations on non-reduced variable domains. Obviously, the search space can be represented by a tree with an exponentially growing width where each level corresponds to a variable of the problem. For instance, in a problem having N binary variables, the search tree can contain a total of 2^{N+1} -1 nodes with 2^k nodes at level k of the hierarchy where k varies from 0 to N if no reduction is done, i.e., if all tuples are enumerated. Industrial problems usually bear thousands of variables with large domains and hundreds or thousands of constraints. Therefore, monitoring such huge hierarchies is definitely a challenging task.

Earlier works have mainly focused on representing such trees in view of supporting the analysis of constraint-based programs, and visualizing solver strategies. These works started with Spy [Brayshaw and Eisenstadt 1991], the Prolog Trace Package (PTP) [Eisenstadt 1984], the Transparent Prolog Machine (TPM) [Brayshaw and Eisenstadt 1990; 1991], and the Textual Tree Tracer (TTT) [Taylor et al. 1991]. Spy provides a linear textual trace of prolog execution using the Byrd Box model. PTP is also a linear textual trace with an extended range of symbols compared to Spy. TPM provides pretty much the same information in the form of an AND/OR tree with a detailed view of individual nodes of the tree. Graphical node-link visualizations were also made available in support of TPM as can be seen in [Brayshaw and Eisenstadt 1990]. TTT uses a non-linear textual notation providing a tree representation of Prolog close to the source code. An evaluation of these four methods is available in [Mulholland 1995].

Recent research projects such as *Discipl* [Deransart et al. 2000] produced a collection of tools aimed at the monitoring of constraint programs and, especially, the visualization of search trees such as the SearchTree visualizer packaged in Chip++ [Deransart et al. 2000] development suite. An important contribution of the *Discipl* project consisted in reaching a better understanding of the needs of CP users. That is, besides search trees, CP users would also be interested in the evolution of variable domains, and the evolution of activity within the graphs of variables, the graphs of constraints, and also within mixed graphs that involve constraints and variables interacting with each other.

In more recent works, authors applied radial space-filling visualization techniques similar to sunburst trees [Stasko et al. 1998] or interRings [Yang et al. 2002] to represent search trees in constraint-based programming applications such as "COMIND" [Lalanne and Pu 2000]. In CLPGUI [Fages 2002], search trees were represented as node-link diagrams in 3D space, and posting queries at certain nodes of the search tree was supported in order to guide the solver on runtime.

All visualization tools are confronted with the major challenge of representing exponentially growing data structures on a limited screen real-estate with an equally limited memory real-estate. [Fekete and Plaisant 2002; Abello and Ham 2004].

Building on the recommendations of the *Discipl* project, we tackle the problem from a different standpoint. Instead of focusing on the search tree per se, we decided to consider the network of variables and constraints that collaborate together to get the problem solved. For example, in Figure 1, the network connects the variables x and y to the first constraint ($x<y$), and the variables y and z to the second constraint ($y<z$.) Section 4 describes various uses of such networks, mainly the visualization of the activity of the variables and constraints during the resolution process.

We have also created activity networks based on "explanations" that are maintained dynamically by a newer generation of constraint solvers such as PaLM [Jussien and Barichard 2000] or Choco [Laburthe 2000]. We describe explanations and their visualization in section 5.

2 Visualization Tools for Activity Graphs

The activity in a network is simply the number of times a variable or a constraint is used over an interval of time. Activity can be represented by a set attribute (a set of timestamps where activity occurred) associated with either the vertices of the network or its edges, or with both. When at time t_0 a variable v is modified, its activity attribute will contain t_0. This modification will then trigger at time t_1 a constraint c that uses v, and add t_1 to the activity attribute of the edge linking v to c. It will also add t_1 to the activity attribute of c. In turn, c will reduce other variables at time t_i and hence, the attributed activity network builds up progressively.

Activity networks are potentially very large and dense, and are characterized by a varying connectivity or edge weight over time. Therefore, the user's vision becomes quickly clouded with traditional node-link representations of graphs, and it becomes difficult to follow how they evolve during the solving process, and to identify which parts of these networks are active. As a side note, we may recall that the size of the network refers to the number of variables and constraints in the problem.

2.1 Matrix-Based Visualization of Graphs

So far, visualization of graphs has mainly focused on node-link diagrams because they are popular and well-understood. However, node-link diagrams do not scale well. Their layout is slow and unstable, and they become quickly unreadable when the size of the graph and the link density increase. We use adjacency matrices instead of node-link diagrams to interactively visualize and explore large graphs, with thousands of nodes and any number of links. This technique relies on the well known property that a graph may be represented by its connectivity matrix, which is an N by N matrix, where N is the number of vertices in the graph, and each row or column in the matrix stands for a vertex. When two vertices V_i and V_j are linked, the corresponding coefficient m_{ij} in the matrix is set to 1, otherwise it is set to 0.

From a visualization standpoint, not only do we switch on or off the cell located at the intersection of V_i and V_j, but we use color coding as well when dealing with weighted links: the heavier the weight, the darker a link. Unlike node-link diagrams, matrix-based representations of graphs do not suffer from link and node overlapping. Virtually, every link (out of the N^2 links) in the graph can be seen separately (Figure 2). When dealing with directed graphs, columns represent the origin of edges and the

rows represent their endpoint vertices, although conventions may vary. With this technique, we can show as many links as the display hardware resolution allows (roughly 2 million pixels on a regular 1600 × 1200 display). Moreover, advanced information visualization techniques such as dynamic queries [Card et al. 1999], fisheye lenses [Carpendale and Montagnese 2001] and Excentric labels [Fekete and Plaisant 1999] enhance the exploration of large graphs, and push the matrix-based visualization one step further in coping with large networks.

2.2 Readability of Matrix-Based Representations

The main tradeoff of such a technique lies in the fact that vertices are no longer represented by a unique graphic symbol. They are rather distributed on both axes of the matrix. This is why users may often need some training before they get familiar with the matrix metaphor. In a controlled user experiment [Ghoniem et al. 2004] carried out on seven exploration tasks and nine graphs of different sizes and densities, we have shown that matrix-based representations perform significantly better than node-links for medium to large graphs (50 to 100 vertices) and for dense graphs with all tasks involved in the experiment, except path finding.

2.3 Related Works

Making sense out of network data often depends on the ability to understand its underlying structure. Therefore, cluster detection has been an active topic of research for a long time. Many works have concentrated on data analysis techniques in order to aggregate the graph into its main components. Bertin [1983] shows that it is possible to reveal the underlying structure of a network represented by a matrix through successive permutations of its rows and columns. This idea relies on the fact that the rows and columns of a matrix can be ordered according to a given criterion, which is another advantage of the matrix metaphor as ordering the vertices and links in a node-link diagram is not straightforward. In [Becker et al. 1995], the authors visualize the load distribution of a telecommunication network using a matrix but most of their effort aims at improving the display of a node-link representation such as displaying half-links or postponing the display of important links to minimize occlusion problems. More recently, in [Ham 2003], the authors implemented a multi-scale matrix-based visualization representing the call graph between software components in a big medical imagery application. In [Ghoniem, Jussien and Fekete 2004], we have shown that a matrix-based representation can be used to effectively grasp the structure of a co-activity graph and assess the activity taking place across time whereas the equivalent node-link representation was unusable.

2.4 Time Filtering

As already mentioned, all links in those graphs are weighted with the number of times the relation is actually established throughout computation. This helps enhance the static structure with dynamic information pointing out active relations. More precisely, we keep a full history of activity within the graph. Thus, we can dynamically query the graph for links that are active within a user-controlled time range and compare the amount of activity between links in that range. The user may visualize the activity in the graph throughout the entire resolution process or in a smaller time range whose bounds and extent are interactively parameterized. By sweeping the time range from the beginning to the end of the history, the user may play back the resolution process and see which links are established, when, and how often.

We also offer user-defined time slices. Simply put, a time slice is a time range between two events of interest. For instance, in our experiments, we were interested in activity between pairs of successive solutions. Our system computes the relevant time slices and allows the user to jump between successive time slices through direct interaction as well.

3 Visualization of Constraint-Variable Networks

Figure 2 shows the activity graph obtained by sorting one hundred variables $x_{i\in]100]}$ in domain [1..100] using 99 constraints: $\forall i \in]1,99], x_i < x_i+1$

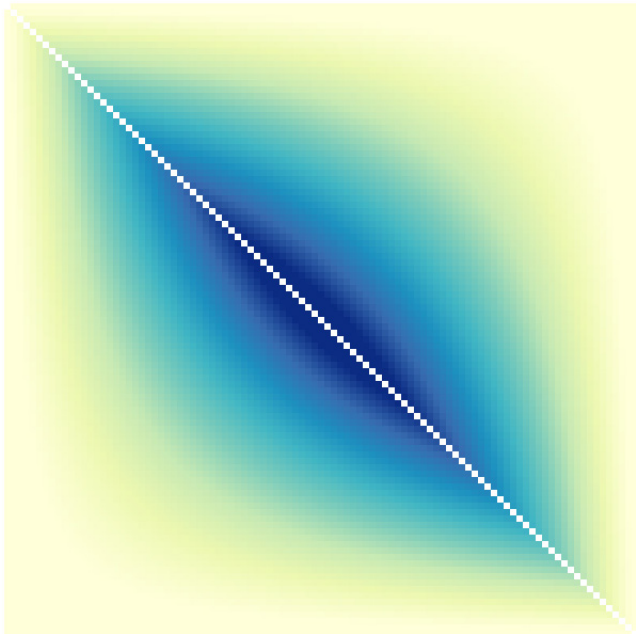


Figure 2: The Activity Graph of Constraints of the Sort Problem.

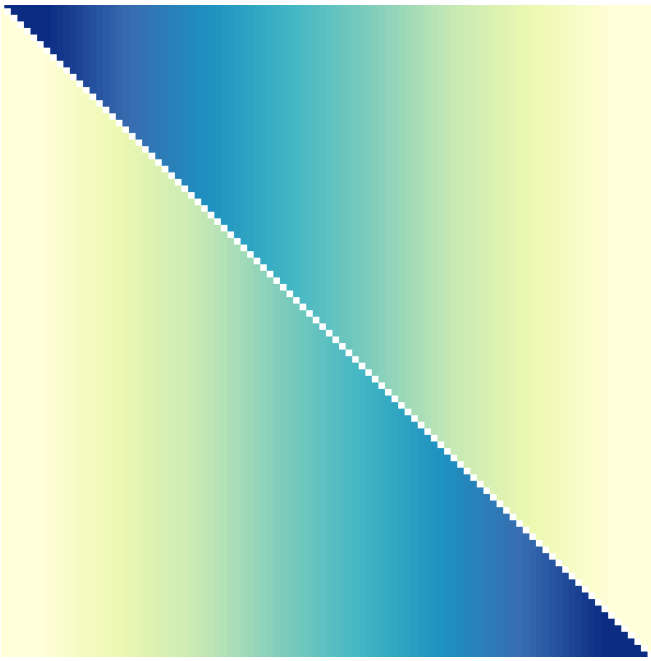


Figure 3: The Activity Graph of Variables of the Sort Problem.

Reductions are sufficient to lead to a solution. Sorting the variables is, therefore, a sheer (and long) propagation problem. On Figure 2, we display the graph of one hundred constraints involved in the resolution of the problem without variables. The edge activity is computed by omitting the intermediate variable when a constraint triggers another constraint. On Figure 3, we represent the graph of variables of the same problem, obtained conversely by omitting the constraints. These graphs provide much information about what happened during propagation. In fact, a strong interaction can be observed between close constraints (in the posting sequence), but we also see that all constraints are related to each other. Moreover, we can see that the farther the constraints from one another in the posting sequence, the less they interact (but they still have some level of interaction). Of course, all that has been said about constraints can be transposed about variables. Moreover, the middle constraints seem to interact more often than peripheral constraints: see the darker part in the center of the constraint graph. What do we learn? Those observations on the constraint graph help us understand why the propagation takes so long in such a problem. All constraints need to be propagated several times as they all have both a short-term and a long-term impact during the resolution process. Moreover, we illustrate here the interest of the visualization of the explanation network with regard to classical analysis of the constraint graph which is not able to provide any information. However, one question still needs to be addressed: why do middle constraints seem to be so active? For that, we need to observe the dynamics of propagation.

One of the key features of our system is that it makes it possible to visualize the dynamics of search and propagation. Animations are not easy to render on a printed paper. However, Figure 4 shows seven time slices from the constraint graph of our sorting problem. The visualization of the dynamics within the propagation phase of this particular problem reveals the existence of two clearly identified activity waves. Actually, this illustrates the way our solver works. The first constraint to be propagated is the last one in the posting sequence (constraints are not propagated upon posting but all at once) inducing a chain of modifications of the upper bounds of the variables through the awakening of all other constraints. Then, the other constraints are propagated, each modifying the lower bound of one variable. The two phases displayed in Figure 4 illustrate those two modification waves. Our animation is invaluable to illustrate such an intimate behavior of the solver as it provides information about the way propagation works. Moreover, this information explains why middle constraints appear to have more activity than other constraints. This is due to the fact that the propagation of those constraints impacts both the upper and lower bounds of the variables whereas the propagation of peripheral constraints impacts only one bound of the variables.

The added value of the visualization tools in this case can be summed up as follows:

1. Using matrix-based representations is essential when handling very dense graphs.
2. The stability of the layout of matrices and the lack of occlusions makes it possible to monitor dynamic graphs, and grasp the changes that occur therein in a pre-attentive fashion.
3. Visualization tools can help illustrate notions such as propagation more pedagogically, as in this example, thanks to proper animation. From this standpoint, visualization tools may be useful to some extent in teaching environments.

4 Visualizing Explanations

We worked with an "explained solver" instrumented to produce a trace. Thus, we could collect a fine grained history of the solver execution for postmortem analysis.

4.1 Explanations about Explanations

Explanations (specializing (A)TMS [Doyle 1979]) have been initially introduced to improve backtracking-based algorithms [Ginsberg 1993]. However, they have been recently used for many other purposes [Jussien 2003] including debugging and analysis of the behavior of constraint solvers.

4.1.1 Definition

An explanation contains enough information to justify a decision (throwing a contradiction, reducing a domain, etc.), it is composed of the constraints and the choices made during the search which are sufficient to justify such an inference.

An explanation of an inference (X) consists of a subset of original constraints ($C' \subset C$) and a set of instantiation constraints (choices made during the search: $d_1, d_2, \ldots, d_k$) such that:

$$C' \wedge d_1 \wedge \ldots \wedge d_n \Rightarrow X$$

$C' \wedge d_1 \wedge \ldots \wedge d_n$ is called an explanation. Roughly speaking, an explanation is a set of constraints that explain decisions taken by the solver, such as the withdrawal of a value from the domain of a variable.

An explanation e_1 is said to be more precise than explanation e_2 if and only if $e_1 \subset e_2$. The more precise an explanation, the more useful it is.

4.1.2 The explanation network

Usually, explanations may be used for user interaction [Freuder et al. 2001; Ouis et al. 2003], dynamic constraint handling [Debruyne et al. 2003], or even improvement of search algorithms [Ginsberg 1993; Jussien et al. 2000, Jussien and Lhomme 2003]. However, explanations, as we defined them, provide insightful information about the constraint solver dynamics.

Considering explanations or, more precisely, the pair of networks they induce between constraints and between variables, proves to be fruitful for understanding the behavior of the constraint solver on a given problem. Indeed, constraints appearing in an

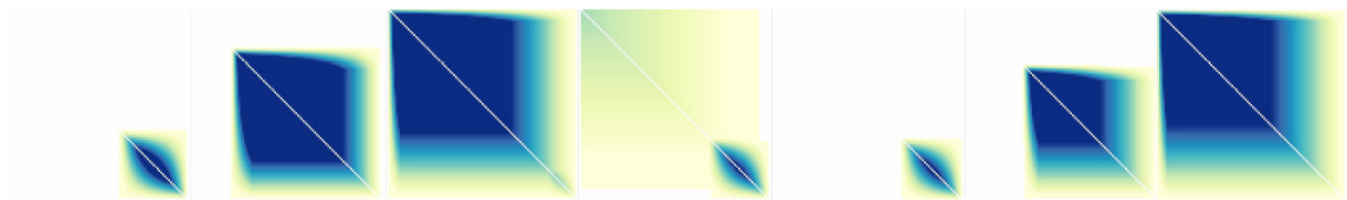


Figure 4: Visualizing the dynamics of propagation in our sorted problem. From left to right: a first slice, in the middle of the first phase, at in the end of the first phase, changing phases, at the beginning of the first phase, in the middle of the second phase, and the last slice.

explanation are cooperating (event if they do not share any variable) to remove some values and are, therefore, related. Similarly, variables related to constraints that appear in an explanation are related during the resolution.

In this paper, we show that it is interesting to visualize, and analyze the relationships (and their dynamics) between constraints and between variables. Note that we are interested in the use of the explanation network as a representation of the solver dynamics. Therefore, a priori [Amilhastre et al. 2002] or a posteriori [Junker 2001] explanation computation techniques would be of no use here.

Recently, resolution algorithms evolve towards adaptive schemas that try and integrate the particularities of a given instance in their behavior. In all cases, the detection of the underlying structure of problems may significantly accelerate the search. In the present work, visualization tools help in classifying the information susceptible of being integrated in such an adaptive behavior, and in revealing the underlying structure of problems as well. In the following cases of study, we are able to exhibit various structures. In the first case, the search activity creates dynamic structures that occlude the intrinsic structure of the problem, and drives the solver into numerous costly enumerations. In the second case, very specific static structures are exhibited with our visualization tools; these structures may help define an ad hoc resolution strategy. In the last case, we show how the visualization can uncover activity patterns during constraint propagation in very dense graphs, and help "feel" the behavior of the solver.

4.2 Investigations on Problem Structure

Due to the combinatorial nature of the problems tackled by CP, the efficiency of any given heuristic is very much related to its ability to discard those areas of the search tree containing no solutions. The quicker the sterile branches are pruned, the faster the solver can concentrate on branches worthy of investigation. In this context, understanding the structure of the problem at hand and identifying its most influential variables may dramatically reduce the resolution time as we shall see in the sequel.

4.2.1 Hypotheses

Recent works [Refalo 2004] proved that the propagation phase provides valuable information that may contribute towards crafting generic search heuristics. One such information is the average reduction of the search space due to a given variable. In the present experiment, we are interested in understanding:

- the direct impact of a variable on the reduction of the search space,
- the indirect impact of a variable, i.e., the reduction of the search space caused by the variable even a long time after its instantiation occurs,
- the areas of the problem which are influenced by a variable,
- the relationships between variables, if any.

Such information relies on the explanations computed by the solver during the resolution process, as they account for the logic underlying the chain of consecutive inferences made by the solver during the propagation. The influence of a variable v_i on another variable v_j may be seen as the number of times where the instantiation of v_i justifies the reduction of v_j. The influence of a variable v_i on the entire problem can then be seen as the sum of influences of v_i on every variable of the problem.

In the following examples, we shall see how visualization tools help understand the structure of the problem at hand, and determine the role played by its influential variables. Such analyses may inspire some improvements on the search strategy of the solver.

4.2.2 Experiment with random structured problems

4.2.2.1 Study of a hard/peculiar instance

In this first experiment, we crafted a binary random problem where we induced a structure by increasing the hardness of some constraints. This results in the emergence of three sets of 10 variables with strong internal relations – i.e. relations between the variables within each set – and loose external relations – i.e. relation with variables in other sets. In this perspective, the variables of a given set are strongly interrelated, and are expected to be involved in the same inferences made by the solver.

This problem is particularly interesting because it proved to be harder than we could initially expect on classical heuristics such as MinDom (the variable with the current domain of minimum size is selected to branch) as well as random instantiation heuristics. This problem raised the following questions.

1. Is it possible to detect the structure induced on the problem, i.e., the aforementioned sets of variables?
2. Can we account for the poor performance of the MinDom heuristic? Is it related to this peculiar instance or is it due to the heuristic per se?
3. How can the information collected from the propagation phase help answer these questions?

4.2.2.2 Results

In Figure 5, we display the graph of 30 variables involved in the problem using a matrix-based representation. The variables of the problem are represented by the rows and columns of the matrix. The cell at the intersection of column i and row j corresponds to the impact of the variable v_i on the variable v_j. The stronger the impact, the heavier the edge, and the darker the cell. The matrix is ordered according to the order of declaration of the variables by the solver, which happens to be a relevant order with regard to CP graphs. A singleton consistency algorithm is applied (every value of every variable is propagated) to ensure that the impact of variables is initialized homogenously. Although the graph is almost entirely connected, the matrix-based visualization makes it possible to see very clearly the structure of the problem, i.e., the three sets of variables having strong internal links.

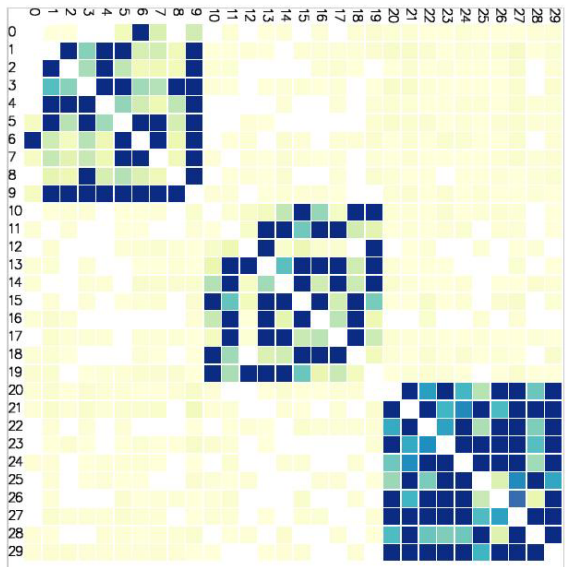


Figure 5: The representation of the graph of variables at the end of the initialization phase.

Figure 6 shows the graph of variables after running the MinDom heuristic. We interrupted the execution after two minutes. Compared to Figure 5, we note that the structure of the problem is no longer visible. The blue area at the bottom left corner means that the variables in the first two sets have a strong influence on the variables belonging to the third set. This can be accounted for by the fact that poor decisions taken early concerning the variables of the first sets lead the solver into numerous try-and-fail steps on the variables of the third set.

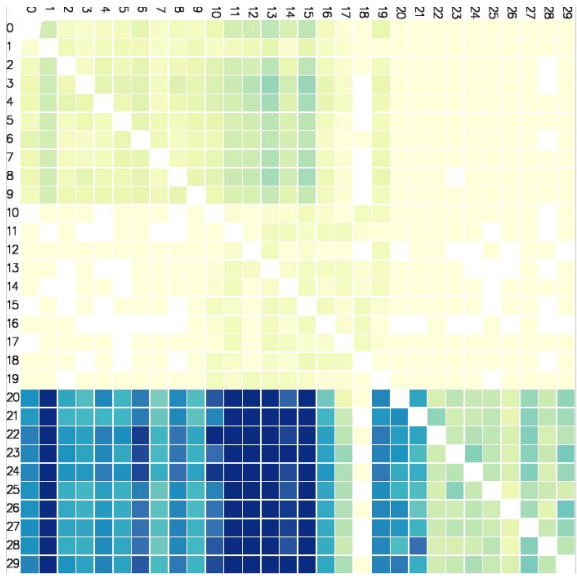


Figure 6: The graph of variables after two minutes of computation using the MinDom heuristic.

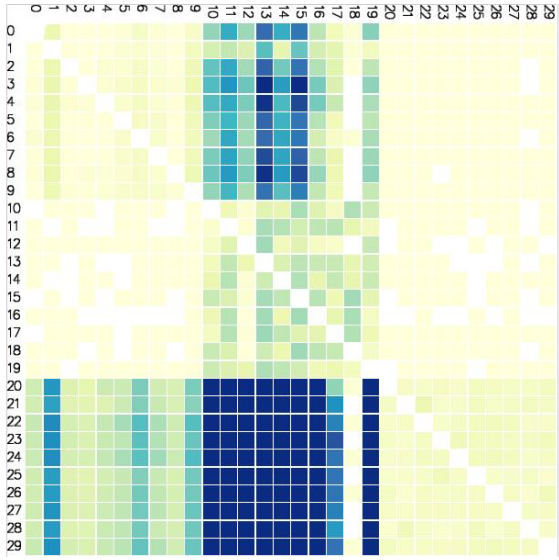


Figure 7: A normalized representation of the graph of variables using the MinDom heuristic.

On Figure 7, we display a normalized representation of the same graph where the influence of a decision taken by the solver is divided by the number of times this decision occurred during the resolution. By doing so, we aim at refining the previous analysis by distinguishing two types of decisions: those having a great influence and are, therefore, repeated frequently, and those having a great influence because the solver is stuck in some inconsistent branch of the search tree. In this way, we can isolate early poor decisions that seem to involve the second set of variables by taking into account the age of decisions as seen on Figure 8.

Figure 8 reflects the activity within the graph of variables after two minutes of computation using the MinDom heuristic. In the assessment of activity, the effect of old decisions is gradually discarded. As expected, it appears that the solver keeps going back and forth between the first and the third set of variables, with very negligible involvement of the second set. This must be related to early poor decisions taken on the variables of the second set. It is also worthwhile to note that the structure of the problem is made visible once more.

In order to further confirm this interpretation, we adapted the search heuristic so that it takes into account a measure of influence of variables during the resolution, and revokes immediately decisions whose influence increase outstandingly. As a consequence, the problem was solved almost instantaneously, and the structure of the problem is exhibited as in Figure 9.

4.2.3 Conclusion

The gist of the previous results is that poor decisions taken early on the variables may trap the solver in a combinatorial, expensive enumeration of values in the variable domains of the problem. An early detection of the critical variables – those having a great influence on the problem – allows the constraint programmer to adapt and direct his heuristic so that it avoids falling into time-consuming enumerations.

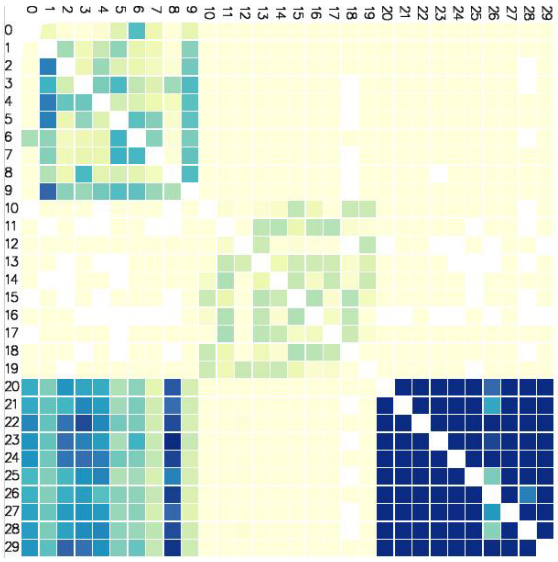


Figure 8: The graph of variables taking into account the age of decisions.

The visualization of the graph of variables plays an essential role in helping the programmer see the underlying structure of the problem, detect the critical variables, and ultimately adapt his heuristics to avoid poor decisions. To the best of our knowledge, the matrix-based representation of graph is definitely the only metaphor that makes the display of a very dense graph possible without overlapping. Moreover, color-coding the edges of such graphs reflects their activity, and makes it possible to grasp the distribution of activity throughout the graph in a pre-attentive fashion. In contrast, the visual clutter induced in node-link layouts makes them of no practical use for monitoring purposes.

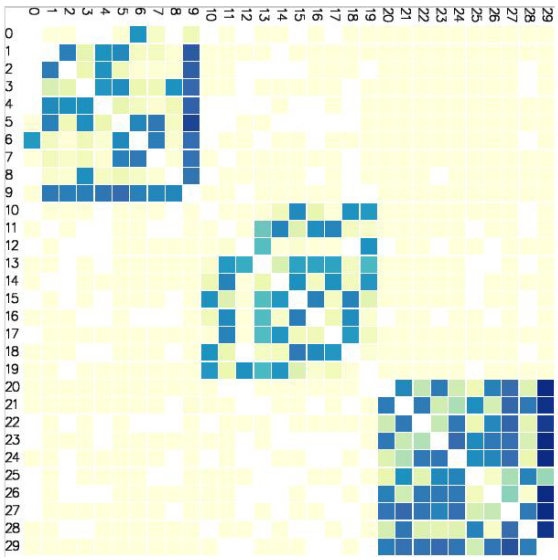


Figure 9: The graph of variables using an adaptive search heuristic.

4.3 Investigations on Incremental Resolution of Problems

In some problems, the inherent structure of a given instance may provide valuable clues in view of accelerating the search strategy. In the previous case, the structure of the problem was occulted, and could only be revealed through the propagation phase. In the present case, we are interested in the use of visualization tools as a means of discovering the underlying structure of hard academic problems. The gained insights can be used to enhance the search strategy.

4.3.1 Graph Coloring Problems

We have examined hard graph coloring problems included in the DIMACS 2002 Challenge[1]. Graph coloring problems consist in using the minimal number of colors so that a color can be attributed to every vertex in the graph, and every two adjacent vertices should have different colors. The presence of cliques in the graph provides an obvious lower bound of the number of colors, and helps accelerate the search. Hence, the detection of maximal cliques is integrated in the best coloring algorithms. Some hard graphs are called "triangle-free" because their maximal cliques contain no more than two vertices. We have studied two categories of triangle-free graphs: Mycielski graphs (graphs based on Mycielski transformation) and *k*-insertion graphs.

4.3.1.1 Mycielski graphs

Figure 10 and Figure 11 reveal the static structure of Mycielski 4 and Mycielski 5 instances obtained as described in the previous section, after the propagation phase in the root node and after the instantiation of the most connected variables (which reduced the symmetries in the problem). The use of a matrix-based representation reveals its peculiar structure, and suggests the following directions for its resolution.

- Examining the relations between the variables of Mycielski 4, we realize that variables 11 to 21 are not related (the bottom right quadrant is blank). Consequently, the coloring of variables 1 to 10 can be immediately extended to (or proved inconsistent with) variables 11 to 21. Such a configuration suggests the decomposition of the problem into a master problem comprising variables 1 to 10 with a combinatory role, and a secondary problem that includes the remaining variables.

- Examining Mycielski 5, we can see a recurrent structure common to all instances of the problem such that the instance of size n can be defined from the instance of size $n-1$. This structure suggests an incremental resolution strategy (kind of Russian dolls search) where the resolution of a subset of the problem may help in setting a new bound that would accelerate the resolution of the following problem. Thus, solving a series of imbricated problems may prove far more effective than tackling one big problem from scratch.

[1] http://mat.gsia.cmu.edu/COLORING02/

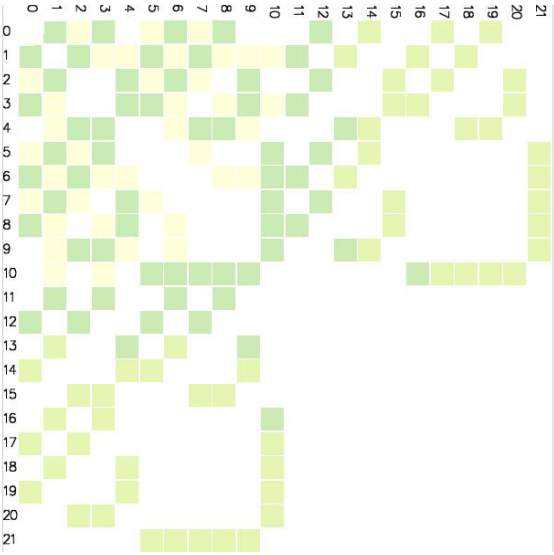


Figure 10: The graph of variables of Mycielski 4.

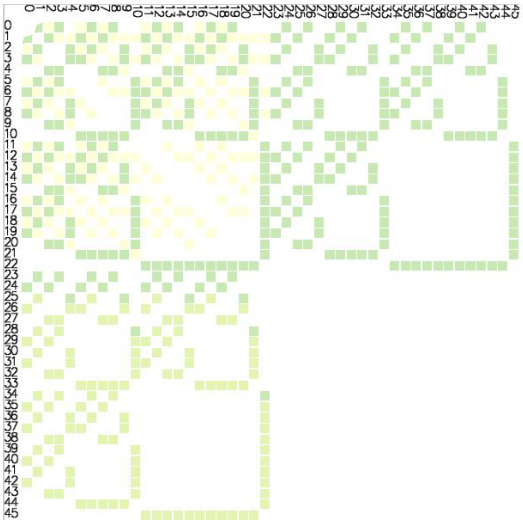


Figure 11: The graph of variables of Mycielski 5.

4.3.1.2 The Contribution of Visualization Tools

In the present experiment, the matrix-based representation provides a clear overview of the graphs at hand. Also, it effectively displays the symmetries present in the problem. Of course, this can be achieved by advanced node-link layout packages when dealing with sparse graphs. An interesting advantage of the matrix-based metaphor lies in its "duality", that is, it displays the "missing links" as well as the established ones. One glance allows the user to see that variables 11 to 21 are not connected at all, whereas this may be very difficult to detect with a node-link representation.

Figure 12: The graph of variables of the 1-insertion problem, instance of size 4.

4.3.1.3 Future developments

In further developments of this case, we may craft a special strategy that benefits from the previous observations, and monitor the way the activity spreads across the graph. We can also tackle other "triangle-free" graphs having a similar incremental nature as the *k*-insertion problem represented in Figure 12. This is a coloring problem consisting in the insertion of *k* vertices into a Mycielski graph without changing its density. As we said earlier, the user is tempted to begin with the resolution of the elementary pattern, and subsequently use the information collected during this stage to accelerate the resolution of the rest of the problem.

5 Conclusion

In this article, we have described a novel use of matrix-based graph representations to visualize dynamic activity graphs generated by constraint-based programs. We have shown how this representation can be used to understand the dynamic behavior of constraint programs. We have applied the visualization to several kinds of graphs such as constraint graphs (constraint-variable, variable-variable, constraint-constraint) and explanation graphs.

These visualization tools have been very effective at opening the black box of constraint-based programs. We have already gained novel insights into the actual behavior of complex solver strategies on hard problems, and we are currently exploring the full potential of our tools. The variety of the use cases discussed in this work suggests strongly that our tools are extendible to many tasks that occur in the life-cycle of constraint-based programs. A controlled user experiment is yet to be designed and carried out in order to measure the effectiveness of such tools against different user populations e.g. students, confirmed constraint programmers, and expert users, in connection with various tasks or requirements. By all means, such an evaluation is a very challenging endeavor and must be carried out on a limited set of tasks and a well identified population for it to be conclusive.

We are currently applying the visualizations for pedagogical purposes, and are investigating their use to further improve

heuristics conceived for challenging combinatorial optimization problems.

Acknowledgments

This work has partly been funded by the French RNTL project OADymPPaC.

More material is available at http://tra4cp.sourceforge.net.

References

Abello, J. and Ham, F. v. *Matrix Zoom: A Visual Interface to Semi-External Graphs.* In Proceedings of the 10th IEEE Symposium on Information Visualization (InfoVis'04), Austin, TX, Oct 2004. IEEE Press. pp. 183-190.

Amilhastre, J., Fargier, H., and Marquis, P. 2002. *Consistency restoration and explanations in dynamic csps - application to configuration.* Artificial Intelligence 135(2002):199–234.

Becker, R.A., Eick, S.G. and Wills, G.J. (1995) *Visualizing network data.* IEEE Transaction on Visualizations and Graphics, 1 (1). 16-28.

Bertin, J. 1983. Semiology of Graphics: *Diagrams, Networks, Maps.* University of Wisconsin Press.

Brayshaw, M. & Eisenstadt, M. (1991). *A Practical Tracer for Prolog.* International Journal of Man-Machine Studies, 42, 597-631.

Byrd, L. 1980. *Understanding the control flow of Prolog programs.* Proceedings of the Logic Programming Workshop. Debrecen, Hungary.

Card, S.; Mackinlay, J.; and Shneiderman, B. 1999. Readings in Information Visualization: Using Vision to Think. San Francisco, USA: Morgan Kaufmann. chapter Dynamic Queries, 235–261.

Carpendale, M. S. T., and Montagnese, C. 2001. *A framework for unifying presentation space.* In Proceedings of the 14th annual ACM symposium on User interface software and technology (UIST'01), 61–70.

Debruyne, R.; Ferrand, G., Jussien, N., Lesaint, W., Ouis, S., and Tessier, A. 2003. *Correctness of constraint retraction algorithms.* In FLAIRS'03: Sixteenth international Florida Artificial Intelligence Research Society conference, 172–176. St. Augustine, Florida, USA: AAAI press.

Doyle, J. 1979. *A truth maintenance system.* Artificial Intelligence 12:231–272.

Eisenstadt, M. (1984). *A Powerful Prolog Trace Package.* Sixth European Conference on Artificial Intelligence, Pisa, Italy.

Eisenstadt, M. & Brayshaw, M. (1990). *A fine grained account of Prolog execution for teaching and debugging.* Instructional Science, 19(4/5), 407-436.

Eisenstadt, M., and Brayshaw, M. *The truth about Prolog execution.* In J.Stasko, J.Domingue, M.H.Brown, and B.A.Price (Eds.) Software visualization: programming as a multimedia experience. Cambridge, MA: MIT Press, 1998.

Fages, F. 2002. *CLPGUI: a generic graphical user interface for constraint logic programming over finite domains.* In ICLP'02 12th Workshop on Logic Programming Environments (WLPE'02).

Fekete, J.-D., and Plaisant, C. 1999. *Excentric labeling: Dynamic neighborhood labeling for data visualization.* In Proceedings of the International Conference on Human Factors in Computing Systems (CHI 99), 512–519. ACM.

Fekete, J.-D., and Plaisant, C. 2002. *Interactive Information Visualization of a Million Items.* In Proceedings of IEEE Symposium on Information Visualization 2002 (InfoVis 2002), Boston, USA, Octobre 2002. IEEE Press, pp. 117-124.

Freuder, E. C., Likitvivatanavong, C., and Wallace, R. J. 2001. *Deriving explanations and implications for constraint satisfaction problems.* In Principles and Practice of Constraint Programming (CP 2001), LNCS, 585–589.

Ghoniem, M., Jussien, N. and Fekete, J.-D., VISEXP: Visualizing Constraint Solver Dynamics Using Explanations. In FLAIRS'04: Seventeenth international Florida Artificial Intelligence Research Society conference, (Miami Beach, FL, May 2004), AAAI press.

Ghoniem, M., Fekete, J.-D. and Castagliola, P. *A Comparison of the Readability of Graphs Using Node-Link and Matrix-Based Representations*. In Proceedings of the 10th IEEE Symposium on Information Visualization (InfoVis'04), Austin, TX, Oct 2004. IEEE Press. pp. 17-24.

Ginsberg, M. 1993. *Dynamic backtracking.* Journal of Artificial Intelligence Research 1:25–46.

Ham, F.v., *Using Multilevel Call Matrices in Large Software Projects*. In Proc. IEEE Symp. Information Visualization 2003, (Seattle, WA, USA, 2003), IEEE Press, 227-232.

Junker, U. 2001. *QUICKXPLAIN: Conflict detection for arbitrary constraint propagation algorithms.* In IJCAI'01 Workshop on Modelling and Solving problems with constraints.

Jussien, N., Debruyne, R., and Boizumault, P. 2000. Maintaining arc-consistency within dynamic backtracking. In Principles and Practice of Constraint Programming (CP 2000), LNCS 1894, 249–261. Singapore: Springer-Verlag.

Jussien, N., and Lhomme, O. 2002. *Local search with constraint propagation and conflict-based heuristics.* Artificial Intelligence 139(1):21–45.

Jussien, N. 2003. The versatility of using explanations within constraint programming. Research Report 03-04-INFO, Ecole des Mines de Nantes, Nantes, France.

Lalanne, D. and Pu, P. *Interactive Problem Solving Via Algorithm Visualization*, IEEE Symposium on Information Visualization 2000 (InfoVis 2000), October 9-10, 2000, Salt Lake City, Utah.

Mulholland, P. (1995). *Prolog without tears: An evaluation of the effectiveness of a non Byrd Box model for students.* 7th Annual Workshop 4-6 January 1995. Collected Papers of the Psychology of Programming Interest Group, University of Edinburgh.

Ouis, S.; Jussien, N., and Boizumault, P. 2003. *k-relevant explanations for constraint programming.* In FLAIRS'03: Sixteenth international Florida Artificial Intelligence Research Society conference, 192–196. St. Augustine, Florida, USA: AAAI press.

Stasko, J., Catrambone, R., Guzdial, M. and McDonald, K.. *An evaluation of space-filling information visualizations for depicting hierarchical structures.* International Journal of Human-Computer Studies, 53(5): 663–694, November 1998.

Taylor, C., du Boulay, B., & Patel, M. (1991). *Outline proposal for a Prolog 'Textual Tree Tracer' (TTT).* CSRP No. 177, University of Sussex.

Yang, J., Ward, M. O. and Rundensteiner, E. A., *InterRing: An Interactive Tool for Visually Navigating and Manipulating Hierarchical Structures*. In *Proceedings of IEEE Symposium on Information Visualization (Infovis 2002)*, IEEE Computer Soc. Press, Boston, Massachusetts, 28-29 October 2002, pp. 77-84.

Deransart, P., Hermenegildo, M. and J. Małuszyński, editors. *Analysis and Visualization Tools for Constraint Programming*. Number 1870 in LNCS. Springer Verlag, 2000. European Project (1997-2000) http://discipl.inria.fr.

Jussien, N. and Barichard, V.. The PaLM system: explanation-based constraint programming. In *Proceedings of TRICS: Techniques foR Implementing Constraint programming Systems, a post-conference workshop of CP 2000*, pages 118–133, Singapore, September 2000.

Laburthe, F.. Choco: implementing a cp kernel. In *Proceedings of TRICS: Techniques foR Implementing Constraint programming Systems, a post-conference workshop of CP 2000*, Singapore, September 2000.

Refalo, P. Impact-Based Search Strategies for Constraint Programming, *Proceedings of the 10th intern. Conference on the Principles and Practice of Constraint Programming (CP 2004)*, LNCS, Springer-Verlag, 2004, pp. 557-571.