Saisie de texte interactive sous Mentor

V. Migot

▶ **To cite this version:**

V. Migot. Saisie de texte interactive sous Mentor. RT-0063, INRIA. 1985, pp.19. inria-00070096

HAL Id: inria-00070096
https://inria.hal.science/inria-00070096

Submitted on 19 May 2006

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.

CENTRE DE ROCQUENCOURT

Rapports Techniques

N° 63

SAISIE DE TEXTE INTERACTIVE SOUS MENTOR

Valérie MIGOT

Décembre 1985

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél. : (1) 39 63 55 11

Saisie de texte interactive sous Mentor

Valérie Migot

INRIA
Domaine de Voluceau, Rocquencourt
BP 105, 78153 Le Chesnay Cedex, France

Résumé

Mentor est un système permettant la manipulation de documents structurés: programmes, spécifications, rapports techniques. Le noyau de Mentor est un éditeur syntaxique dans lequel les objets manipulés sont représentés par des arbres de syntaxe abstraite. Cet article décrit un nouveau composant du système: la saisie de texte guidée par menus. Des menus et des messages d'aide pour l'utilisateur sont automatiquement créés à partir de la spécification du langage manipulé.

Abstract

Mentor is a general system for the manipulation of structured documents such as programs, specifications or technical reports. The kernel of Mentor is a syntax directed editor in which objects are represented by abstract syntax trees. This paper describes a new component of the system: the menu driven input mode. Menus together with associated user assistance for interactive construction of new fragments of documents are mechanically generated from a high-level syntactic specification of the manipulated language.

PAPIER RECUPERE ET RECYCLE

1. Introduction

Mentor est un système qui permet la manipulation de documents structurés ([Don 83] [Don 84a] [Don 84b]). Cet article décrit un nouveau composant du système permettant la saisie de texte guidée par menus. Ce processeur aide l'utilisateur dans la saisie d'un document ou pour effectuer des modifications dans un document déjà existant. Pour cela, la méthode pour saisir le texte est analogue à celle utilisée dans des systèmes tels que Gandalf [Fei 81], le Cornell Program Synthesizer [Tei 81]. Des noeuds de la syntaxe abstraite sont construits explicitement à l'aide de menus et de schémas. La méthode utilisée dans Mentor a l'avantage d'être plus souple: à tout moment, l'utilisateur peut passer du mode de saisie guidée par menus au mode de saisie standard de Mentor. D'autre part, ce processeur est indépendant du langage manipulé. Il est particulièrement utile pour construire des documents multi-langages: le changement de langage est fait automatiquement et l'utilisateur ne mélange pas les différentes syntaxes abstraites. Ce processeur a été rajouté à Mentor pour faciliter la saisie de texte et donc améliorer l'interface utilisateur. Cette méthode évite de saisir les mots clés du langage et facilite l'apprentissage d'un nouveau langage: il est inutile de se souvenir de toutes les structures syntaxiques du langage car elles sont proposées par le menu au bon moment et des messages d'aide indiquent à quoi elles correspondent.

Les menus proposés, ainsi que les messages d'aide affichés simultanément, sont créés à partir de la spécification en Métal du langage manipulé. La saisie guidée par menus peut donc être effectuée avec n'importe quel langage connu par Mentor; cependant les messages d'aide doivent être contenus explicitement dans la spécification en Métal, sous forme d'annotations.

Cet article présente rapidement les notions utiles pour la compréhension du système de saisie par menus sous Mentor. Il décrit ensuite les fonctionnalités, puis l'utilisation du système; son implémentation est décrite en annexe.

2. Mentor

2.1. Syntaxe abstraite

Un document est représenté sous Mentor par un arbre étiqueté. Pour chaque langage, une grammaire d'arbres, appelée syntaxe abstraite définit les arbres corrects du langage. La définition de la syntaxe abstraite d'un langage repose sur deux notions: les opérateurs et les phyla. Les opérateurs étiquettent les noeuds des arbres de syntaxe abstraite et définissent les sous-arbres légaux de ces noeuds. Pour chaque opérande d'un opérateur donné, la syntaxe abstraite définit l'ensemble des opérateurs possibles pour les sous-arbres correspondant. Un tel ensemble est appelé un phylum. Il y a deux sortes d'opérateurs: les opérateurs d'arité fixe et les opérateurs de listes. Le nombre d'opérandes d'un opérateur d'arité fixe est fixé (il est égal à zéro pour les atomes), et un phylum est spécifié pour chaque opérande. Un opérateur de listes peut avoir un nombre arbitraire d'opérandes appartenant tous au même phylum.

Voici un exemple de définitions d'opérateurs et de phyla extrait de la syntaxe abstraite de Pascal:

Opérateurs d'arité fixe:

```
nil       -->                (* opérateur atomique *)
neq       --> EXP       EXP
plus      --> EXP       EXP
assign    --> VARIABLE  EXP
call      --> IDENT     EXP_LIST
```

```
while       -->  EXP        STAT
```

Opérateurs de listes:

```
exp_list    -->  EXP        ...
stat_list   -->  STAT       ...
```

Phyla:

```
VARIABLE  :: ident     meta      unref     index     dot

EXP       :: ident     meta      intcst    alfacst   charcst
             nil       hexcst    realcst   setof     not
             uplus     uminus    unref     hexa      eql
             lss       gtr       neq       leq       geq
             in        intdiv    mod       div       mult
             plus      minus     or        and       index
             dot       format    call

STAT      :: meta      goto      repeat    assign
             call      case      while     with
             labstat   stat_list for       if

EXP_LIST  ::  exp_list  meta
```

2.2. *Schéma*

Un schéma est un arbre dans lequel certains sous-arbres ont été remplacés par des noeuds atomiques particuliers appelés méta-variables. Sous Mentor, les méta-variables sont repérées par des identificateurs préfixés par le caractère dollar. Par exemple:

```
if X=0 then $MV1 else $MV2
```

est un schéma Pascal dans lequel les méta-variables $MV1 et $MV2 remplacent des instructions Pascal.

A chaque opérateur de la syntaxe abstraite correspond un schéma prédéfini: l'arbre le représentant est étiqueté par cet opérateur. Si l'arité de l'opérateur est zéro, l'arbre est réduit à une méta-variable. Sinon chacun de ses sous-arbres est remplacé par une méta-variable dont le nom est construit à partir de celui du phylum du sous-arbre. Par exemple, le schéma prédéfini de l'opérateur while dans la syntaxe abstraite de Pascal est:

```
while $EXP1 do $STAT1.
```

2.3. *Annotations et mécanisme multi-langage dans Mentor*

Un document est très souvent annoté avec diverses informations n'interférant pas avec la structure générale du document. Pour un programme, il peut s'agir de commentaires, d'anciennes versions du code, de spécifications. Pour un rapport technique, il peut s'agir de notes en bas de page, de références bibliographiques.

Mentor permet de manipuler cette notion d'annotations et ainsi de composer un document contenant divers formalismes. Un document multi-langage est représenté par un arbre dont certains noeuds sont annotés par des arbres appartenant à des langages différents. Les annotations sont organisées suivant leurs caractéristiques. Formellement, on définit un décor d'annotations en lui donnant un nom, en spécifiant le langage auquel l'annotation appartient et le langage des arbres qu'elle peut annoter. A n'importe quel noeud d'un arbre, on peut accrocher des annotations appartenant à des décors différents. Plusieurs langages peuvent donc être manipulés simultanément dans la même session et peuvent coexister dans le même arbre.

2.4. Métal: un langage de spécification modulaire de haut niveau

Mentor est indépendant du langage qu'il manipule. Pour ajouter un nouveau langage sous Mentor, il faut décrire ce langage dans un langage de spécification appelé Métal [Mel 82]. Un programme Métal décrit différents types d'informations nécessaires pour créer l'environnement de programmation associé à un langage particulier: la syntaxe abstraite du langage, la syntaxe concrète, les fonctions de construction des arbres, les spécifications de décompilation. La compilation d'un programme Métal produit les tables utiles au système pour manipuler des objets appartenant au langage défini par le programme Métal. Les tables sont interprétées par des processeurs spécialisés: analyseur-constructeur, décompilateur, primitives de manipulation d'arbres.

La structure de Métal et le mécanisme d'annotation permettent une définition modulaire des différents types d'informations nécessaires pour décrire un langage. Chaque nouveau type d'information est représenté par un décor d'annotations. Cette structure modulaire de Métal se reflète dans la structure de Mentor. A un nouveau composant de Métal correspond un nouveau processeur du système. La structure d'arbre est utilisée par plusieurs processeurs indépendants qui constituent avec le noyau du système un environnement de programmation. Un de ces composants est l'interface utilisateur. Elle inclue en particulier deux processeurs qui vont être rapidement décrits: le traitement des messages et la saisie du texte.

2.5. Traitement des messages

Dans Mentor, tous les dialogues entre le système et l'utilisateur (messages d'erreurs, diagnostics, questions) sont effectués par un processeur spécialisé. Un message est un fragment de texte qui peut contenir des paramètres et des directives de formatage et qui est affiché dans une fenêtre qui dépend du type de ce message (diagnostic, question, information, ..).

Les messages sont regroupés dans des fichiers particuliers, suivant leurs fonctionnalités. Chaque message est caractérisé par le nom du fichier dans lequel il se trouve et un numéro d'ordre dans ce fichier.

2.6. Saisie de texte

Dans ses premières versions, Mentor apportait peu d'aide pour la saisie d'un nouveau document ou d'un fragment de texte. L'utilisateur disposait d'un éditeur textuel simplifié qui lui permettait de saisir son texte. Le texte était analysé ligne par ligne. Si le texte contenait des erreurs, certains fragments pouvaient être perdus et devaient alors être resaisis correctement.

Plus récemment, un nouveau mode de saisie a été introduit dans Mentor: pour saisir un fragment de texte on appelle un éditeur textuel externe (Emacs, Ed). Le texte saisi par l'utilisateur est alors conservé dans un fichier temporaire et il peut être réédité en cas d'erreur [Mel 85b]. Ces deux méthodes permettent des saisies très rapides par des utilisateurs expérimentés mais n'offrent que peu d'aide aux débutants.

Le mode de saisie guidée par menus, décrit dans la suite de ce papier, vient combler ce vide en offrant une aide efficace aux utilisateurs débutants. En outre, elle a un caractère pédagogique évident pour l'apprentissage d'un nouveau langage. La construction explicite de noeuds de syntaxe abstraite à l'aide de menus et de schémas prédéfinis est un bon moyen pour guider l'utilisateur

dans la construction de son programme. A tout moment, il est aidé: le menu lui indique les constructions correctes pour la position courante. Ce mode de saisie est utile pour des utilisateurs qui ne sont pas familiers avec la notion de syntaxe abstraite, ou qui connaissent mal la syntaxe d'un langage. Il est également apprécié des utilisateurs expérimentés qui manipulent plusieurs langages à la fois. Il évite de plus d'avoir à saisir tous les mots clés du langage.

3. Les fonctionnalités du mode de saisie guidée par menus

Le mode de saisie guidée par des menus est un nouveau composant de Mentor qui assiste l'utilisateur durant la saisie de texte. La méthode suivie s'inspire de celle utilisée par des systèmes tels que Gandalf et le Cornell Program Synthesizer. Un menu fournit la liste des possibilités parmi lesquelles l'utilisateur fait son choix [Pit 84]. Dans le contexte de Mentor, les menus proposent les schémas du langage qu'il est légal d'insérer à l'endroit courant pour construire des programmes syntaxiquement corrects.

L'originalité de la technique utilisée réside dans les faits suivants: à tout moment, l'utilisateur peut quitter ce mode (temporairement ou définitivement) pour revenir au mode standard de saisie sous Mentor et appeler directement l'analyseur. Le mode guidé par menus impose a priori une construction en ordre préfixe. Cependant l'utilisateur peut quitter temporairement le mode guidé, et modifier un autre sous-arbre en rappelant le mode guidé à cet endroit. Quand il a fini de saisir ce sous- arbre, il peut revenir à l'endroit précédent et reprendre la saisie. Il est donc possible de rappeler récursivement le mode guidé. D'autre part, ce processeur est multi-langage: on peut saisir, en étant guidé, des arbres appartenant à des langages différents.

3.1. Création d'un document

Pour créer un nouveau document (un programme par exemple), l'utilisateur doit choisir l'opérateur racine de l'arbre à construire, parmi la liste des opérateurs du langage qui lui sont

proposés. Le schéma prédéfini de l'opérateur qu'il a choisi est alors créé et affiché sur l'écran. Cet arbre initial est ensuite parcouru en ordre préfixe: pour chaque méta-variable rencontrée, un menu est proposé pour construire le sous-arbre correspondant. Le même processus (choix dans le menu, construction du schéma prédéfini correspondant, parcours de chaque sous-arbre de gauche à droite) se répète jusqu'aux feuilles de l'arbre.

Pour créer un fragment de programme Ada, l'utilisateur peut par exemple choisir l'opérateur 'loop' dans la liste des opérateurs du langage Ada. Le schéma prédéfini correspondant est alors affiché:

```
$ITERATION1 loop
   $L_STM1
end loop;
```

Le système va guider l'utilisateur pour compléter le premier sous-arbre représenté par la méta-variable $ITERATION1. Le menu propose alors la liste des opérateurs autorisés à cet endroit: void, while, for, reverse. Si l'utilisateur choisit de construire une boucle while, le fragment de programme devient:

```
while $EXP1 loop
   $L_STM1
end loop;
```

Le système guide ensuite l'utilisateur pour compléter le premier sous-arbre représenté par $EXP1 en lui proposant la liste des opérateurs d'expression en Ada. Si l'utilisateur choisit l'opérateur "appel de fonction" (function_call), l'affichage devient:

```
while $NAME1($L_PARAM_ASSOC1) loop
   $L_STM1
end loop;
```

Le système demande alors le nom de la fonction à mettre à la place de $NAME1, le système analyse la réponse et crée l'identificateur Foo:

```
while Foo($L_PARAM_ASSOC1) loop
   $L_STM1
end loop;
```

La liste de paramètres est ensuite remplie:

```
while Foo(Bar) loop
   $L_STM1
end loop;
```

Le premier fils de l'opérateur while est maintenant complet, la méta-variable $L_STM1 peut alors être remplacée.

3.2. Modification d'un document

Le mode guidé par menus est aussi adapté à toutes les modifications d'un programme et

celles-ci ne nécessitent plus aucune connaissance particulière de la structure de l'arbre représentant le programme. Lorsque ce mode est invoqué sur un fragment de programme existant, le premier menu demande si l'on veut remplacer tout le fragment de programme; dans ce cas, on est ramené à la situation décrite ci-dessus. Dans le cas contraire, l'arbre est parcouru en ordre préfixe; pour chaque sous-arbre, le menu demande s'il faut le modifier. Ce processus se répète jusqu'aux feuilles de l'arbre.

Lors de la traversée de l'arbre, si le sous-arbre courant est une méta-variable, le menu propose automatiquement la liste des opérateurs pouvant la remplacer. Si le sous-arbre courant est un élément de liste, le menu demande si l'on veut modifier le sous-arbre, ou insérer un nouvel élément dans la liste.

3.3. L'aide apportée par le système de menus

Les programmes saisis en mode guidé sont corrects par construction: c'est le système qui s'occupe de fournir tous les éléments de la syntaxe du langage. Seuls les atomes (identificateurs, entiers, ...) sont entrés directement par l'utilisateur. L'analyse de ces atomes peut parfois échouer en cas d'erreur de l'utilisateur, mais dans ce cas le système redemande la valeur jusqu'à obtenir une réponse correcte.

Tous les menus proposés sont accompagnés des indications nécessaires à leur bonne compréhension et de précisions sémantiques ou lexicales concernant l'opérateur choisi par l'utilisateur. Ces messages ont été créés automatiquement à partir de spécifications (cf annexe) et ils sont affichés dans une fenêtre réservée à cet usage. Ils sont affichés systématiquement dans le cas d'opérateur atomique; mais l'utilisateur peut questionner le système à tout moment pour avoir des informations complémentaires sur un opérateur particulier.

4. Utilisation du processeur de saisie guidée par menus

Pour saisir un texte en étant guidé par des menus sous Mentor, il faut appeler la commande Mentol *menu*. Quand cette commande est appelée sur un arbre encore indéfini, le premier menu affiche tous les opérateurs non atomiques du langage de l'arbre. Il est en effet inutile d'appeler ce mode pour construire un arbre réduit à une feuille. L'utilisateur répond en donnant les premières lettres distinctives du nom de l'opérateur qu'il a choisi pour commencer la construction de son arbre. Si sa réponse est incorrecte (nom d'opérateur inconnu ou réponse ne permettant pas de faire un choix unique), il doit refaire un nouveau choix.

Les réponses autorisées pour un menu quelconque sont:

.	pour sortir
?	pour appeler à l'aide
&	pour analyser directement le texte
!	pour sortir temporairement du menu ($ pour revenir)
un nom	contenu dans le menu

Au cours de l'insertion dans une liste:

M	pour inserer un nouvel élément dans la liste
.	pour terminer le traitement de la liste

A tout moment, l'utilisateur peut demander de l'aide en répondant '?' au menu. Il entre alors dans le système d'aide général de Mentor [Mel 85a]. Il peut également répondre '?' suivi d'un nom d'opérateur pour obtenir des informations concernant cet opérateur.

Il peut répondre '&' s'il veut utiliser directement l'analyseur-constructeur pour construire le sous-arbre courant sans aide. Dans ce cas, Mentor lui donne le nom du point d'entrée de l'analyseur correspondant à son sous-arbre comme s'il avait exécuté la commande Mentol 'c&'. Cette possibilité de quitter temporairement le mode guidé par menus est agréable pour saisir une expression ou pour un utilisateur expert du langage.

Une autre possibilité pour sortir temporairement du mode guidé est la touche '!'. Si l'utilisateur répond '!', il se trouve alors au niveau de commandes Mentol. Pour revenir au mode guidé, à l'endroit où il était avant de s'interrompre, il faut taper '$'.

La touche '!' est particulièrement utile dans une session multi-langage: on peut commencer la construction d'un programme, sortir en tapant '!', créer un décor d'annotations dans un autre langage, appeler le mode guidé sur cette nouvelle annotation (et donc dans le langage de cette annotation), lorsque l'annotation est complète, revenir à la construction du premier arbre en tapant '$'. Le passage d'un langage à l'autre est fait automatiquement par le système de menus.

La saisie en mode guidé s'arrête automatiquement lorsque le sous-arbre est complet. La touche '.' permet de quitter le mode guidé lorsqu'on le souhaite lors du traitement d'un sous-arbre d'arité fixe.

Nous décrivons maintenant le traitement des listes. Au début du traitement d'une liste, le menu est le suivant:

M(ore ...) . & ? !

dans lequel, à la place des trois points, apparaît le nom du phylum des éléments de la liste. Si la réponse est la lettre 'M', le premier élément de la liste est inséré. Si la réponse est '.', une liste vide est construite.

Durant le traitement de la liste, le point termine l'insertion, tandis que la touche 'M' permet d'insérer un autre élément dans la liste. Rappelons que la touche '&' permet d'analyser

directement un élément de la liste.

Quand la position courante est un atome, le système donne le nom de l'opérateur qui a été choisi et l'utilisateur doit donner sa valeur (identificateur, entier, ...). Pour terminer la saisie, deux cas se présentent suivant le contexte (c'est-à-dire suivant le phylum contenant l'opérateur):

- dans le cas où le phylum ne contient que des opérateurs atomiques, la réponse est directement interprêtée après le retour-chariot,

- dans les autres cas, l'analyseur attend un point en début de la ligne suivante pour terminer la saisie.

Un message d'aide indique à l'utilisateur dans quel cas il se trouve. D'autre part s'il y a une erreur lors de l'analyse du texte entré, le système de menus demande d'entrer une nouvelle valeur, et cela jusqu'à ce que la réponse soit analysée correctement.

5. Conclusion

La saisie de texte guidée par menus s'adresse plus particulièrement aux utilisateurs débutants, qu'ils soient débutants dans l'utilisation de Mentor ou débutants dans la connaissance d'un langage manipulé sous Mentor. Mais il peut être également apprécié des utilisateurs experts, pour la facilité apportée dans la construction de documents multi-langages, en particulier avec Mentor-Rapport. Il a de plus l'avantage de pouvoir être appelé récursivement et avec des langages différents.

Le mode de saisie guidée par menus est actuellement disponible sur n'importe quel terminal alpha-numérique, les saisies se faisant à l'aide du clavier. Une version de Mentor sur terminal bitmap est aujourd'hui opérationnelle et utilise le système de gestion d'écran ASH [Rei 84]. Elle fait appel à des menus déroulants pour le dialogue avec l'utilisateur.

Annexe

Implémentation du processeur de saisie guidée par menus

Le processeur de saisie utilise les tables de définition des langages, les primitives de manipulation et de parcours d'arbres de Mentor et une extension qui a été faite dans Métal, à l'aide d'annotations, pour spécifier les messages d'aide à l'utilisateur.

1. Construction des menus à partir de la syntaxe abstraite

Les menus sont calculés à partir de la syntaxe abstraite du langage de l'arbre et de la position courante dans cet arbre. Le menu propose les noms des opérateurs qui peuvent être mis à cet endroit, c'est-à-dire la liste des opérateurs appartenant au phylum du sous-arbre courant. Si on commence la saisie en mode guidé sur un arbre vide, le phylum courant n'est pas défini; dans ce cas, le menu propose la liste de tous les opérateurs non atomiques du langage.

Lorsque l'utilisateur a donné le nom de l'opérateur qu'il a choisi, le sous-arbre courant est remplacé par le schéma prédéfini de l'opérateur choisi. Le nouveau sous-arbre est alors parcouru en ordre préfixe, les méta-variables correspondant à chaque fils sont remplacées par le schéma choisi par l'utilisateur. Ce processus s'arrête de lui-même sur les feuilles de l'arbre.

2. Spécifications des messages d'aide

Les menus sont accompagnés d'indications sémantiques sur les opérateurs. Ces indications apparaissent dans une fenêtre particulière. Ces messages doivent être spécifiés dans le programme Métal qui décrit le langage car ils ne peuvent être entièrement créés mécaniquement à partir de la définition syntaxique du langage.

Pour décrire ces messages d'aide, un nouveau composant a été rajouté à la définition d'un langage en Métal. Les messages sont décrits à l'aide d'annotations accrochées dans le programme Métal. Un décor d'annotation appelé 'help' a été défini spécialement pour cela. Ces annotations doivent apparaitre en des endroits bien précis du programme Métal: une annotation appartenant au décor 'help' est accrochée sur un nom d'opérateur dans la définition d'un phylum. Cette annotation est le message décrivant les particularités de cet opérateur dans le phylum. Plus précisément, l'annotation est un commentaire Métal composé d'une ou plusieurs lignes de texte.

Les messages doivent donc être écrits à la main après avoir écrit le programme Métal définissant le nouveau langage. Les messages peuvent être obtenus en s'aidant du manuel du langage. Pour les opérateurs atomiques, il est utile qu'ils donnent une description lexicographique des valeurs autorisées pour ces atomes. Ils peuvent également être vides.

Afin d'éviter les duplications de messages identiques pour un opérateur appartenant à plusieurs phyla, les opérateurs apparaissant dans la définition du phylum every (le phylum every contient tous les opérateurs du langage) sont annotés par un message qui est indépendant du phylum. Dans les autres phyla, les opérateurs sont annotés seulement s'ils y présentent des particularités. Le message correspondant devra décrire uniquement ces particularités, sans reprendre le texte du message générique de cet opérateur dans le phylum every.

La compilation de ce composant de Métal (par la commande *metahelp*) crée un fichier de messages (qui doit s'appeler OPL.mess où L est le nom du langage tronqué à six caractères). A chaque message correspond un numéro qui dépend du phylum et de l'opérateur décrit.

Lors de la saisie en mode guidé, les messages d'aide affichés dépendent d'abord de l'opérateur choisi par l'utilisateur et ensuite du phylum courant contenant cet opérateur. Le système affiche toujours le message correspondant à l'opérateur dans le phylum every, puis

en complément, il affiche le message correspondant dans le phylum courant. Lorsque l'utilisateur doit saisir un atome du langage, le système affiche automatiquement le message. Par contre, pour les autres opérateurs, les messages ne sont affichés que sur demande et sont ceux apparaissant dans le phylum every.

Bibliographie

[Bar 81]
P.J.Barnard, N.V.Hammond, J.Morton, J.B.Long, I.A.Clark, Consistency and compatibility in human-computer dialogue, Int.J.Man-Machine Studies (1981) 15, 87-134

[Don 83]
V.Donzeau-Gouge, G.Kahn, B.Lang, B.Mélèse, E.Morcos, Outline of a tool for document manipulation, IFIP, septembre 1983, Paris

[Don 84a]
V.Donzeau-Gouge, B.Lang, B.Mélèse, Practical Applications of a Syntax Directed Program Manipulation Environment, Proceedings of the 7th Int. Conf. on Soft. Eng., Orlando, Florida, March 1984

[Don 84b]
V.Donzeau-Gouge, G.Kahn, B.Lang, B.Mélèse, Documents Structure and Modularity in Mentor, ACM SIGSOFT/SIGPLAN Soft. Eng. Symp. on Practical Software Development Environments, Pittsburgh, April 1984

[Fei 81]
P.H.Feiler, R.Medina-Mora, An incremental programming environment, IEEE Trans. on Soft. Eng. SE-7, No 5, pp.472-481, Sept 1981

[Gai 81]
Brian R.Gaines, The technology of interaction-dialogue programming rules, Int.J.Man-Machine Studies - 1981 p 133-150

[Hef 82]
Michael J.Heffler, Description of a Menu Creation and Interpretation System, Software Practice and Experience, Vol 12, 269-281 (1982)

[Hou 84]
R.C.Houghton, Online Help Systems: A Conspectus, ACM February 1984 Volume 27 Number 2

[Mag 82]
Martin Maguire, An evaluation of published recommandations on the design of man-computer dialogues, Int.J.Man-Machine Studies (1982) 16, 237-261

[Mel 82]
B.Mélèse, Métal, un langage de spécification pour le système Mentor, Technique et Science Informatique (AFCET), Vol. 1 No 4, Juillet-Aout 1982

[Mel 85a]
B.Mélèse, V.Migot, D.Verove, The Mentor-V5 Documentation, Rapport Technique No.43, INRIA, Janvier 1985

[Mel 85b]
B.Mélèse, Edition de Documents Multi-langages sous Mentor-Rapport, Rapport Technique No.54, INRIA, Mai 1985

[Pit 84]
James A. Pitcairn-Hill, Menus and Menu Systems: An Approach to the User Interface, UKC Computing Laboratory Report No 24 - October 1984

[Rei 84]
Steven P.Reiss, A Screen Handler, September 1984

[Rei 85]
Steven P.Reiss, Pecan: Program Development Systems that Support Multiple Views, IEEE Transactions on Software Engineering, Vol. SE 11, No 3, March 1985

[Tei 81]
T.Teitelbaum, T.Reps, The Cornell Program Synthesizer: A Syntax directed programming environment, Communication of the ACM, vol.24, no.9, pp.563-573, Sept 1981

[Wil 83]
R C.Williges, B H.Williges, Human-Computer Dialogue Design Considerations, Automatica vol19 no6 p767-773 - 1983

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique