Version 1.1 – 2012 / 10 / 15

http://www.lias-lab.fr/forge/projects/kmade

Mickaël Baron et Dominique Scapin
(baron@ensma.fr et dominique.scapin@inria.fr)

TABLE DES MATIERES

LICENCE

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order,

agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS

WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING,REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) year name of author
>
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
>
> This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program

`Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989

Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.

AVERTISSEMENT

Ce manuel[1] est destiné à décrire les concepts principaux du modèle et les procédures permettant l'utilisation de l'outil K-MADe.

- **Le chapitre 1** résume les principales caractéristiques du modèle et de l'outil ;
- **Le chapitre 2** concerne le démarrage et l'installation du logiciel ;
- **Le chapitre 3** concerne la création d'un projet, c'est-à-dire d'un « dossier » pour la modélisation des activités ;
- **Le chapitre 4** est la section dédiée à la description de l'activité, sous ses aspects tâches, objets, etc. ;
- **Le chapitre 5** concerne la cohérence du modèle ;
- **Le chapitre 6** détaille l'impression du modèle et la recherche textuelle ;
- **Le chapitre 7** a trait à la recherche d'informations sur le modèle ;
- **Le chapitre 8** décrit le service de simulation ;
- **Le chapitre 9** concerne les préférences utilisateur vis-à-vis de l'outil ;
- **Le chapitre 10** est constitué des annexes (DTDs et Grammaires) ;
- **Le chapitre 11** est constitué d'un exemple complet (prise en main rapide) ;
- **Le chapitre 12** concerne l'historique et la bibliographie.

Remarques préliminaires :

De manière générale, après le sommaire détaillé, le texte qui suit adopte les formats et codages suivants [2] :

- d'abord présentation des concepts à connaître (fond grisé) ;
- puis les instructions d'utilisation de l'outil (fond blanc, titre encadré) ;
- ensuite, les valeurs possibles et les exemples de code sont de couleur verte ;
- enfin, les éléments en cours de développement ou de rédaction sont de couleur bleu.

Un point notable dans l'utilisation de K-MADe est que :

- cet outil peut être utilisé à des objectifs divers (par ex. : modélisation issue des utilisateurs suite à des entretiens, modélisation théorique du système par le concepteur, etc.) ;
- et à divers niveaux d'utilisation (par ex. : description graphique seulement, description graphique avec des relations entre tâches, description complète avec les objets formalisés, etc.).

En effet, de nombreux champs sont facultatifs, ce qui permet une certaine souplesse, et une prise en main progressive. Cependant, certaines des fonctionnalités complètes de l'outil (par ex., la simulation) ne sont possibles que si la modélisation a été la plus complète, précise, et formelle possible (utilisation des objets de l'état du monde).

Contacts en cas de problèmes lors de l'utilisation de K-MADe :

- BARON Mickaël (développement) : mickael.baron@ensma.fr ou baron.mickael@gmail.com
- SCAPIN Dominique (responsable du projet MErLIn) : dominique.scapin@inria.fr

[1]. *Ce manuel est en cours d'édition, notamment pour incorporer les modifications apportées au logiciel K-MADe*
[2] Du fait du codage graphique utilisé, ne pas imprimer / photocopier en noir et blanc, mais en couleur.

Avant tout contact, prière de préparer une description du problème et d'indiquer la version de votre système d'exploitation, la version de machine virtuelle de Java et de la version de K-MADe.

CHAPITRE 1

Généralités sur le modèle et l'outil K-MADe

K-MADe est un outil destiné à contribuer à l'intégration de l'ergonomie au processus de conception des systèmes interactifs, par l'analyse de l'activité et des tâches. L'originalité de cet outil est de se baser sur un modèle dont les capacités d'expression reposent sur une sémantique formelle. Ceci facilite la description et l'analyse de tâches, mais surtout autorise l'interrogation du modèle et le passage entre les modèles du domaine et les étapes du cycle de vie des logiciels.

Ce chapitre donne une description globale et générale du modèle K-MAD et des capacités de l'outil K-MADe en insistant sur :

- Le public visé ;
- Les activités concernées par l'outil ;
- Les modes d'utilisation ;
- La sémantique du modèle ;
- Les capacités de l'outil.

1 - Utilisateurs potentiels

L'outil K-MADe est destiné aux personnes souhaitant décrire, analyser et formaliser les activités d'opérateurs humains, d'utilisateurs, dans des environnements informatisés ou non, en situation réelle ou simulée ; sur le terrain, ou en laboratoire. Bien que toutes sortes de profils de personnes soient possibles, cet environnement est plus particulièrement destiné aux ergonomes et spécialistes en IHM (Interaction Homme-Machine).

2 - Finalités de l'outil K-MADe

K-MADe permet de créer, d'éditer et d'interroger des modèles de tâches utilisateurs. C'est un environnement issu de la recherche en matière d'ergonomie et d'IHM, destiné à faciliter la mise en oeuvre d'une perspective analytique centrée-utilisateur, avec une approche basée sur les modèles (Task-based Models), pour la conception et l'évaluation ergonomique des logiciels interactifs.

Il s'agit de contribuer à l'intégration de l'ergonomie au processus de conception, notamment : de faciliter, par une sémantique formelle et des formats génériques la description, la lecture, et l'analyse des tâches ; de limiter les variations interindividuelles ; d'augmenter le niveau de complétude des descriptions; d'autoriser le traitement informatique ultérieur des descriptions, notamment les interrogations du modèle ; et de faciliter les passerelles entre les modèles du domaine et les étapes du cycle de vie des logiciels.

3 - Phases d'utilisation

K-MADe peut être utilisé lors des phases de recueil sur le terrain ; lors de l'analyse des activités utilisateur ; lors de la validation des modèles ; etc. Il peut être également utilisé pendant les diverses phases du cycle de développement, par exemple : pour la spécification de modèles hypothétiques de tâches, pour aider à la conception et à l'évaluation de l'utilisabilité ainsi qu'à la rédaction de la documentation. K-MADe peut également être envisagé pour aider l'utilisateur final dans sa tâche via une aide sous forme de documentation, notamment lors d'apprentissages.

4 - Modèle K-MAD

K-MADe est basé sur le modèle N-MDA (*Noyau du Modèle de Description de l'Activité*), (Lucquiaud et al., 2002 ; Lucquiaud, 2005a et b) ou (en anglais) K-MAD (Kernel of Model for Activity Description).

Le modèle est hiérarchique. Il représente l'activité de l'utilisateur sous forme d'arbre de tâches, du plus général (tâche-mère) au plus détaillé (actions élémentaires), en passant par des tâches intermédiaires (tâches-filles).

Tâches : une tâche est définie par un nom, un numéro (automatique), un but, une durée, un retour d'information (effets observables par l'utilisateur) :

- une tâche peut donner lieu à des observations;
- une tâche est caractérisée par un niveau d'importance (peu, assez, très) ;
- une tâche est caractérisée par une fréquence (élevée, moyenne, faible) ;
- une tâche est caractérisée par un exécutant (utilisateur, système, interactif, abstrait) ;
- quand l'exécutant est un utilisateur (un individu ou une organisation (groupe d'individus) intervenant dans l'activité décrite), la tâche est caractérisée par des modalités (sensori-motrice, cognitive). Les différents utilisateurs associés aux tâches sont identifiés en tant qu'acteurs, caractérisés par un nom, une expérience (novice, moyenne, expert) et une compétence.
- quand l'exécutant est un système (une machine ou un parc de machines), la tâche peut être associés à des acteurs systèmes, caractérisés par un nom, une expérience (novice, moyenne, expert) et une compétence.

Conditions d'exécution : à une tâche sont associées des conditions d'exécution éventuelles : des préconditions (condition à respecter pour que la tâche puisse être exécutable), des itérations (condition de répétition d'une tâche).

Effets de bord : à une tâche sont associés des effets de bord éventuels : des actions (actions résultantes de la tâche, i.e. dynamique des objets du modèle : modification des valeurs des attributs, création ou suppression d'objets), des évènements (événements pouvant être engendrés lors de l'exécution de la tâche).

Les objets du modèle : ces objets caractérisent l'environnement de l'utilisateur, ce sont les objets qu'il manipule ou qui influencent le déroulement de l'activité (aléas extérieurs dont l'utilisateur n'a pas l'initiative, caractérisés par un nom et une source).

Représentation des objets : les divers types d'objets sont les suivants :

- Objets abstraits : caractéristiques relatives aux concepts manipulés par l'utilisateur ;

- Attributs abstraits : caractéristiques de l'objet abstrait ;
- Objets concrets : correspond à une instance d'un objet abstrait ;
- Groupes : regroupement d'objets concrets ;
- Attributs concrets : permet d'associer une valeur pour chacune des caractéristiques définies par les attributs abstraits de l'objet abstrait ;
- Individus (utilisateur): ensemble des individus qui interviennent dans l'activité décrite ;
- Organisation (utilisateur): ensemble des organisations qui interviennent dans l'activité décrite
- Machine (équipement) : ensemble des machines qui interviennent dans l'activité décrite
- Parc de Machines(équipement) : ensemble des parcs de machines qui interviennent dans l'activité décrite
- Evénements : ensemble des événements qui peuvent être déclenchés ou provoqués par l'activité décrite.

Ordonnancement : une tâche peut avoir plusieurs tâches-filles dont l'ordonnancement est décrit par : un événement déclencheur, une nécessité (optionnelle, obligatoire), une interruptibilité (oui, non), et des opérateurs (séquentiel, alternatif, parallèle, ordre entrelacé et élémentaire).

5 - Outil K-MADe

L'outil K-MADe implémente l'ensemble des caractéristiques du modèle. Il permet d'éditer, modifier et interroger (les fonctionnalités d'interrogation du modèle sont en cours d'élaboration) les modèles de tâches. L'outil est disponible en français et en anglais et se compose principalement des outils suivants :

- Éditeur graphique de modèles de tâches K-MAD. Utilisation de techniques de manipulation directe pour la construction, la manipulation et la suppression des tâches ;
- Éditeur de caractéristiques (voir la liste ci-dessus) de tâches. L'éditeur se présente sous trois formes. Une forme allégée où sont disposées les caractéristiques jugées les plus importantes, une forme qui liste dans des tableaux toutes les caractéristiques et finalement une forme qui présente de manière agencée les caractéristiques de la tâche (précondition, corps de la tâches et effets de bord). Cette dernière représentation est utilisée pour la phase d'impression des caractéristiques d'une tâche ;
- Éditeur d'objets abstrait, d'objets concrets, d'individus, d'organisations, de machines, de parcs de machines et d'événements. Possibilité d'ajouter, modifier et supprimer des objets. La modification et la suppression d'un objet entraînent obligatoirement une modification sur les objets directement liés ;
- Outil d'édition des expressions pour les préconditions, les actions et les itérations. L'outil est à même de vérifier si la grammaire d'une expression est correcte ;
- Simulateur pour animer des modèles de tâches K-MAD ;
- Différents outils d'analyse de modèles de tâches (statistiques, cohérence sur modèle, recherche approfondie, …) ;
- Outil d'impression des arbres de tâches et des caractéristiques des tâches.

CHAPITRE 2

Installation et démarrage

Ce chapitre a trait à l'installation et à l'exécution de l'outil K-MADe. La version actuelle de K-MADe est en cours de développement. Dans ce sens, si vous rencontrez des problèmes quant au téléchargement, à l'installation et à l'exécution, veuillez nous contacter aux adresses suivantes :

- BARON Mickaël à baron@ensma.fr or baron.mickael@gmail.com
- SCAPIN Dominique à dominique.scapin@inria.fr

1 - Pré requis logiciels

Cette version de K-MADe nécessite au minimum la version Java 1.5.0 et fonctionnera sur les versions supérieures. Par ailleurs K-MADe est utilisable sur tous les systèmes d'exploitation qui supportent la version 1.5.0 de Java.

L'outil a été testé sous Windows XP SP2 et sous MAC OS X 10.4

Pour télécharger la machine virtuelle Java veuillez vous rendre sur le site de Sun : *www.java.sun.com*

***Note** : la dernière version de K-MADe utilise le module vidéo QuickTime qui n'est pas encore disponible sur la plateforme Linux. Par conséquent, K-MADe ne fonctionne pas sur des systèmes d'exploitation de type Linux. Toutefois, si vous souhaitez utiliser K-MADe sur une plateforme Linux, une version sans le module multimédia est disponible sur le site Web de K-MADe.*

2 - Pré requis matériel

K-MADe ne nécessite pas une configuration matériel importante : un Pentium 3 pour Windows et un G4 pour MAC OS X avec 256 MO de ram sont suffisants.

Toutefois, pour une utilisation plus souple nous vous recommandons d'utiliser un maximum de mémoire de façon à alléger le système et éviter le swapping de disque qui pourrait causer des ralentissements pendant l'utilisation de K-MADe. Cet aspect lié à la mémoire est d'autant plus préoccupant si vous souhaitez effectuer une modélisation riche.

3 - Installation, mise à jour et démarrage

Télécharger le fichier KMADe à partir du site http://www.lias-lab.fr/forge/projects/kmade.

Nous proposons ci-dessous plusieurs solutions pour le démarrage de K-MADe. Si, parmi ces solutions aucune ne permet d'exécuter K-MADe, veuillez nous contacter.

Windows (fichier EXE)

- Double cliquer sur le fichier exécutable KMADe.

Windows (fichier JAR)

- Double cliquer directement sur le fichier *kmade.jar* (dans le cas où les archives Jar ont été associées avec la machine virtuelle Java).

Ou

- Sauvegardez le fichier archive Jar dans le répertoire « Mes Documents » ;
- Allez sur le menu Démarrer de Windows ;
- Sélectionner « Exécuter … » ;
- Saisir « cmd » (une console normalement sur fond noir apparaît) ;
- A partir de la console saisissez la commande « cd » (le chemin courant est modifié pour être celui du répertoire « Mes Documents ») ;
- Saisir la commande *java –jar kmade.jar*.

MAC OS X (uniquement fichier JAR)

- Double cliquer sur le fichier JAR.

Ou (si problème)

- Configurer la version courante de Java de telle sorte que la version 5.0 soit activée [Applications/Utilitaires/Java/j2SE 5.0/JavaPreferences]) ;
- Double cliquer sur le fichier archive Jar de K-MADe.

4 - Désinstallation

K-MADe génère automatiquement, si l'accès à l'écriture de fichier est autorisé, un fichier *config.ini*. La désinstallation nécessite la suppression de l'archive *K-MADe.jar* et le fichier *config.ini*.

CHAPITRE 3

Création / Edition d'un projet K-MADe

Ce chapitre s'intéresse à la création, l'édition et le traitement d'un projet K-MADe. On décrit dans une première partie la notion de projet puis nous détaillons le fonctionnement d'un projet sous K-MADe.

Concept : projet

La notion de projet recouvre simplement la description générale d'une modélisation particulière ou d'un ensemble de modélisations menés dans un contexte particulier. Il s'agit en quelque sorte d'un dossier dans lequel se trouvent toutes les informations génériques caractéristiques d'un projet devant ou ayant donné lieu modélisation des tâches, par exemple : pour quelle entreprise, quand, dans quel contexte, quels sont les intervenants, etc.

Deux éléments sont à renseigner :

- la description générale ;
- le référencement des intervenants.

Description générale : les informations sur la ou les modélisations courantes :

- Entreprise : le nom de l'entreprise pour laquelle cette modélisation est réalisée (exemple : INRIA) ;
- Site : précise un site de l'entreprise (exemple : Rocquencourt) ;
- Type de poste : la nature du poste modélisé (exemple : Service Administratif) ;
- Date du cas étudié : date de démarrage et de finalisation de la modélisation ;
- Autres Ressources : autres informations qui pourraient aider à la compréhension du contexte dans lequel a été effectué la modélisation de tâches.

Descriptif / Exigences :

- Entreprise : texte, facultatif ;
- Site : texte, facultatif ;
- Type de poste : texte, facultatif ;
- Date du cas étudié : texte, facultatif ;
- Autres Ressources : texte, facultatif.

Référencement des intervenants : auprès de qui les données de la tâche ont-elles été recueillies, et par qui :

- Interviewé : la personne impliquée dans le recueil (exemple : Christelle DuLac) ;
- Coordonnées : les coordonnées de l'interviewé (exemple : bureau 13) ;
- Statut : le statut de l'interviewé au sein de l'entreprise (exemple : assistante de projet) ;
- Ancienneté : précise depuis combien de temps, l'interviewé est dans l'entreprise ;
- Date : date du recueil ;
- Type de recueil : comment s'est déroulé l'entretien (exemples : téléphone, email, face à face) ;
- Informations : informations complémentaires concernant l'interview ou l'interviewé ;
- Interviewer : celui qui est en charge de l'entretien et éventuellement de la modélisation.

Descriptif / Exigences :

- Interviewé : texte, **obligatoire** ;
- Coordonnées : texte, facultatif ;
- Statut : texte, facultatif ;
- Ancienneté : texte, facultatif ;
- Date : texte, facultatif ;
- Type de recueil : texte, facultatif ;
- Informations : texte, facultatif ;
- Interviewer : texte, facultatif.

***Note** : pour une plus grande souplesse d'utilisation, K-MADe rend la saisie facultative de certains champs. Toutefois pour la compréhension des analyses de tâches il est préférable de saisir le plus d'informations possibles.*

1 - Comment débuter une modélisation de tâches ?

- Choisir « Nouveau Projet » dans le menu « Fichier » de K-MADe (si besoin sauvegarder le projet en cours de construction) ;
- Saisir l'ensemble des champs de la description générale et des interviewés.

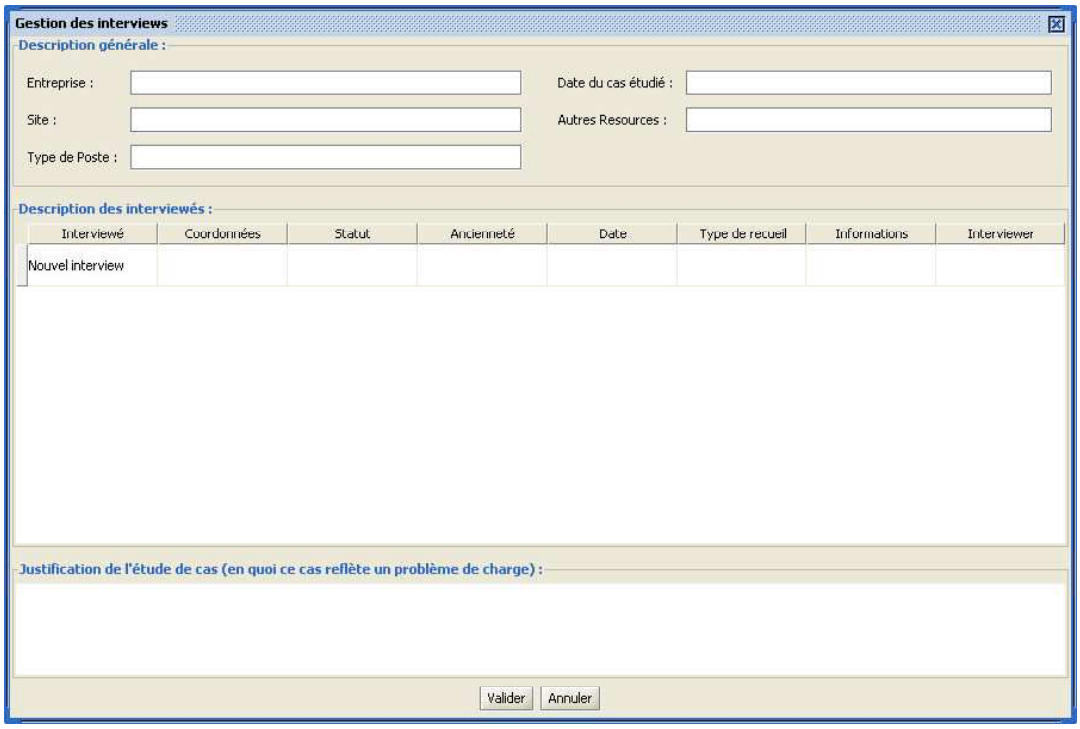

Figure 1 – Aperçu général de la gestion des interviews

La Figure 1 présente l'interface utilisateur permettant de saisir des informations concernant les personnes impliquées dans le recueil et la localisation.

2 - Comment ajouter une nouvelle interview ?

- Double cliquer sur la cellule « Nouvelle interview » ;
- Saisir le nom de la personne impliquée dans le recueil ;

- Compléter les autres informations (coordonnées, statut, ancienneté, date, type de recueil, informations et interviewer).

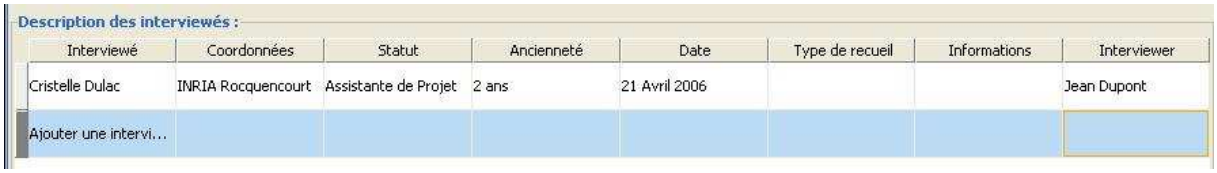

Figure 2 – Zone d'ajout des personnes impliquées dans le recueil

3 - Comment supprimer une interview ?

- Sélectionner la ligne à supprimer ;
- Raccourci clavier « Supprimer ».

4 - Comment modifier les paramètres d'un projet pendant l'édition d'une analyse de tâches ?

- Dans le menu « Projet » de l'outil K-MADe, sélectionner l'action « Propriétés ».

5 - Comment charger un projet ?

Un fichier d'un projet K-MADe porte l'extension « *.kxml » dont la signification est **K**-MADe **XML**.

- Dans le menu « Fichier », sélectionner l'action « Ouvrir Projet ».

***Note** : avant l'ouverture d'un nouveau projet, K-MADe vous demandera si vous souhaitez enregistrer le projet courant (s'il existe).*

6 - Comment fermer un projet ?

- Dans le menu « Fichier », sélectionner l'action « Fermer Projet ».

***Note** : si un projet est en cours d'édition et qu'il n'a pas été sauvegardé depuis des modifications, K-MADe vous demandera avant la fermeture du projet si vous désirez sauvegarder le projet.*

7 - Comment sauvegarder un projet ?

La sauvegarde d'un projet peut s'effectuer de deux manières : soit en indiquant explicitement le fichier de sauvegarde soit en sauvegardant le projet dans le fichier en cours d'édition.

- Dans le menu « Fichier », sélectionner l'action « Enregistrez Projet sous … » pour une sauvegarde explicite.

Ou

- Dans le menu « Fichier », sélectionner l'action « Enregistrer Projet » pour une sauvegarde dans le fichier en cours d'édition.

8 - Comment quitter K-MADe ?

- Dans le menu « Fichier », sélectionner l'action « Fermer Projet ».

***Note** : si un projet est en cours d'édition et qu'il n'a pas été sauvegardé depuis des modifications, il vous est demandé avant l'arrêt de K-MADe si vous désirez sauvegarder le projet.*

CHAPITRE 4

Description de l'activité avec K-MADe

Ce chapitre décrit la partie liée à la description de l'activité qui reste la plus importante et la plus dense pour l'outil K-MADe.

Une première partie a trait à la description des tâches et de leurs caractéristiques.

Plus précisément nous y détaillons toute d'abord la façon de :

- Créer des tâches ;
- Associer des tâches entre elle pour construire un arbre de tâches ;
- La manipulation des tâches (suppression et déplacement) ;
- Les outils qui facilitent l'édition d'un modèle de tâches (copier/coller, agencement automatique, outil de visualisation).

Nous détaillons ensuite les caractéristiques d'une tâche et la manière de les éditer :

- Les caractéristiques générales (numéro, nom, durée, but, multimédia, libellé, affichage, exécutant, etc) ;
- Les caractéristiques d'ordonnancement (décomposition, nécessité, interruptible, déclenchement, acteur, acteur système, précondition et itération) ;
- Les caractéristiques d'effets de bord (événement, actions) ;
- Edition des caractéristiques à différents endroits de l'outil.

Dans une deuxième partie nous traitons la description et l'édition des objets de l'état du monde :

Plus précisément nous débutons par :

- Les objets abstraits ;
- Les groupes ;
- Les attributs abstraits.

Ensuite nous présentons :

- Les objets concrets ;
- Les attributs concrets.

Nous continuons par

- La notion d'utilisateur (individus et organisation) ;
- La notion d'utilisateur système (machines et parcs de machines);
- La relation entre utilisateur et tâche qui représente les acteurs.
- La relation entre utilisateur système et qui représente les acteurs systèmes.

Nous détaillons :

- Les événements ;
- La relation entre événement et tâche.

Nous terminons par :

- Les labels (appelés également libellés);
- La relation entre libellés et tâche.

Enfin dans une troisième partie nous montrons comment effectuer la relation entre les tâches et les objets de l'état du monde et plus précisément nous discutons des expressions contenues dans la précondition, action et itération. **Cette partie est très importante car elle permet de contraindre le déroulement d'une activité et participe activement au fonctionnement de la simulation et de l'interrogation du modèle.**

Nous présentons de manière générale l'outil d'édition d'une expression :

- La construction d'une expression ;
- Les outils de vérification et d'évaluation ;
- L'association entre les objets de l'état du monde et les expressions (objet abstrait, groupe et attribut abstrait)

Puis, pour chaque type d'expression (précondition, action et itération) nous expliquons :

- Les opérations et les fonctions disponibles ;
- Les grammaires pour construire les expressions.

***Note** : ce chapitre est important pour la compréhension du modèle puisqu'il décrit l'essentiel pour construire un modèle K-MAD prenant en compte les tâches mais également les objets de l'utilisateur.*

A - Description et édition de tâches utilisateurs et de leurs caractéristiques

Cette partie est structurée en deux sous parties :

- La première s'intéresse à la création de tâches et de l'arbre de tâches. Nous expliquons également les outils proposés pour faciliter la manipulation d'arbres de tâches ;
- La seconde sous partie a trait d'une part à la présentation des caractéristiques qui composent une tâche, et d'autre part à la manière d'éditer ces caractéristiques. Pour chaque caractéristique d'une tâche, nous détaillons :
  - leur rôle ;
  - leur type ;
  - leur caractère obligatoire ou facultatif ;
  - un exemple d'illustration.

A.1 - Création/Edition de tâches utilisateurs

Concept : Modèle de tâches

Un modèle de tâches K-MAD est un modèle hiérarchique. Il représente l'activité de l'utilisateur sous forme d'arbres de tâches, du plus général (tâche mère) au plus détaillé (actions élémentaires), en passant par des tâches intermédiaires (tâches filles).

1 - Qu'est ce que l'espace de tâches ?

L'espace de tâches est la zone où sont construits et modifiés les modèles de tâches K-MAD. La Figure 3 présente cet espace de tâches. Elle contient une zone d'édition pour construire et manipuler des arbres K-MAD (repère 1) et une boîte à outils de tâches (repère 2) qui offre des outils pour ajouter de nouvelles tâches et des outils pour modifier le taux d'agrandissement de la zone de travail et l'agencement des tâches.

***Note** :K-MADe n'autorise qu'un seul espace de tâches mais celui-ci peut contenir plusieurs arbres de tâches K-MAD (plusieurs racines).*

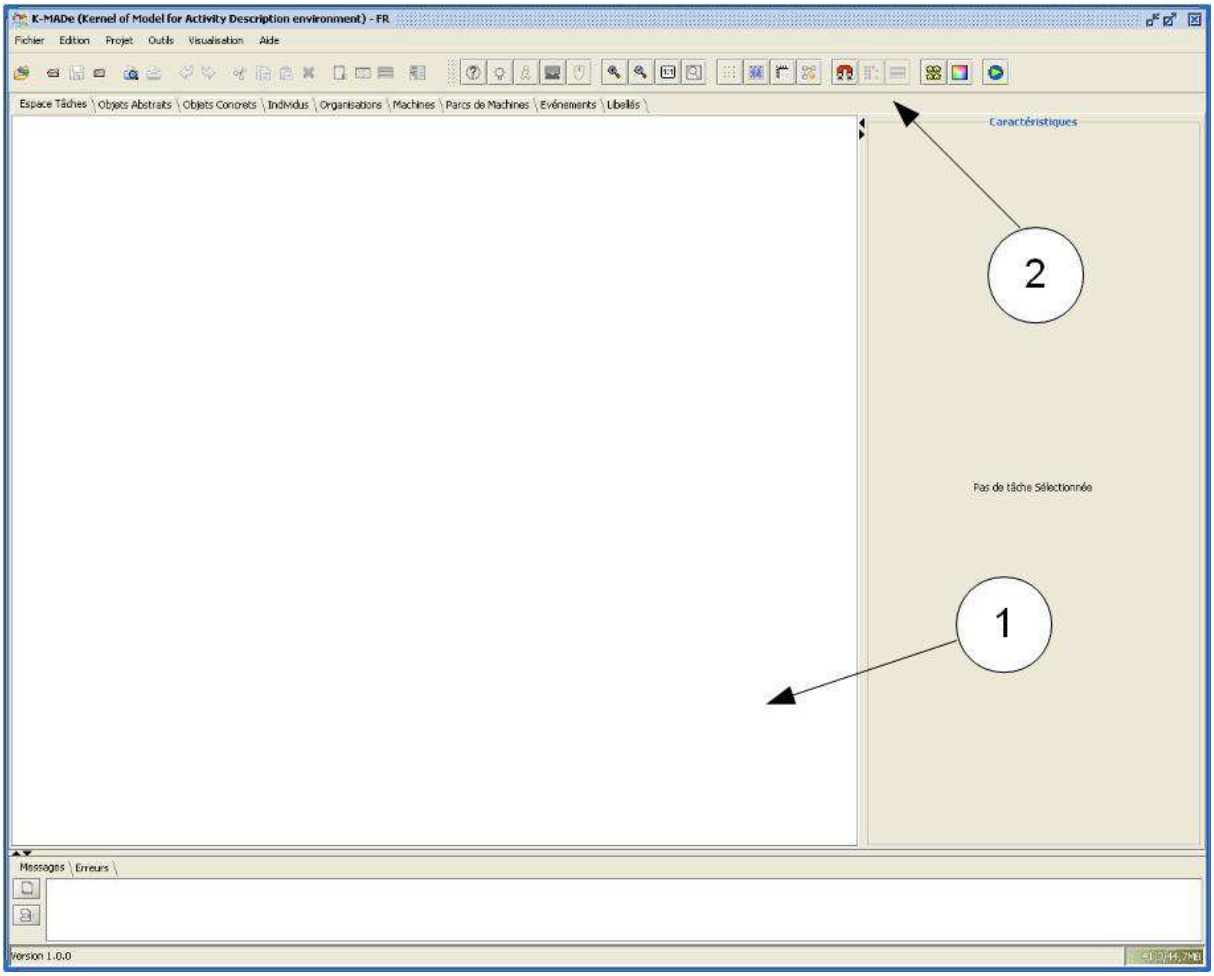

Figure 3 - Espace de tâches

2 - Comment atteindre l'espace de tâches ?

- Cliquer sur l'onglet « Espace de Tâches » en haut de l'outil K-MAD (voir Figure 4)

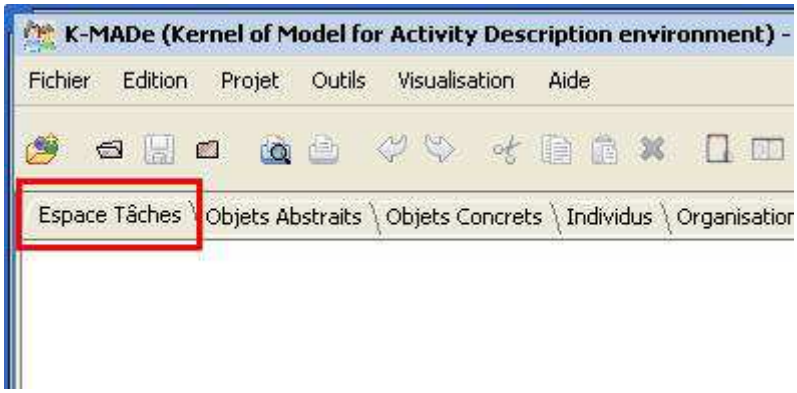

Figure 4 - Onglet pour atteindre l'espace de tâches

3 - Comment créer une nouvelle tâche sur l'espace de tâches ?

Pour créer une nouvelle tâche deux solutions sont proposées :

- Sélectionner dans la boîte à outils de tâches (Figure 5) la catégorie de l'exécutant souhaitée (le concept « exécutant » d'une tâche est décrit en détail dans la section A.2) ;
- Pointer sur l'espace de tâches à l'endroit où vous souhaitez créer la tâche.

Figure 5 - Outils pour créer des tâches

Ou

- Faire apparaître sur l'espace de tâches le menu circulaire en cliquant sur le bouton droit de la souris (Figure 6) ;
- Quand le menu est apparu, choisir la catégorie de la tâche souhaitée.

Figure 6 - Menu circulaire pour créer des tâches

4 - Comment sélectionner des éléments de l'espace de tâches ?

Vous avez la possibilité de sélectionner un ou plusieurs éléments dans l'espace de tâches. Un élément est soit une tâche soit un lien qui relie une tâche mère à une tâche fille.

***Note** : un élément de type lien ne peut être déplacé explicitement. Son déplacement est relatif au déplacement de sa tâche mère ou de sa tâche fille.*

Pour sélectionner un élément :

- Cliquer avec le bouton sélection sur l'élément désiré. La tâche est encadrée d'un trait pointillé coloré (Figure 7).

Figure 7 – Apparence d'une tâche sélectionnée

Pour sélectionner plusieurs éléments (Figure 8):

- Faire un rectangle de sélection qui englobe les tâches que vous souhaitez sélectionnées

Ou

- Tout en maintenant enfoncée la touche « ctrl », sélectionner élément par élément.

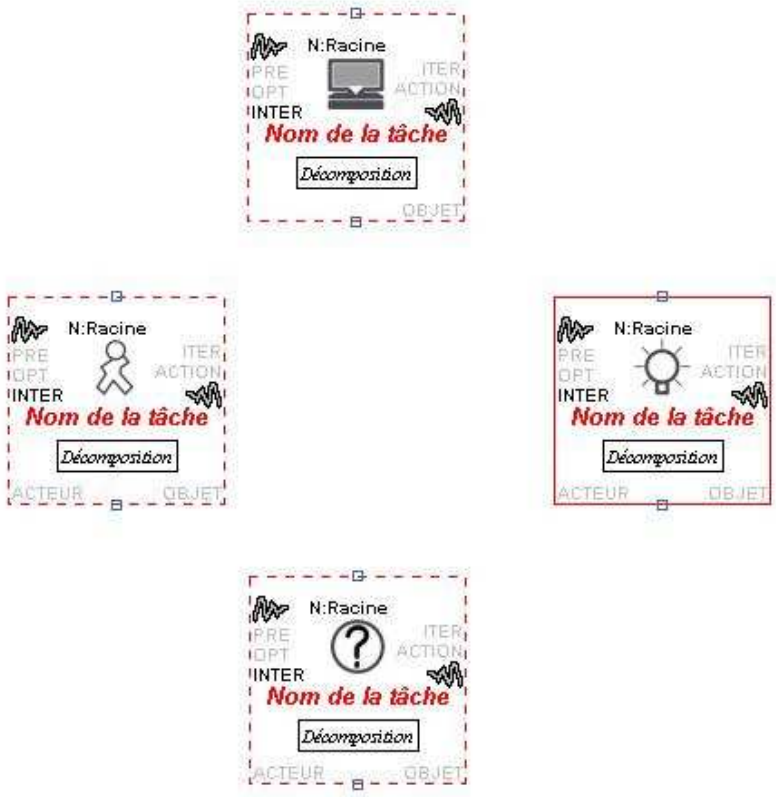

Figure 8 – Trois tâches sélectionnées et une tâche non sélectionnée

5 - Comment modifier la position d'une tâche dans l'espace de tâches ?

- Sélectionner la tâche en cliquant dessus (une zone de sélection apparaît) ;
- Tout en maintenant enfoncé le bouton de sélection de la souris, déplacer la tâche vers sa nouvelle position.

6 - Comment supprimer une tâche ?

- Sélectionner la tâche ;
- Exécuter l'action « Supprimer élément (s) du modèle » du menu principal.

Ou

- Raccourci clavier « Supprimer ».

Ou

- Faire apparaître le menu circulaire en cliquant sur le bouton droit de la souris ;
- Choisir l'action « Supprimer ».

***Note** : quand vous supprimez une ou plusieurs tâche (s) les liens associés sont automatiquement supprimés.*

7 - Comment associer une tâche à une autre tâche ?

Concept : Liaison d'une tâche mère et des tâches filles

Chaque tâche est en relation avec les autres tâches. L'arbre K-MAD est une décomposition hiérarchique du plus général au plus détaillé. A partir de la tâche racine, chaque tâche peut avoir, d'une part, des liens hiérarchiques avec une tâche hiérarchiquement supérieure (tâche-mère) ou inférieures (tâches-filles, lesquelles sont les éléments de la décomposition de la tâche considérée), et d'autre part des tâches-soeurs, de même niveau hiérarchique ... ; etc.

Une tâche graphique K-MAD est composée de deux ancres situées respectivement au niveau supérieur et inférieur. C'est à partir de ces ancres que les liens sont créés et par conséquent les relations entre tâches. Pour créer un lien, il faut choisir l'ancre de départ. Si vous choisissez l'ancre supérieure d'une tâche, vous souhaitez que cette tâche devienne sous-tâche. Au contraire si vous choisissez l'ancre inférieure d'une tâche, cette tâche sera considérée comme tâche mère.

- Sélectionner l'ancre inférieure ou supérieure d'une tâche en déplacement le curseur de la souris dessus (la forme du curseur change) ;
- Tout en maintenant enfoncé le bouton sélection de la souris, déplacer le curseur de la souris vers une autre tâche et l'une de ses ancres associées (un trait est dessiné, Figure 9) ;
- Relâcher le bouton de la souris.

***Note** : si vous faites une erreur lors de la création du lien (relier les ancres d'une même tâche, …), K-MADe n'autorise pas la création du lien.*

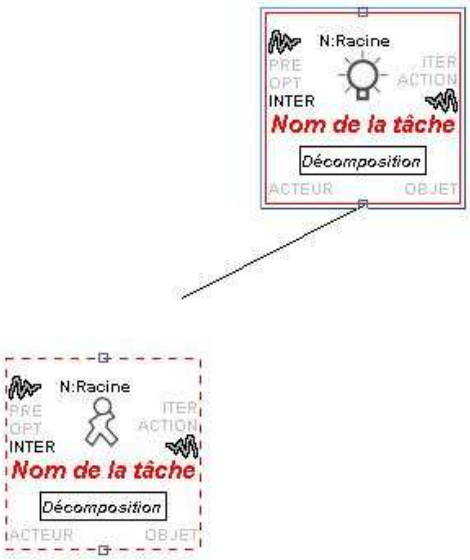

Figure 9 – Association d'une tâche fille à une tâche mère

8 - Comment supprimer une association entre deux tâches ?

Supprimer l'association entre des tâches consiste à supprimer le lien qui les unit.

- Sélectionner le lien ;
- Exécuter l'action « Supprimer élément (s) du modèle » du menu principal.

Ou

- Raccourci clavier « Supprimer ».

9 - Comment modifier l'ordre des sous-tâches ?

Concept : Ordre des sous-tâches

L'ordre des sous-tâches d'une même tâche mère (priorité des tâches ou contrainte de précédence) est défini par la position sur l'espace de tâches. Une tâche placée plus à gauche est considérée prioritaire par rapport à celles qui sont situées sur la droite.

- Sélectionner la tâche que vous souhaitez modifier (sur la Figure 10 nous déplaçons la tâche C);
- Déplacer la jusqu'à la position souhaitée (sur la Figure 11 la tâche D est maintenant précédée par la tâche C).

***Note** : le numéro d'une tâche permet de connaître la position d'une tâche par rapport à ses sœurs.*

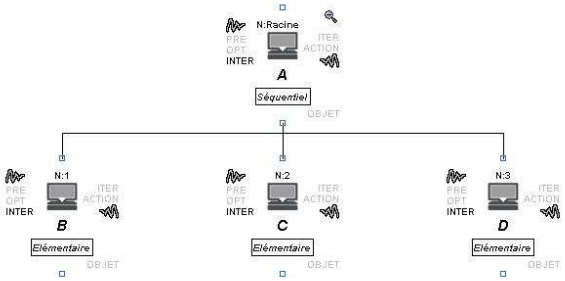

Figure 10 – Sous-tâches ordonnées de A dans l'ordre B, C et D

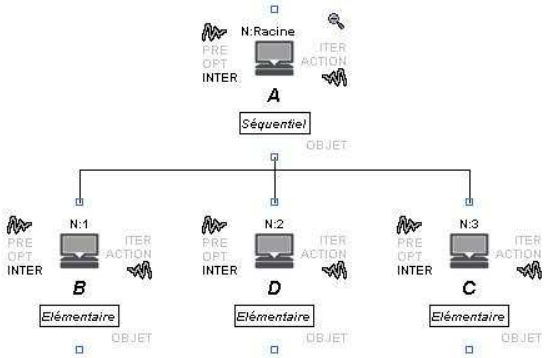

Figure 11 – Sous-tâches ordonnées de A dans l'ordre B, D et C

10 - Comment couper une tâche ou un ensemble de tâche ?

- Sélectionner une ou des tâche(s) à couper ;
- Exécuter l'action « Couper » du menu principal.

Ou

- Raccourci clavier « Couper ».

Ou

- Faire apparaître le menu circulaire dans l'espace de tâches en cliquant sur le bouton droit de la souris ;
- Choisir l'action « Couper ».

11 - Comment copier une tâche ou plusieurs tâches ?

- Sélectionner une ou des tâche(s) à copier ;
- Exécutez l'action « Copier » du menu principal.

Ou

- Raccourci clavier « Copier ».

Ou

- Faire apparaître le menu circulaire dans l'espace de tâches en cliquant sur le bouton droit de la souris ;
- Choisir l'action « Copier ».

12 - Comment coller une tâche ?

- Exécuter l'action « Coller » du menu principal (la tâche « coller » sera positionnée au côté de la tâche copiée / coupée).

Ou

- Raccourci clavier « Coller » (la tâche « coller » sera positionnée au côté de la tâche copiée / coupée).

Ou

- Faire apparaître le menu circulaire sur l'espace de tâches en cliquant sur le bouton droit de la souris ;
- Choisir l'action « Coller » (tâche collée à la position du menu flottant).

13 - Comment réduire / éclater un sous arbre ?

- Cliquer sur l'icône (- pour réduire ou + pour éclater) de la visualisation d'une tâche pour réduire ou éclater un sous arbre (Figure 12 et Figure 13).

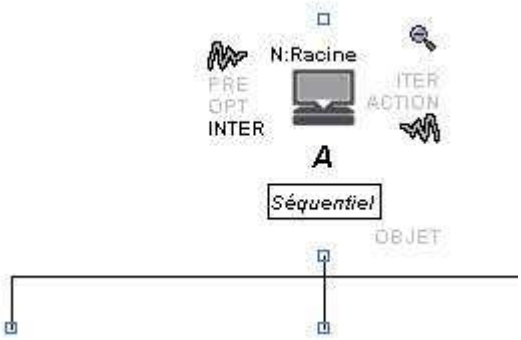

Figure 12 – Tâche Eclatée

Figure 13 – Tâche réduite

14 - Comment modifier automatiquement l'agencement des tâches ?

L'agencement consiste à modifier la position des tâches d'un arbre de tâches de manière uniforme en hauteur (hauteur identique entre tâche mère et sous-tâches) et en largeur (écart identique entre tâches de même niveau).

- Sélectionner la tâche mère dans l'arbre (l'agencement agira sur ces sous-tâches) ;
- Exécuter l'action « re-spatialisation d'une tâche » (Figure 14).

***Note** : l'espace vertical et horizontal entre les tâches peuvent êtres paramétrés.*

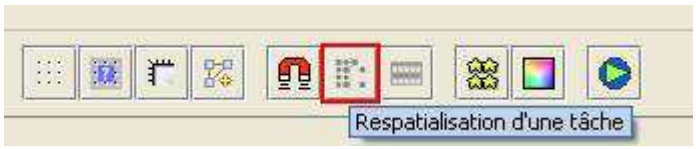

Figure 14 - Outil de re-spatialisation

15 - Comment conserver les contraintes de position de sous-tâches lors des déplacements de la tâche mère ?

Par défaut lorsque vous déplacez une tâche les sous-tâches associées (jusqu'aux feuilles de l'arbre) ne sont pas déplacées. L'option « aimant » permet d'activer le déplacement des sous-tâches lorsqu'une tâche mère est en déplacement.

- Activer l'option « Aimant » dans la boîte à outils de tâches (Figure 15).

Lors des prochains déplacements d'une tâche mère toutes les sous-tâches sont déplacées également. Pour désactiver l'option cliquer une nouvelle fois sur l'option « Aimant ».

***Note** : si une tâche a son sous arbre de réduit l'aimant est automatiquement activé pendant le déplacement de la tâche.*

Figure 15 - Outil Aimant

16 - Comment se déplacer dans un modèle de tâches K-MAD ?

K-MADe fournit un aperçu complet du modèle en construction. La zone visible de l'espace de tâches est symbolisée par un rectangle rouge.

- Ouvrir la fenêtre de l'aperçu complet à partir de la boîte à outils de tâches (Figure 16).

Figure 16 – Outil d'aperçu complet

Pour déplacer la zone visible de l'espace de tâches :

- Sur la fenêtre de l'aperçu complet (Figure 17), sélectionner le rectangle rouge ;
- Effectuer un glissé sur le rectangle rouge et déplacer le curseur de la souris.

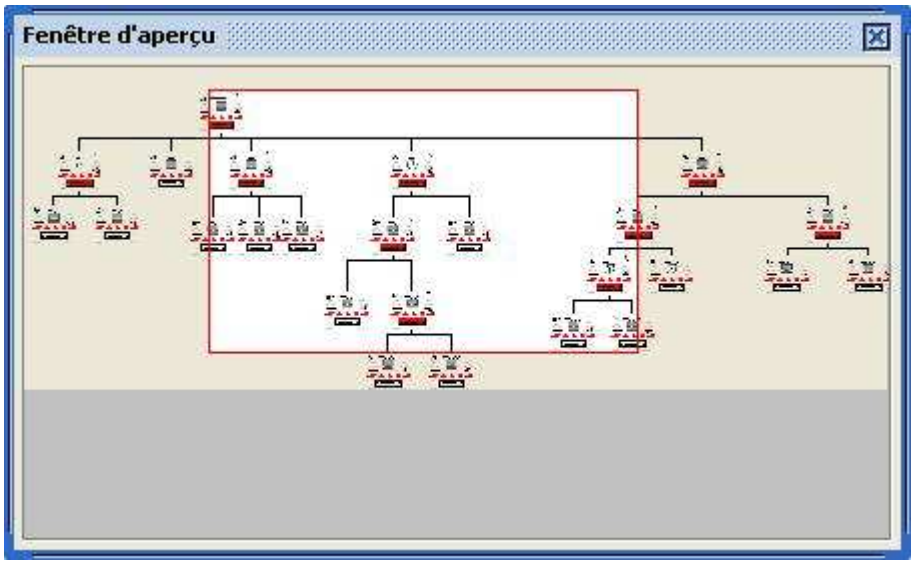

Figure 17 - Aperçu complet d'un arbre de tâches

A.2 - Édition des caractéristiques de tâches

Concept : Caractéristiques de tâches

Une tâche est composée d'un ensemble de caractéristiques qui représentent le pouvoir d'expression relatif à la tâche. Ces caractéristiques se divisent en trois catégories :

- Les caractéristiques générales ;
- Les caractéristiques liées à l'ordonnancement ;
- Les caractéristiques liées aux traitements des actions.

Concept : Caractéristiques de type général

Les caractéristiques générales sont les premières renseignées et sont prévues pour être décrites par l'ensemble des intervenants de l'analyse d'une activité. Ces caractéristiques sont informelles et ne font pas références à des objets de l'état du monde.

Numéro : pour chaque tâche d'un arbre de tâche un numéro unique est défini. Il contient la profondeur de la tâche et sa position parmi ses tâches sœurs. **Remarque : ce numéro est calculé automatiquement par l'outil et n'est pas éditable.**

Descriptif / Exigences : texte, obligatoire

Exemple : N:2.3 (une sous-tâche située au niveau 3 ; troisième tâche de la seconde tâche de la racine)

Nom : *est le nom de la tâche donnée par l'analyste*

Descriptif / Exigences : texte, obligatoire

Exemple : Réserver billets de train

Durée **:** caractérise la durée de la tâche. Toutes sortes d'informations relatives au temps sont acceptées.

Descriptif / Exigences : texte, facultative, pas d'unité

Exemple : 1 heure, 3 jours

But : *désigne l'objectif assigné à la tâche.*

Descriptif / Exigences : texte, facultative

Exemple : cette tâche permet de réserver des billets de trains à partir d'une agence de voyage.

Multimédia : cette caractéristique permet d'associer au but d'une tâche une information multimédia (vidéo ou un son).

Descriptif / Exigences : chemin physique, facultative

Exemple : c:/interview1/reserverBillets.mov

Libellé : cette caractéristique permet d'étiqueter une tâche de manière à pouvoir identifier des tâches. Un libellé ou également appelé label permet notamment d'associer une couleur.

Descriptif / Exigences : Label, facultatif

Note *: par l'intermédiaire de cette caractéristique, l'analyste peut choisir d'afficher ou pas des tâches en sélectionnant les libellés qu'il désire voir actifs. Sur la Figure 18 la sous-tâche du milieu a été associée à un label rendu invisible. Seules les ancres de la tâche sont perceptibles.*

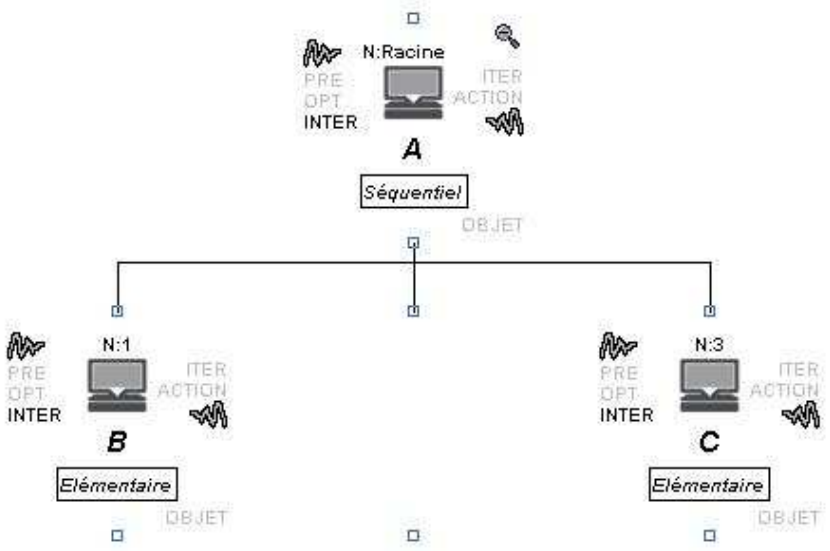

Figure 18 - Utilisation du libellé pour masquer certaines tâches

Objets : Cette caractéristique permet de lister l'ensemble des objets utilisés par une tâche. Les objets peuvent être :

- Objet abstrait, groupe, attribut abstrait ;
- Objet concret et attribut concret ;
- Evénement ;
- Utilisateur ;
- Libellé.

Cette caractéristique n'est pas éditable et est obtenue automatiquement par une extraction des caractéristiques de la tâche qui exploitent des objets.

Descriptif / Exigences : Texte, facultative

Affichage : cette caractéristique permet la description des effets observables (ou souhaitables) par l'utilisateur pendant la réalisation d'une tâche. C'est la réponse du système à l'utilisateur.

Descriptif / Exigences : texte, facultative

Exemple : Le bouton d'ajout est supprimé, une boîte de dialogue modale apparaît

Observation : cette caractéristique permet de donner des informations supplémentaires qui ne peuvent pas être renseignées par les autres caractéristiques. Il s'agit d'un complément d'information relative à l'objectif de la tâche. Par exemple, elle peut servir de pense bête quant à l'enrichissement prochaine de la tâche.

Descriptif / Exigences : texte, facultative

Exemple : Décomposer cette tâche en prenant en compte le caractère itératif

Exécutant : Précise qui réalise la tâche. Les valeurs possibles sont « Utilisateur, Système, Interactif et Abstrait, Inconnue ». Les désignations graphiques sont données sur la Figure 19.

- Une tâche dont l'exécutant est « Utilisateur » implique que la tâche est réalisée par un utilisateur (individu(s) et/ou organisation(s)) plutôt que par une action d'un

équipement(machine(s) et/ou parcs machines) (exemples : acheter billets, réfléchir sur le lieu de départ).

- Une tâche dont l'exécutant est « Système » implique que la tâche est réalisée complètement par un équipement(exemples : impression d'un document).
- Une tâche dont l'exécutant est « Interactif » est une tâche déclenchée par l'utilisateur et réalisée conjointement sur un équipement *(exemples : lancer une impression des billets, cliquez sur parcourir)*.
- Une tâche dont l'exécutant est « Abstrait » est soit une tâche qui n'est pas complètement décrite soit qui contient des sous-tâches d'exécutants différents.
- Enfin une tâche dont l'exécutant est « Inconnue ». Il s'agit d'un état non déterminé.

Descriptif / Exigences : Enumération = {Utilisateur, Système, Interactif, Abstrait, Inconnue}, obligatoire

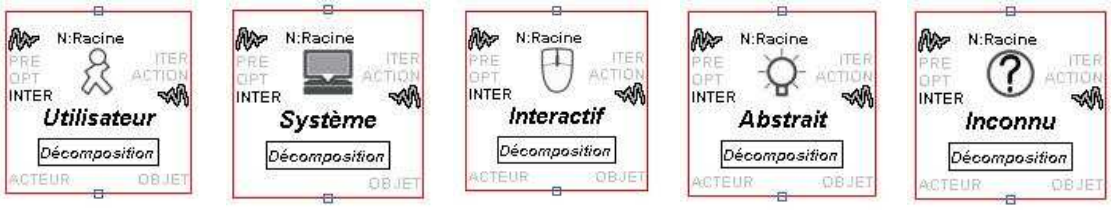

Figure 19 – Exécutants possible Utilisateur, Système, Interactif, Abstrait et Inconnu

Modalité : précise la nature de l'action de l'exécutant « Utilisateur ».

Deux valeurs sont possibles :

- Sensori-Motrice : il s'agit d'une action physique (exemple : remplir l'imprimante de papier) ;
- Cognitive : il s'agit d'une action mentale (exemple : calcul mental des frais de missions).

Descriptif / Exigences : Enumération = {Sensori-Motrice, Cognitive}, obligatoire

***Note** : l'exécutant doit être un « Utilisateur »*

Fréquence : cette caractéristique tient lieu d'appréciation, de la fréquence d'utilisation de la tâche.

Descriptif / Exigences : Enumération = {Elevée, Moyenne, Faible}, facultatif

Valeur Fréquence : cette caractéristique permet de donner une valeur à la fréquence.

Descriptif / Exigences : texte, facultatif

Exemple : 5 fois par jour

Importance : précise l'aspect priorité de la tâche par rapport aux autres tâches sœurs.

Descriptif / Exigences : Enumération = {Très Importante, Assez Importante, Peu Importante, Non Déterminée}, obligatoire

***Note** : cette caractéristique permet lors de la simulation de déterminer un ordre dans l'exécution des tâches.*

Concept : Caractéristiques de type Ordonnancement

Les caractéristiques de type ordonnancement sont les deuxièmes renseignées et participent à la dynamique d'un arbre de tâches K-MAD. Ce sont ces caractéristiques qui déterminent si une tâche est exécutable ou pas.

Nous les présentons dans un ordre quelconque, toutefois, elles sont évaluées suivant un ordre précis lors de la simulation qui est : la décomposition, le caractères nécessité et interruptible en premier, l'acteur en deuxième, le déclenchement en troisième, la précondition en quatrième et l'itération en dernier.

Note : *Les acteurs systèmes ne sont pas pris en compte dans la simulation actuelle.*

Décomposition : cette caractéristique détermine la façon dont les sous-tâches d'une tâche mère vont être exécutées.

- Séquentiel : les sous-tâches sont exécutées une par une et dans l'ordre défini par l'analyste (de gauche à droite) ;
- Parallèle : les sous-tâches sont exécutées de manière concurrente ;
- Alternatif : une seule sous-tâche est exécutée ;
- Pas d'ordre : les sous-tâches sont exécutées de manière entrelacée. Toutes les sous-tâches doivent être exécutées et une par une ;
- Elémentaire : la tâche mère n'est pas décomposable, c'est une feuille de l'arbre.

Descriptif / Exigences : Enumération = {séquentiel, parallèle, alternatif, pas d'ordre, élémentaire}, obligatoire

Note *: cette caractéristique sera détaillée dans la partie Simulation.*

Nécessité : Indique si la tâche est obligatoire ou facultative. Une tâche obligatoire doit être exécutée alors qu'une tâche facultative peut être omise.

Du point de vue de la visualisation de cette caractéristique, elle apparaît au dessus de l'icône de l'exécutant avec le texte OPT. Sur la Figure 20 nous montrons une tâche abstraite optionnelle.

Descriptif / Exigences : Enumération = {Optionnelle, Obligatoire}, obligatoire

Note *: si une tâche a pour exécutant « Système» la valeur de la caractéristique nécessité est « obligatoire ».*

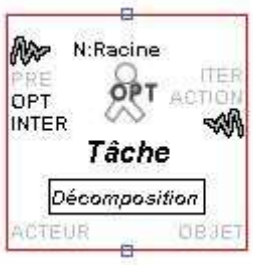

Figure 20 - Tâche abstraite optionnelle

Interruptible : indique si la tâche est interruptible ou pas.

Descriptif / Exigences : Enumération = {Interruptible, Non Interruptible}, obligatoire

Déclenchement : cette caractéristique indique que l'exécution de la tâche est contrainte par un événement.

Descriptif / Exigences : Evénement, facultatif

Exemple : la tâche « exécuter la procédure d'extinction des feux » est exécutée que si l'événement « incendie » est parvenu.

Note *: un seul événement est autorisé*

Acteur : Cette caractéristique contient la liste des utilisateurs autorisée à exécuter la tâche.

Descriptif / Exigences : Utilisateurs, facultatif

***Note** : un seul utilisateur peut exécuter une tâche.*

Acteur système : Cette caractéristique contient la liste des équipements autorisés(machines et/ou parcs de machines) à participer à la tâche.

Descriptif / Exigences : Equipements, facultatif

Précondition : cette caractéristique indique que l'exécution de la tâche est contrainte par une condition booléenne. Si la condition est vrai, la tâche est exécutée sinon elle ne l'est pas. Cette condition est relative aux objets manipulés par l'utilisateur : les objets de l'état du monde.

Descriptif / Exigences : Expression, obligatoire (valeur par défaut : true)

Exemple : isExist($Garage, $Marque == 'Renault')

Itération : cette caractéristique précise l'aspect itératif d'une tâche pouvant être exécutée plusieurs fois ou devant être exécutée jusqu'à la satisfaction d'une condition. Cette condition est relative aux objets manipulés par l'utilisateur.

Descriptif / Exigences : Expression, obligatoire (valeur par défaut : [1])

Exemple : Une tâche « Rechercher Voiture » est itérative, elle est exécutée jusqu'à ce qu'une voiture soit sélectionnée while (isExist($Garage, $Selection != true))

Concept : Caractéristique de type Actions

Les caractéristiques de type action sont les troisièmes renseignées et participent à la modification de l'état du monde. Elles permettent de générer des événements et de modifier/créer ou supprimer des objets concrets de l'état du monde.

Événement : cette caractéristique engendre un ensemble d'événements.

Descriptif / Exigences : Evénements, facultatif

Exemple : génération des événements « annulation conférence », « grève transport ».

Action : cette caractéristique modifie les objets de l'état du monde.

Descriptif / Exigences : Expressions, obligatoire (valeur par défaut : void)

Exemple : set($BudgetEquipe, $Somme -= getValue($Conference, $Prix))

1 - Comment éditer les caractéristiques d'une tâche ?

Pour éditer les caractéristiques d'une tâche plusieurs modes sont disponibles. Les modes se différencient d'une part par la façon de représenter les informations et d'autre part par le nombre de caractéristiques à modifier. K-MADe fournit trois modes :

Édition à partir de l'espace de tâches (voir Figure 21) :

- Sélectionner une tâche ;
- Double cliquer sur l'élément à modifier.

***Note** : les différents éléments modifiables par ce biais sont décrits dans la partie 4.*

Ou

- Choisir l'action « Edition de la tâche » à partir du menu principal ou du menu circulaire.

Édition à partir de l'éditeur complet de tâche (voir Figure 23) :

- Sélectionner une tâche ;
- Choisir l'action « Edition complète de la tâche » à partir du menu principal ou du menu circulaire.

Édition à partir de la zone de caractéristiques (voir Figure 22):

- Sélectionner une tâche ;
- Les caractéristiques sont éditables sur la partie droite de l'outil.

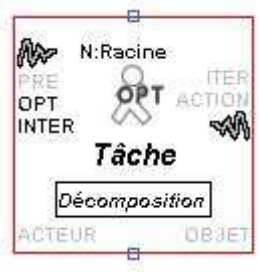

Figure 21 –Edition à partir de l'espace de tâches

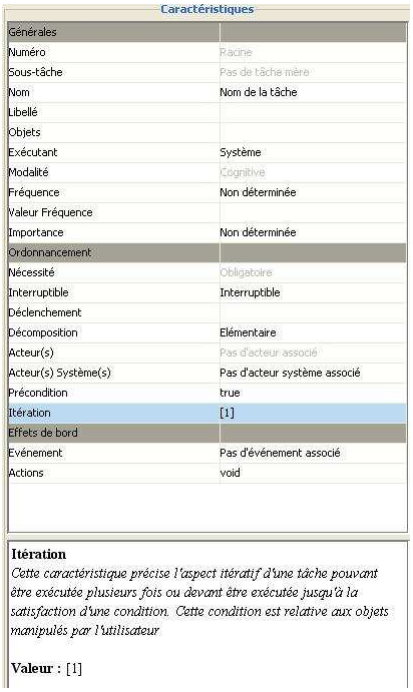

Figure 22 - Edition à partir de la zone de caractéristiques

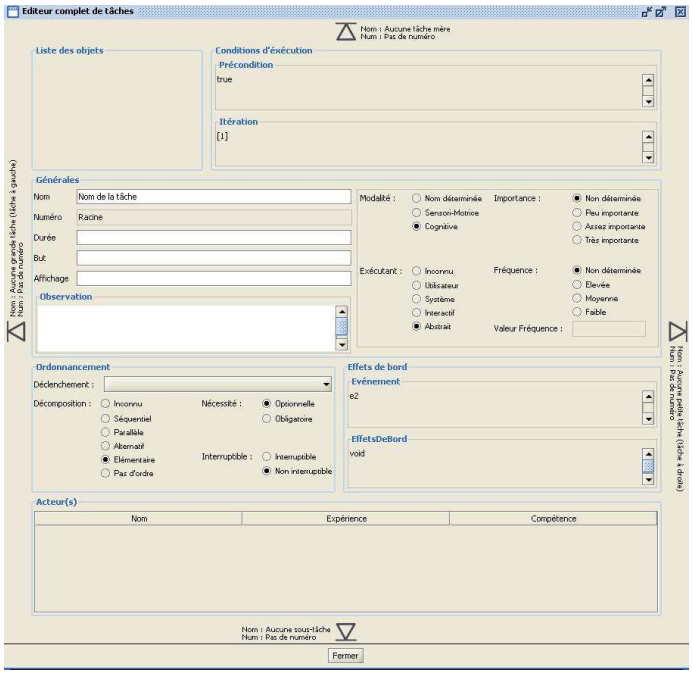

Figure 23 – Editeur complet des caractéristiques de tâches

2 - Comment obtenir des informations sur la signification d'une caractéristique de tâche ?

K-MADe fournit pour chaque caractéristique d'une tâche une légende reprenant les explications données du modèle K-MAD données ci-dessus.

- Sélectionner une tâche ;
- Dans la zone de caractéristiques, sélectionner une caractéristique ;
- La légende est située dans la partie basse de la zone de caractéristique (voir Figure 24).

Figure 24 – Information concernant la caractéristique de tâche « Nécessité » sélectionnée

3 - Description de l'édition de la caractéristique média

Rappelons que cette caractéristique permet d'associer au but d'une tâche une information multimédia (vidéo ou son). K-MADe fournit un lecteur de médias permettant l'ouverture de fichiers sons ou vidéos. Il permet notamment de « marquer » le média pour indiquer une zone correspondant à la description du but de la tâche. Ainsi, vous n'êtes pas obligé d'effectuer un post-traitement des vidéos de vos interviews.

Figure 25 – Outil d'édition de la caractéristique média

4 - Codage graphique d'une tâche par rapport à ces caractéristiques

Nous détaillons ici les différentes caractéristiques que l'on retrouve dans la représentation graphique d'une tâche.

La Figure 26 précise les caractéristiques suivantes :

- Repère 1 : évènement déclencheur (modifiable) ;
- Repère 2 : précondition (modifiable) ;
- Repère 3 : nécessité(modifiable) ;
- Repère 4 : interruptible(modifiable);
- Repère 5 : acteur(modifiable) ;
- Repère 6 : décomposition (modifiable) ;
- Repère 7 : Objets ;
- Repère 8 : nom de la tâche (modifiable) ;
- Repère 9 : événement généré (modifiable) ;
- Repère 10 : action (modifiable) ;
- Repère 11 : itération(modifiable) ;
- Repère 12 : exécutant(modifiable) ;
- Repère 13 : le numéro (calculé automatiquement) ;

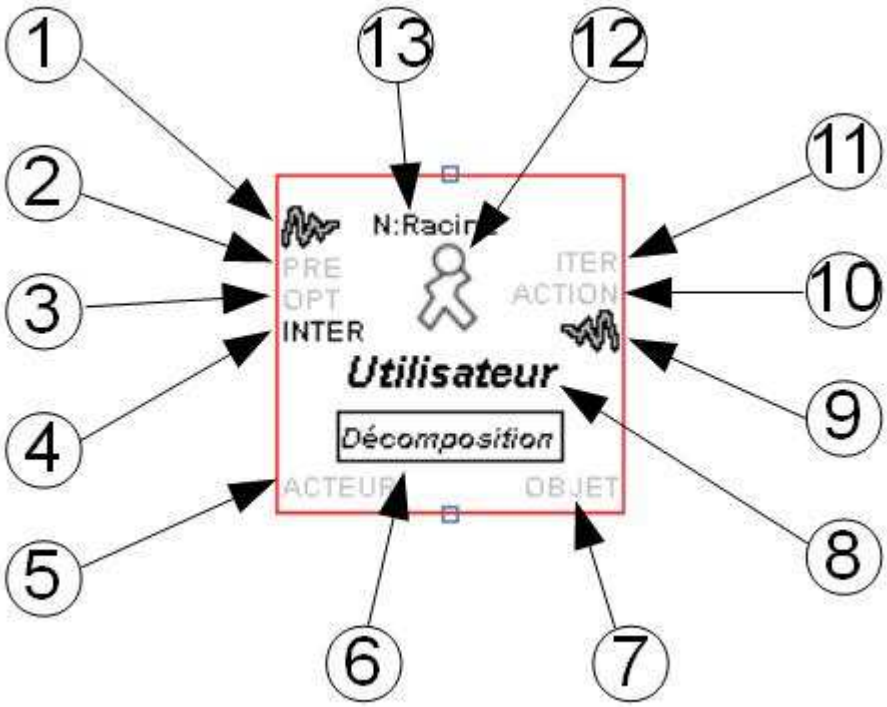

Figure 26 – Codage graphique d'une tâche par rapport à ces caractéristiques.

B - Description et édition des objets de l'état du monde

Cette partie décrit les objets qui interviennent dans la description de l'activité. Nous étudierons la manière de créer, supprimer et éditer les objets suivants :

- Les objets abstraits, les groupes et les attributs abstraits ;
- Les objets concrets et les attributs concrets ;
- Les utilisateurs (individus et organisations) et les acteurs ;
- Les équipements (machines et parcs machines) et les acteurs systèmes ;
- Les événements (déclencheur et générés) ;
- Les labels ou libellés.

Concept : Objets de l'état du monde

Ces objets permettent de décrire les concepts manipulés par l'utilisateur qui ont une influence sur le déroulement de son activité. Le modèle K-MAD propose différents objets (objets et attributs abstraits, groupes, objets et attributs concrets, utilisateurs, équipements, événements et labels).

B.1 - Création/Edition des « objets abstraits »

Concept : Objet abstrait et attribut abstrait

Un objet abstrait est une abstraction de l'objet manipulé utilisé par un utilisateur. Par exemple une « voiture » ou une « mission ». À titre de comparaison, un objet abstrait est semblable au paradigme à objet d'une classe sans l'aspect héritage. Des instances d'objets abstraits peuvent donc être créé et sont appelées « objets concrets ». Un objet abstrait contient un ensemble d'attributs abstraits.

Un attribut abstrait permet d'exprimer des propriétés concernant un objet abstrait.

Sur la Figure 27 nous présentons un schéma d'association des relations entre Objet Abstrait, Attribut Abstrait, Groupe, Objet Concret et Attribut Concret.

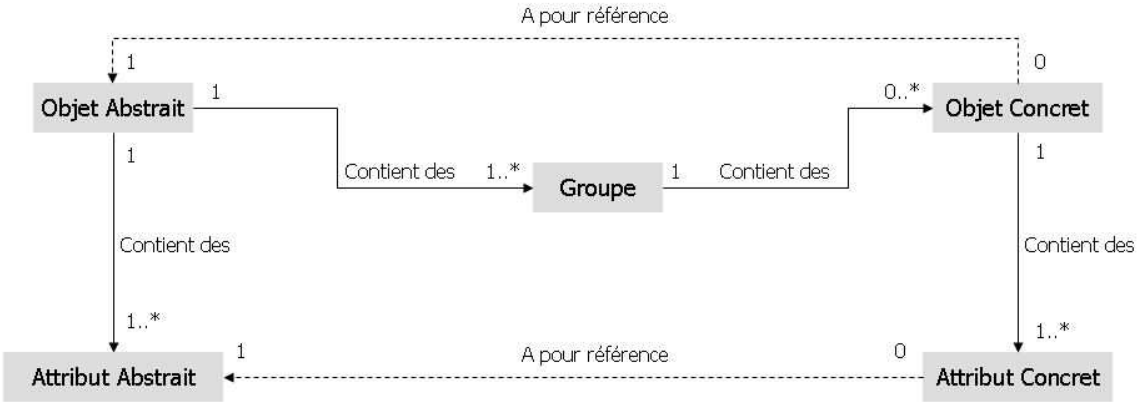

Figure 27 – Schéma des relations entre Objet Abstrait, Attribut Abstrait, Groupe, Objet Concret et Attribut Concret

Objet abstrait : un objet abstrait est identifié par

- Nom : le nom de l'objet ;
- Observation : remarques concernant l'objet abstrait considéré.

Descriptif / Exigences

- Nom : texte, unique, obligatoire
- Observation : texte, facultatif

Exemple : un objet abstrait « Mission »

Note 1 *: un objet abstrait est par ailleurs associé à un ensemble de « Groupe » et d' « attributs abstraits ».*

Note 2 *: un objet abstrait doit être obligatoirement composé d'au moins 1 groupe et 1 attribut abstrait..*

Attribut Abstrait : un attribut abstrait est défini par :

- Nom : le nom de l'attribut abstrait ;
- Description : commentaire sur le rôle de l'attribut abstrait considéré ;
- Type : le type de descripteur utilisé pour l'attribut abstrait.

Cinq types sont disponibles :

- Booléen ;
- Entier ;
- Texte ;
- Enumération ;
- Intervalle.

Descriptif / Exigences

- Nom : texte, obligatoire ;
- Description : texte, facultatif
- Type : Enumération = {Booléen, Entier, Texte, Enumération, Intervalle}

Exemple : pour une « Mission », il peut s'agir de la date de départ (departDate) et la date d'arrivée (arrivéeDate). Ces deux attributs sont tous les deux de types texte.

Note *: K-MAD ne gère pas des références à d'autres objets abstraits.*

Concept : Types composés [Intervalle et Enumération]

Les types composés sont les types construits par l'utilisateur. Il existe deux types composés :

- Les énumérés ;
- Les intervalles.

Intervalle : ce type permet de définir une borne minimum et une borne maximum dans l'ensemble des entiers. Un intervalle est défini par :

- Nom : le nom donné à l'intervalle pour le manipuler ultérieurement ;
- Description : commentaire sur l'intervalle ;
- Valeur minimum : borne inférieure de l'intervalle ;
- Valeur maximum : borne supérieure de l'intervalle ;

Description / Exigences

- Nom : texte, unique, obligatoire ;
- Description : texte, facultatif ;
- Valeur minimum : entier, obligatoire ;
- Valeur maximum : entier, obligatoire.

Exemple : Par exemple pour définir le nombre de place dans une voiture : 1 à 5. Le nom donné à l'intervalle est « nombreDePlaces » dont la valeur minimum est « 1 » et la valeur maximum est « 5 ».

***Note** : le nombre minimum doit être strictement inférieur au nombre maximum*

Enumération : l'objectif de l'énumération est d'ajouter à des valeurs une signification. Il permet de remplacer des valeurs entières par des valeurs textuelles qui sont beaucoup plus « parlantes ». Dans le cas d'une alerte incendie, il y a plusieurs niveaux d'urgence. Sans l'énumération nous aurions utilisé un entier pour les identifier : 3, 2, 1 et 0 correspondant respectivement à urgence faible ; urgence importance, urgence aggravée et extrême urgence. Ainsi manipuler uniquement un entier oblige à ce souvenir de la correspondance. Avec l'énumération seules les valeurs textuelles sont indiquées. Ainsi une énumération est composée :

- Nom : le nom donné à l'énumération pour le manipuler ultérieurement (associer à un type d'un attribut abstrait) ;
- Description : commentaire sur l'énumération ;
- Valeurs : contient la liste des valeurs possibles pour cet énuméré.

Description / Exigences

- Nom : texte, unique, obligatoire ;
- Description : texte, facultatif ;
- Valeurs : List de type texte, au moins une valeur, unique sur l'ensemble des valeurs de l'énuméré considéré.

Exemple : le moyen de transport pour se déplacer à une conférence, le nom est « TransportAutorisé » est les valeurs possibles sont : voiture, avion, train, à pied.

Concept : Groupe

Un groupe est une sorte de structure (Pile, Liste, Ensemble, Unique ou Tableau) qui consiste à stocker les instances d'un même objet abstrait. Ces instances sont appelées objets concrets.

Un groupe est donc associé à un objet abstrait.

Groupe : un groupe est défini par

- Nom : le nom du groupe ;

- Description : explication quant à la signification du groupe ;
- Type : nature de la structure de stockage. Plus précisément, le type de groupe permet de définir des contraintes de cardinalités concernant la gestion des instances des objets abstraits (par la suite nous utiliserons le terme d'objet concret) contenus dans un groupe.

Cinq structures définissant le type de groupe sont disponibles :

- La « Liste » est une structure dans laquelle le premier objet concret stocké est le premier extrait (FIFO), c'est l'objet concret le plus ancien qui est appelé ;
- La « Pile » est une structure dans laquelle le premier objet concret stocké est le dernier extrait (FILO), c'est l'objet concret le plus récent qui est appelé ;
- Un « Ensemble » est une structure dans laquelle, il n'y a pas d'ordre, l'objet concret extrait est déterminé par l'utilisateur ;
- Un « Tableau » est une structure dans laquelle l'ordre d'accès est important, l'emplacement est déterminé par l'utilisateur ;
- L' « Unique » ou « Singleton » est une structure dans laquelle, il n'y a qu'un seul objet concret.

Descriptif / Exigences

- Nom : texte, unique, obligatoire ;
- Description : texte, facultatif ;
- Type : Enumération = {Liste, Pile, Ensemble, Unique,Tableau}, obligatoire.

***Note 1** : Un groupe ne peut pas contenir à la fois des instances d'objet abstrait « Voiture » et « Mission ».*

***Note 2** : Le nom d'un groupe est unique sur l'ensemble des groupes et quelque soit l'objet abstrait considéré.*

Exemple : un objet abstrait « Mission » contient deux groupes « Conférence » et « Enseignement » tous les deux de types « Ensemble ».

1 - Comment atteindre le module « objets abstraits » ?

- Cliquer sur l'onglet « Objets Abstraits » à haut de l'outil K-MAD.

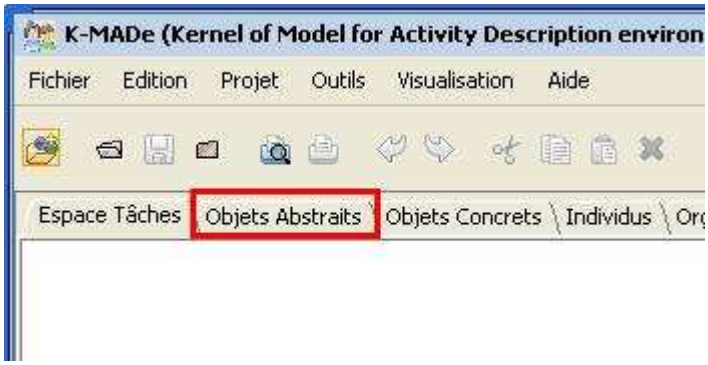

Figure 28 – Onglet pour atteindre le module des objets abstraits et des groupes

2 - Présentation du module d'édition des « objets abstraits » ?

Sur la Figure 29 est présentée le module d'édition des « objets abstraits »

- Zone des « objets abstraits » repère 1 ;
- Zone des « attributs abstraits » repère 2 ;
- Zone des « groupes » repère 3.

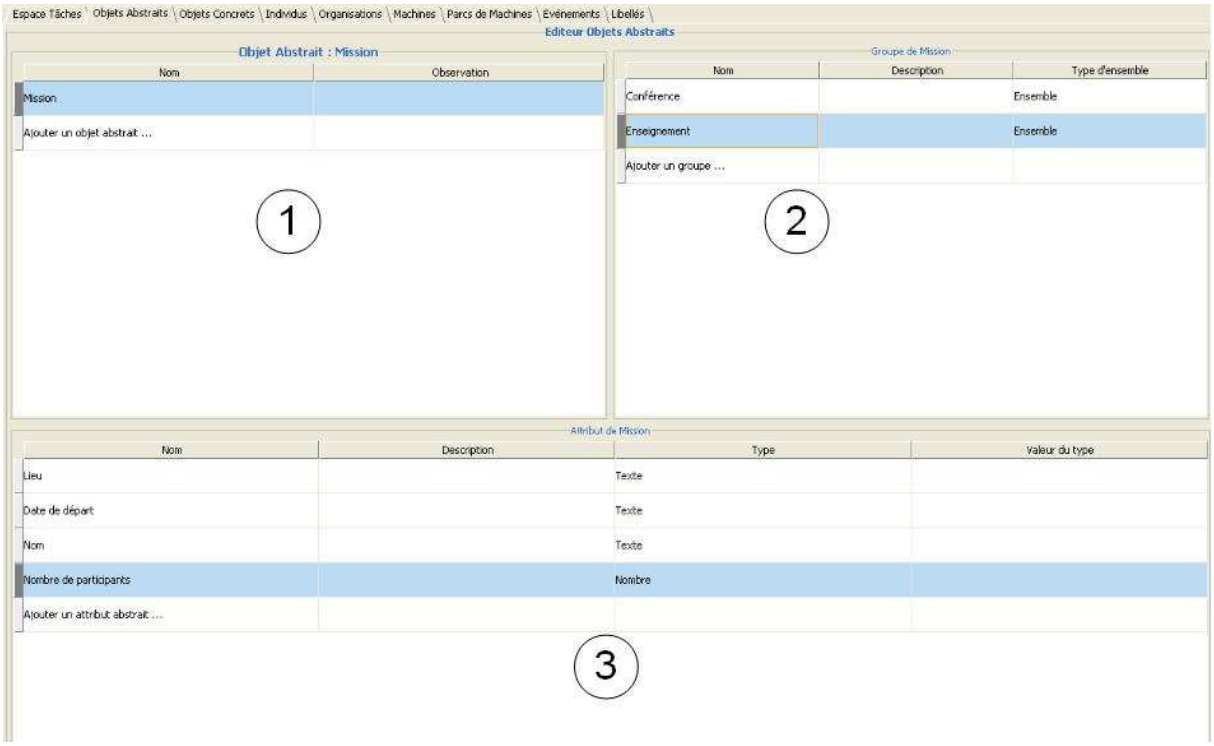

Figure 29 – Editeur des concepts abstraits

3 - Comment créer un objet abstrait ?

- Dans la zone « Objet Abstrait » (Figure 30), sélectionner la cellule « Ajouter un objet abstrait … » (repère 1);
- Double cliquer dessus et saisir un nom (obligatoire) pour l'objet abstrait dans la colonne prévue à cet effet ;
- Saisir éventuellement une observation dans la colonne prévue à cet effet (repère 2).

***Note** : le nom doit être unique. Si un doublon existe, ou qu'une erreur existe dans le nom l'outil vous propose de le renommer. Un nouveau nom vous est proposé si possible.*

Figure 30 - Editeur des objets abstraits

4 - Comment supprimer un objet abstrait ?

- Sélectionner l'objet abstrait dans la partie « Objet Abstrait » ;
- Raccourci clavier « Supprimer ».

***Note** : l'outil supprime automatiquement (après confirmation auprès de l'utilisateur) les attributs abstraits, les groupes, les objets concrets et les attributs concrets en relation directe avec l'objet abstrait à détruire.*

5 - Comment créer un groupe ?

- Sélectionner l'objet abstrait qui sera associé avec le groupe devra appartenir (zone « Objet Abstrait ») ;
- Dans la zone « Groupe » (Figure 31), sélectionner la ligne « Ajouter un groupe … » ;
- Double cliquer dessus et donner un nom au groupe (repère 1);
- Donner éventuellement une description (repère 2);
- Modifier le type (par défaut le type est « Liste ») (repère 3).

***Note** : le nom doit être unique. Si un doublon existe, ou qu'une erreur existe dans le nom l'outil vous propose de le renommer. Un nouveau nom vous est proposé si possible.*

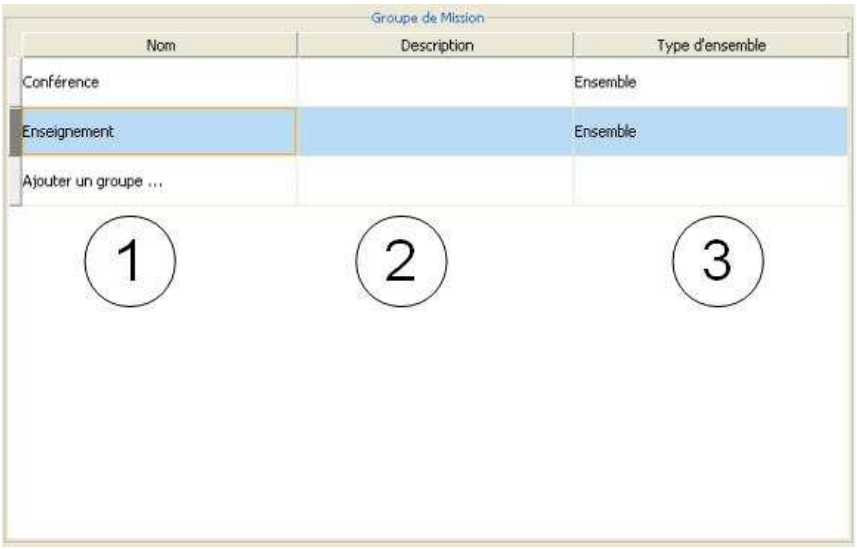

Figure 31 – Editeur des groupes

6 - Comment supprimer un groupe ?

- Sélectionner l'objet abstrait auquel appartient le groupe à supprimer (zone Objet Abstrait) ;
- Dans la zone « Groupe », sélectionner le groupe à supprimer ;
- Raccourci clavier « Supprimer ».

***Note** : l'outil K-MADe supprime automatiquement (après confirmation auprès de l'utilisateur) les objets concrets et les attributs concrets associés à ce groupe.*

7 - Comment créer un attribut abstrait ?

- Sélectionner un objet abstrait pour lequel vous souhaitez ajouter un attribut abstrait (zone Objet Abstrait) (Figure 32);
- Dans la zone « Attribut Abstrait », sélectionner la ligne « Ajouter un attribut abstrait … » ;
- Double cliquer dessus et donner un nom (repère 1) ;
- Donner éventuellement une description (repère 2) ;
- Modifier son type (par défaut le type est « texte ») (repère 3);
- La colonne « Nom du type » est utilisée quand le type est soit une énumération soit un intervalle (repère 4).

***Note** : : le nom doit être unique. Si un doublon existe, ou qu'une erreur existe dans le nom l'outil vous propose de le renommer. Un nouveau nom vous est proposé si possible.*

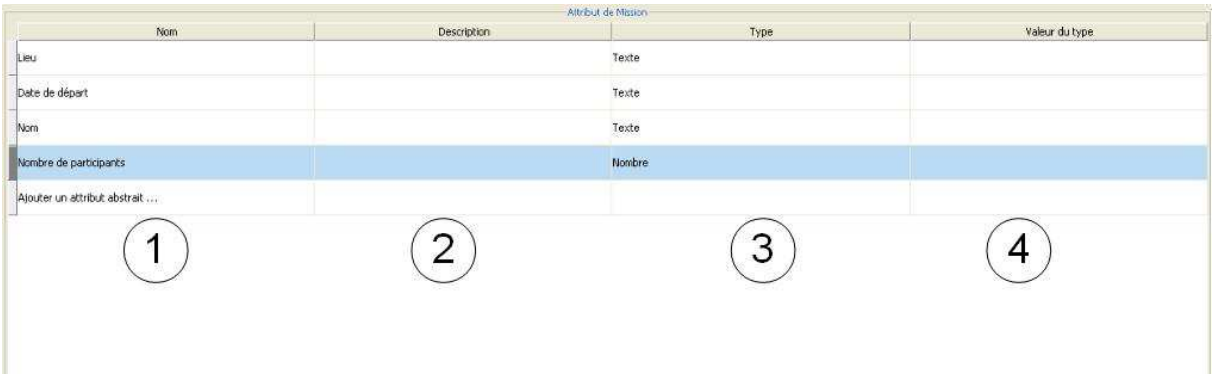

Figure 32 – Editeur des attributs abstraits

8 - Comment supprimer un attribut abstrait ?

- Dans la zone « Objet Abstrait », sélectionner l'objet abstrait auquel appartient l'attribut abstrait à supprimer ;
- Dans la zone « Attribut Abstrait », sélectionner l'attribut abstrait à supprimer ;
- Raccourci clavier « Supprimer ».

***Note** : K-MADe supprime automatiquement (après confirmation auprès de l'utilisateur) les attributs concrets associés à cet attribut abstrait.*

9 - Modification d'un objet abstrait, d'un groupe ou d'un attribut abstrait

- Sélectionner l'élément à modifier dans les zones prévues (objet abstrait, groupe ou attribut abstrait). Dans le cas où il s'agit d'un groupe ou d'un attribut abstrait à modifier, sélectionner en premier lieu l'objet abstrait auquel appartient l'élément à modifier ;
- Modifier les éléments dans les colonnes respectives (nom, désignation, …).

10 - Création d'un intervalle et d'une énumération

Les intervalles et les énumérations ont été désactivés. Ces fonctionnalités seront disponibles dans une prochaine version de l'outil K-MADe.

EN COURS DE DEVELOPPEMENT

11 - Comment supprimer plusieurs éléments en même temps ?

K-MADe offre la possibilité de supprimer plusieurs éléments d'une même zone (s'applique à tous les éléments qui sont présentés dans un table).

- Sélectionner les éléments que vous souhaitez supprimer (utilisation des touches SHIFT et CTRL pour une sélection multiple). Dans le cas où il s'agit d'un groupe ou d'un attribut abstrait à supprimer, sélectionner tout d'abord l'objet abstrait auquel appartient l'élément à modifier ;
- Raccourci clavier « Supprimer ».

B.2 - Création/Edition des « objets concrets »

Concept : Objets concrets

Un objet concret est un objet réel dont la description est dérivée d'un objet abstrait. Il est comparable avec la notion d'instance dans le paradigme à objet. Contrairement à un objet abstrait et aux éléments qui le constituent, un objet concret n'est pas utilisé pendant la description de l'activité. En effet, il participe à la dynamique du modèle et sera donc utilisé pendant la phase d'interrogation du modèle K-MAD (simulation par exemple). Toutefois, nous avons préféré en discuter ici puisque nous utilisons les objets concrets pour évaluer les caractéristiques : précondition, action et itération.

Par ailleurs, un objet concret est composé d'un ensemble « d'attributs concrets ». Un attribut concret est une caractéristique d'un objet concret. La liste des attributs concrets dépend de l'objet abstrait auquel l'objet concret est associé.

Objet concret : un objet concret est identifié par :

- Nom : nom de l'objet
- Observation : remarques concernant l'objet concret ;
- Groupe : nom du groupe dans lequel l'objet concret sera stocké. Par ailleurs, le groupe permet d'indiquer l'objet abstrait qui devra être instancié.

Descriptif / Exigences :

- Nom : texte, unique, obligatoire ;
- Observation : texte, facultatif ;
- Groupe : Groupe, obligatoire

***Note** : un objet concret ne peut être créé que s'il existe au moins un objet abstrait correspondant.*

Exemple : missionLasVegas stocké dans le groupe Conférence et enseignementPoitiers stocké dans le groupe Enseignement sont tous les deux des objets de Mission.

Attribut concret : un attribut concret est défini par :

- Nom : nom défini automatiquement à partir de l'attribut abstrait correspondant défini préalablement. Ce nom n'est pas éditable ;
- Valeur : valeur de l'attribut concret. Cette valeur est un objet dont le type est fonction du type de l'attribut abstrait.

Descriptif / Exigences

- Nom : Nom.Attribut Abstrait, obligatoire, automatique
- Valeur : {Booléen, Nombre, Texte, Enumération, Intervalle}, obligatoire, valeur par défaut : booléen = true ; Nombre = 0 et texte = ''.

Exemple : pour une mission qui contient deux attributs abstraits date de départ et date d'arrivée de type Texte ont pour valeur 12052006 et 15052006.

1 - Comment atteindre le module « objets concrets » ?

- Cliquer sur l'onglet « Objets Concrets » en haut de l'outil K-MADe (Figure 33).

Figure 33 - Onglet pour atteindre le module des objets concrets

2 - Comment créer un objet concret ?

- Dans la zone « Objet Abstrait », sélectionner un objet abstrait auquel l'objet concret doit être associé (création d'une instance d'un objet abstrait particulier) ;
- Dans la partie « Objet concret » et dans la colonne « Nom » sélectionner le groupe désiré en identifiant un nœud « Ajouter un objet concret … » (*Figure 34*);
- Double cliquer sur la ligne « Ajouter un objet concret … » et saisir un nom (repère 1);
- Saisir éventuellement la désignation (repère 2).

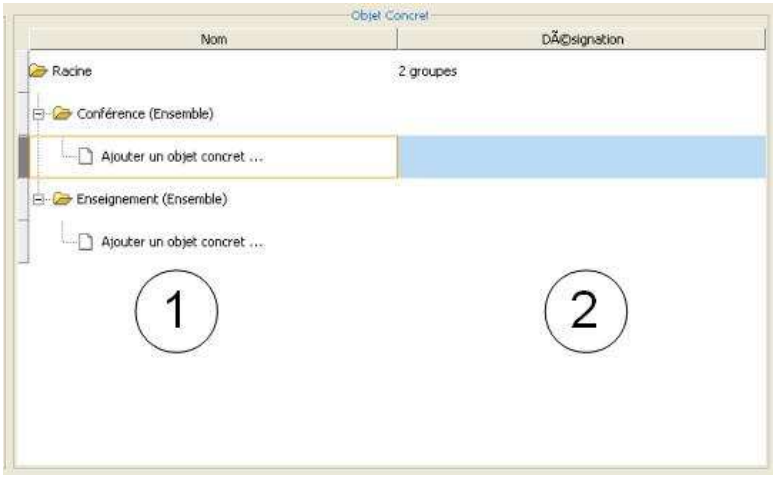

Figure 34 – Editeur des objets concrets

***Note** : le nom doit être unique. Si un doublon existe, ou qu'une erreur existe dans le nom l'outil vous propose de le renommer. Un nouveau nom vous est proposé si possible*.

3 - Comment modifier la valeur d'un attribut concret ?

Lorsque vous avez créé un objet concret, K-MADe construit automatiquement les attributs concrets avec des valeurs par défaut.

- Texte : chaîne vide ;
- Entier : 0 ;
- Booléen : true.

Il vous suffit d'éditer la valeur de l'attribut concret (Figure 35 du repère 1).

- Sélectionner l'objet concret dans la zone objet concret (voir capture d'écran précédente) ;
- Saisir une valeur dans la colonne Valeur (repère 2).

***Note** : K-MADe contrôle automatiquement le type de la valeur saisie en fonction du type de l'attribut abstrait. Si une erreur se produit (un caractère dans une valeur de type entier), K-MADe la refuse.*

Figure 35 – Editeur des attributs concrets

4 - Comment supprimer un objet concret ?

- Sélectionner l'objet abstrait auquel vous souhaitez supprimer un objet concret particulier (zone Objet Abstrait) ;
- Sélectionner l'objet concret dans la zone objet concret ;
- Raccourci clavier « Supprimer ».

***Note 1** : il est impossible de supprimer directement un attribut concret, il faut alors supprimer son attribut abstrait associé.*

***Note 2** : la suppression d'un objet concret implique la suppression des attributs concrets associés.*

5 - Est-il possible de changer directement de groupe (pas d'objet abstrait) pour un objet concret ?

La réponse est non. Le changement de groupe impose une suppression de l'objet concret puis la création dans le nouveau groupe. Les attributs concrets devront être recréés.

- Supprimer l'objet concret concerné ;
- Créer un objet concret dans le groupe désiré ;
- Modifier les valeurs des attributs concrets.

6 - Est-il possible de modifier le type d'un attribut concret ?

La modification du type d'un attribut concret nécessite de modifier le type de son attribut abstrait correspondant.

Toutefois si d'autres attributs concrets sont associés à ces même attributs abstraits, les valeurs de ces derniers sont remises à leur valeur par défaut.

- Ouvrir le module « objets abstraits » ;
- Sélectionner un objet abstrait (zone « objet abstrait ») pour lequel vous souhaitez modifier le type de l'attribut abstrait ;
- Sélectionner l'attribut abstrait (zone « attribut abstrait ») que vous souhaitez modifier ;
- Modifier le type de l'attribut abstrait ;
- Ouvrir le module « objets concrets » ;
- Modifier toutes les valeurs des attributs concrets concernés par cette modification.

B.3 - Création/Edition des « utilisateurs »

Concept : Utilisateur et acteur

Une activité fait intervenir plusieurs personnes. L'objet « utilisateur » dans K-MAD permet de les recenser. Les caractéristiques d'un utilisateur sont décrites indépendamment des tâches qu'il accomplit. Le lien entre tâches et utilisateurs est appelé « Acteur » et fait intervenir d'autres considérations.

Le concept d'acteur désigne l'accomplissement d'une tâche par un ou plusieurs utilisateurs. En d'autres termes, il s'agit d'un objet qui permet de lier un utilisateur à une tâche.

On distingue parmi les utilisateurs : les individus et les organisations.

Les individus sont des utilisateurs uniques tandis que les organisations sont des regroupements d'individus.

Utilisateur : un utilisateur est décrit par :

- Nom : nom donné à l'utilisateur pour l'identifier ;
- Statut : statut de l'utilisateur au sein de l'entreprise ;
- Rôle : précise la place de l'utilisateur dans l'entreprise ;
- Photographie (Logo pour les organisations) : identifie visuellement un utilisateur.
- Organisation / Membres : précise pour un individu les organisations auquel il appartient / précise pour une organisation les membres qui en font partie

Descriptif / Exigences :

- Nom : texte, unique, obligatoire ;
- Statut : texte, facultatif ;
- Rôle : texte, facultatif ;
- Photographie : chemin physique, facultatif.
- Organisation / Membres : utilisation des fenêtres d'inscription, facultatif.

Exemple : l'utilisateur dont le nom est Isabelle Dulac est assistante de projet (statut) son rôle est de s'occuper des missions. Elle fait partie de l'organisation : département informatique.

***Note** : il ne peut y avoir de doublons dans les noms d'utilisateur qu'ils soient individus ou organisations.*

Acteur : un acteur est défini par :

- Nom : nom de l'utilisateur associé ;
- Expérience : expérience de l'utilisateur par rapport à la tâche à accomplir ;
- Compétence : un commentaire éventuel sur l'acteur.

Descriptif / Exigences :

- Nom : Nom. Utilisateur, obligatoire ;
- Expérience : Enumération = {Non déterminée, Expert, Moyenne, Novice}, facultatif ;
- Compétence : Texte, facultatif.

Note 1 *: une tâche peut contenir un ensemble d'acteurs. Une tâche pourra donc être exécutée par plusieurs utilisateurs.*

Note 2 *: si l'exécutant d'une tâche est « système », la tâche ne contiendra aucun acteur.*

Note 3 *: au moins un utilisateur doit exister pour créer un acteur.*

Note 4 *: un utilisateur est utilisé une seule fois comme acteur dans une tâche. Il ne peut y avoir deux acteurs avec un même utilisateur.*

1 - Comment atteindre le module « Individus » et « Organisations » ?

- Cliquer sur un des onglets «Individus» ou « Organisation » en haut de l'outil K-MADe (Figure 36).

Figure 36 – Onglet pour atteindre les modules utilisateurs

2 - Comment créer un individu ?

- Sélectionner l'onglet « Individu »
- Sélectionner la cellule « Ajouter un individu … » (Figure 37) ;
- Double cliquer dessus et saisir un nom d'utilisateur (repère 1) ;
- Saisir éventuellement un statut (repère 2) ;
- Saisir éventuellement un rôle (repère 3) ;
- Saisir éventuellement une photographie (repère 4).

Figure 37 – Editeur d'individus

3 - Comment supprimer un individu ?

- Sélectionner l'individu que vous souhaitez supprimer ;
- Raccourci clavier « Supprimer ».

4 - Comment créer une organisation ?

- Suivez les mêmes étapes que pour créer un individu

5 - Comment supprimer une organisation ?

- Sélectionner l'organisation que vous souhaitez supprimer ;
- Raccourci clavier « Supprimer ».

6 - Comment inscrire un individu à une organisation ?

- Sélectionner l'onglet « Individu ».
- Sélectionner l'individu. Deux tableaux s'affichent (Figure 38).
- Sélectionner l'organisation à laquelle s'inscrit l'individu.(Repère 2)
- Double cliquer pour l'ajouter, ou appuyer sur Entrée.
- Une fois l'organisation dans le tableau de gauche (Repère 1), celui-ci est bien inscrit.

Figure 38 – Editeur d'organisation

Ou

- Sélectionner l'onglet « Organisation ».
- Sélectionner l'organisation. Deux tableaux s'affichent.
- Sélectionner l'individu à inscrire dans le tableau de droite.
- Double cliquer pour l'ajouter, ou appuyer sur Entrée.
- Une fois l'individu dans le tableau de gauche, celui-ci est bien inscrit.

7 - Comment désinscrire un individu d'une organisation ?

- Sélectionner l'onglet « Individu ».

- Sélectionner l'individu. Deux tableaux s'affichent.
- Sélectionner l'organisation dont l'individu se désinscrit dans le tableau de gauche.
- Double cliquer pour le désincrire, ou appuyer sur entrée ou sur Suppr.
- Une fois l'organisation dans le tableau de droite, celui-ci est bien désinscrit.

OU

- Sélectionner l'onglet « Organisation ».
- Sélectionner l'organisation. Deux tableaux s'affichent.
- Sélectionner l'individu à désinscrire dans le tableau de gauche.
- Double cliquer pour le désincrire, ou appuyer sur Entrée ou sur Suppr.
- Une fois l'individu dans le tableau de gauche, celui-ci est bien inscrit.

8 - Comment atteindre le module de création d'acteur ?

- Dans l'espace de tâches, sélectionner la tâche sur laquelle vous souhaitez associer un utilisateur ;
- Éditer la caractéristique « Acteur » (Figure 39).

Interruptible	Interruptible
Déclenchement	
Décomposition	Elémentaire
Acteur(s)	Dulac Estelle
Acteur(s) Système(s)	Pas d'acteur système associé
Précondition	true
Itération	[1]

Figure 39 – Caractéristique pour associer un utilisateur à une tâche

OU

- Dans l'espace de tâches, double cliquer sur « Acteur » dans le cadre de la tâche(Figure 40).

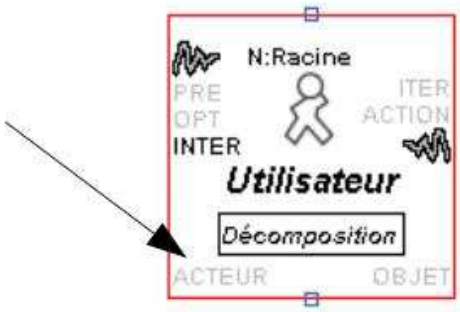

Figure 40 – Caractéristique pour associer une utilisateur à une tâche

9 - Comment ajouter un acteur ?

Deux manières de procéder pour la sélection d'un acteur :

- Sélectionner la cellule « Ajouter un acteur … » dans le tableau du haut (Figure 41 et repère 1);
- Cliquer dessus et sélectionner l'utilisateur que vous souhaitez utiliser ;

Ou

- Sélectionner l'utilisateur que vous souhaitez voir accomplir la tâche dans le tableau du bas (repère 2) ;
- Double cliquer dessus ;

- Modifiez l'expérience éventuellement (repère 3) ;
- Modifiez la compétence éventuellement (repère 4).

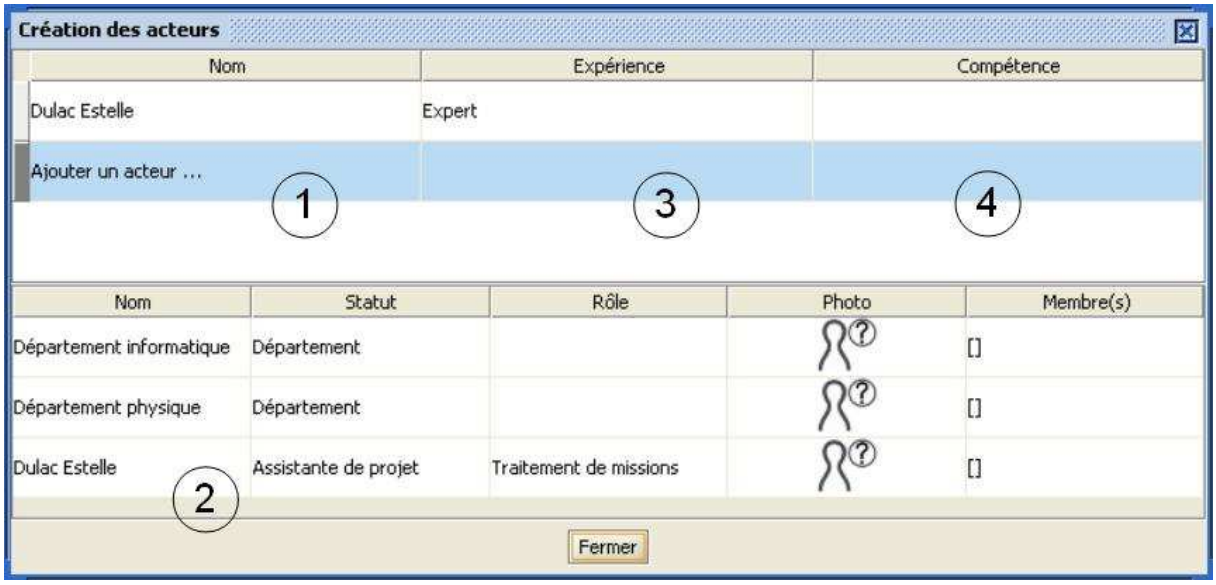

Figure 41 – Outil pour l'édition des acteurs

10 - Comment supprimer un acteur ?

- Sélectionner l'acteur que vous souhaitez supprimer ;
- Raccourci clavier « Supprimer ».

***Note** : quand un utilisateur est supprimé tous ses acteurs (les utilisateurs associés à une tâche) sont automatiquement supprimés.*

B.4 - Création/Edition d'un « équipement »

Concept : Equipement et acteur système

Une activité fait intervenir plusieurs types d 'équipement. L'objet « équipement » dans K-MAD permet de les recenser. Les caractéristiques d'un équipement sont décrites indépendamment des tâches dont il fait partie. Le lien entre tâches et équipements est appelé « Acteur système » et fait intervenir d'autres considérations.

Le concept d'acteur système désigne l'utilisation par une tâche d'un ou plusieurs équipements. En d'autres termes, il s'agit d'un objet qui permet de lier un équipement à une tâche.

On distingue parmi les équipements : les machines et les parcs machines.

Les machines sont des équipements uniques tandis que les parcs machines sont des regroupements de machines.

Equipement : un équipement est décrit par :

- Nom : nom donné à l'équipement pour l'identifier ;
- Description : permet d'ajouter une description à l'équipement ;
- Est informatisé(machine uniquement) : permet de définir si la machine est informatisé ou non ;
- Image : identifie visuellement un utilisateur ;
- Parcs de machines / Machine(s) : précise pour un équipement les parcs auquel il appartient / précise pour un parc les machines qui en font partie ;

Descriptif / Exigences :

- Nom : texte, unique, obligatoire ;
- Description : texte, facultatif ;
- Est informatisé : booléen, obligatoire ;
- Image : chemin physique, facultatif.
- Parcs de machines / Machine(s) : utilisation des fenêtres d'inscription, facultatif.

Exemple : l'équipement dont le nom est Ordinateur BE-12-01 est l'ordinateur 01 se trouvant en salle BE 12 (description), il est informatisé. Il fait partie des parcs de machines : BE et BE 12.

***Note** : il ne peut y avoir de doublons dans les noms d'équipements qu'ils soient machines ou parcs de machines.*

Acteur système : un acteur système est défini par :

- Nom : nom de l'équipement associé ;
- Expérience : expérience de l'équipement par rapport à la tâche à accomplir ;
- Compétence : un commentaire éventuel sur l'acteur système.

Descriptif / Exigences :

- Nom : Nom. Utilisateur, obligatoire ;
- Expérience : Enumération = {Non déterminée, Expert, Moyenne, Novice}, facultatif ;
- Compétence : Texte, facultatif.

***Note 1** : une tâche peut contenir un ensemble d'acteurs systèmes.*

***Note 2** : si l'exécutant d'une tâche est « utilisateur », la tâche ne contiendra aucun acteur système.*

***Note 3** : au moins un équipement doit exister pour créer un acteur système.*

***Note 4** : un équipement est utilisé une seule fois comme acteur système dans une tâche. Il ne peut y avoir deux acteurs systèmes avec un même équipement.*

1 - Comment atteindre le module « Machines » et « Parcs de machines » ?

- Cliquer sur un des onglets «Machines» ou « Parcs de Machines » en haut de l'outil K-MADe (Figure 42).

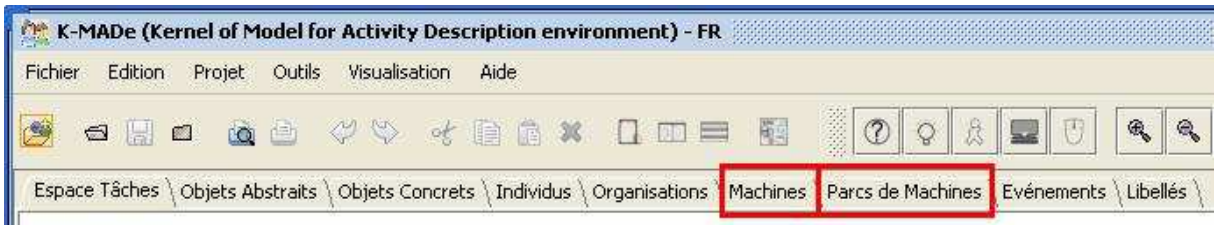

Figure 42 – Onglets pour atteindre les modules équipements

2 - Comment créer une machine?

- Sélectionner l'onglet « Machines »
- Sélectionner la cellule « Ajouter une machine… » (Figure 43) ;
- Double cliquer dessus et saisir un nom de machine (repère 1) ;
- Saisir éventuellement une description (repère 2) ;
- Modifier éventuellement la valeur de « est informatisé »(repère 3) ;
- Saisir éventuellement une photographie (repère 4).

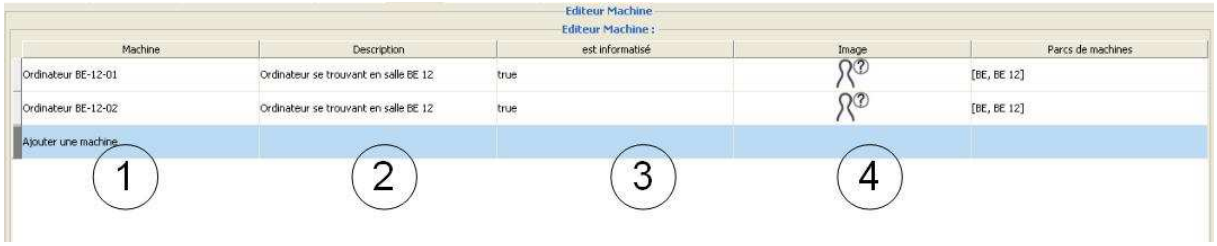

Figure 43 – Editeur d'individus

3 - Comment supprimer une machine?

- Sélectionner la machine que vous souhaitez supprimer ;
- Raccourci clavier « Supprimer ».

4 - Comment créer un parc de machines?

- Suivez les mêmes étapes que pour créer une machine

5 - Comment supprimer une parc de machines ?

- Sélectionner le parc de machines que vous souhaitez supprimer ;
- Raccourci clavier « Supprimer ».

6 - Comment inscrire une machine à un parc de machines ?

- Sélectionner l'onglet « Machines ».

- Sélectionner la machine. Deux tableaux s'affichent. (Figure 44)
- Sélectionner le parcs de machines auquel s'inscrit la machine.(Repère 2)
- Double cliquer pour l'ajouter, ou appuyer sur Entrée.
- Une fois le parce de machines dans le tableau de gauche (Repère 1), celui-ci est bien inscrit.

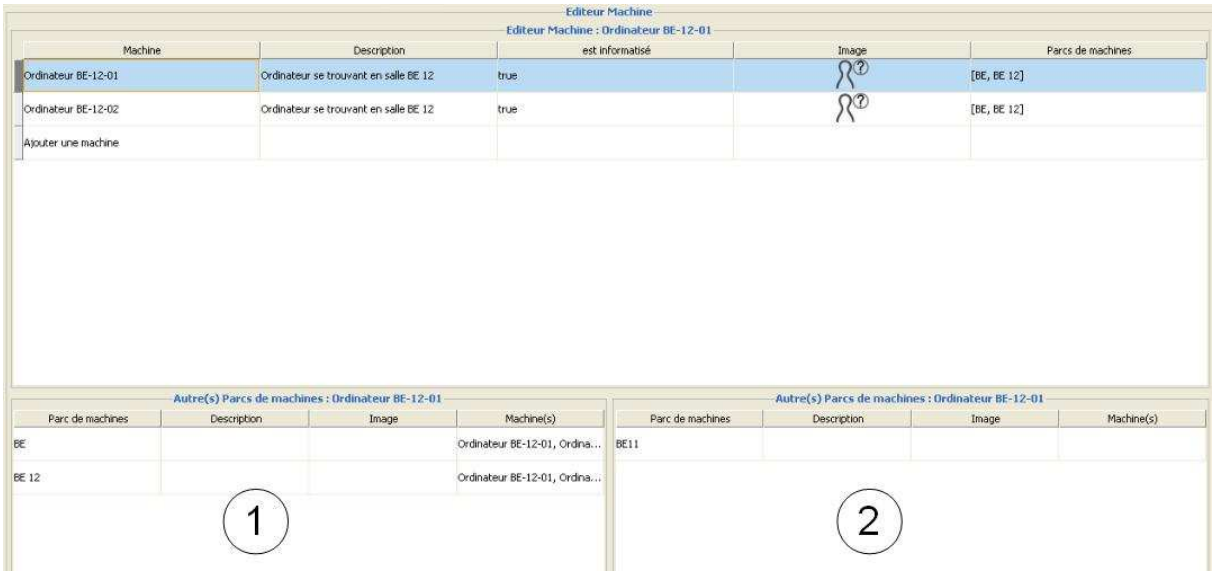

Figure 44 – Editeur de machines

Ou

- Sélectionner l'onglet « Parc de Machines ».
- Sélectionner le parc de machines. Deux tableaux s'affichent.
- Sélectionner la machine à inscrire dans le tableau de droite.
- Double cliquer pour l'ajouter, ou appuyer sur Entrée.
- Une fois la machine dans le tableau de gauche, celle-ci est bien inscrite.

7 - Comment désinscrire une machine d'un parc de machines ?

- Sélectionner l'onglet « Machines ».
- Sélectionner la machine. Deux tableaux s'affichent.
- Sélectionner le parc de machines dont la machine se désinscrit dans le tableau de gauche.
- Double cliquer pour le désincrire, ou appuyer sur entrée ou sur Suppr.
- Une fois le parc de machines dans le tableau de droite, celui-ci est bien désinscrit.

OU

- Sélectionner l'onglet « Parc de machines ».
- Sélectionner le parc de machines. Deux tableaux s'affichent.
- Sélectionner la machine à désinscrire dans le tableau de gauche.
- Double cliquer pour le désincrire, ou appuyer sur Entrée ou sur Suppr.
- Une fois le machine dans le tableau de gauche, celle-ci est bien inscrite.

8 - Comment atteindre le module de création d'acteur système?

- Dans l'espace de tâches, sélectionner la tâche sur laquelle vous souhaitez associer un utilisateur ;
- Éditer la caractéristique « Acteur système » (Figure 45).

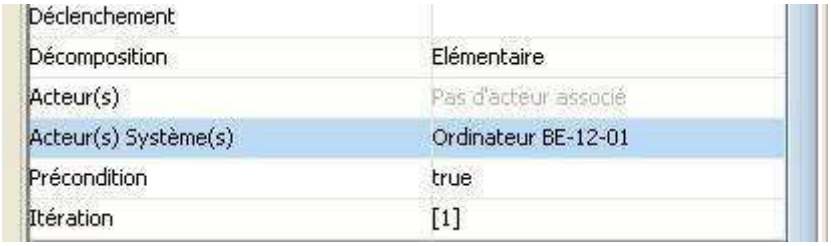

Figure 45 – Caractéristique pour associer un équipement à une tâche

9 - Comment ajouter un acteur système ?

Deux manières de procéder pour la sélection d'un acteur système :

- Sélectionner la cellule « Ajouter un acteur système … » dans le tableau du haut (Figure 46 et repère 1);
- Cliquer dessus et sélectionner l'équipement que vous souhaitez utiliser ;

Ou

- Sélectionner l'équipement que vous souhaitez voir accomplir la tâche dans le tableau du bas (repère 2) ;
- Double cliquer dessus ;

- Modifiez l'expérience éventuellement (repère 3) ;
- Modifiez la compétence éventuellement (repère 4).

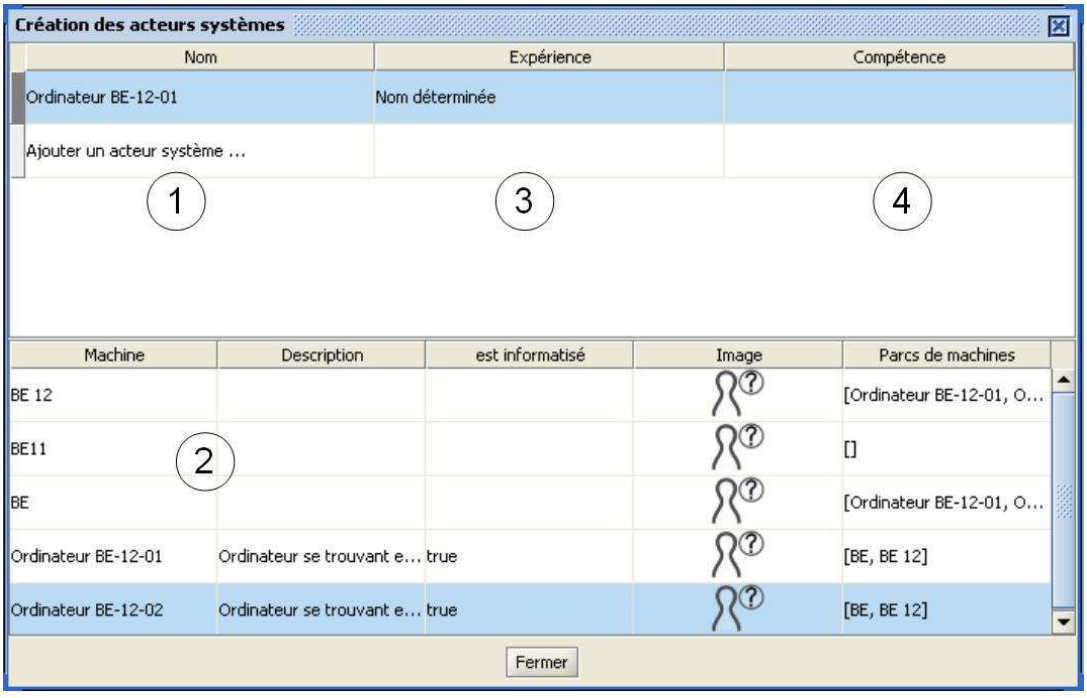

Figure 46 – Outil pour l'édition des acteurs systèmes

10 - Comment supprimer un acteur système?

- Sélectionner l'acteur système que vous souhaitez supprimer ;
- Raccourci clavier « Supprimer ».

***Note** : quand un équipement est supprimé tous ses acteurs systèmes (les équipements associés à une tâche) sont automatiquement supprimés.*

B.5 - Création/Edition des « événements »

Concept : Evénement

Un événement est un aléa qui n'est pas du ressort de l'utilisateur mais qui influence le déroulement de son activité *(Exemple : l'absence de papier dans une imprimante)*. La notion d'événement permet d'enrichir la description de l'activité en intégrant l'environnement de l'utilisateur qui n'est pas toujours descriptible avec les objets de l'état du monde.

Evénement : un événement est décrit par

- Nom : nom donné à l'événement pour l'identifier ;
- Description : détail le rôle de cet événement.

Descriptif / Exigences :

- Nom : texte, unique, obligatoire ;
- Description : texte, facultatif.

Exemple : un événement « annulation conférence ».

***Note** : il ne peut y avoir de doublons dans les noms d'événement.*

1 - Comment atteindre le module « événement » ?

- Cliquer sur l'onglet « Evénements »

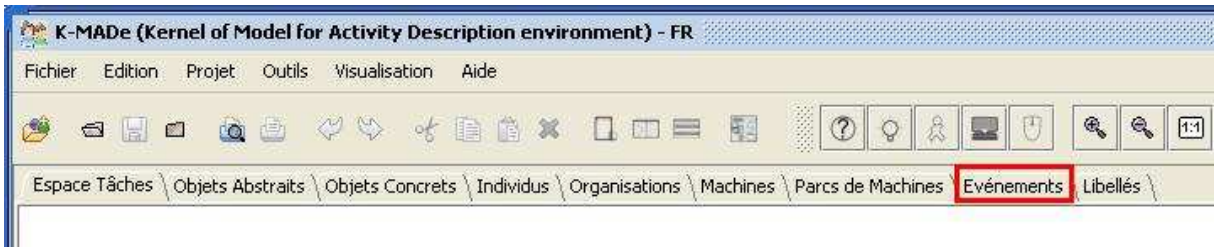

Figure 47 – Onglet pour atteindre le module événements

2 - Comment créer un événement ?

- Sélectionner la cellule « Ajouter un événement … » (Figure 48);
- Double cliquer dessus et saisir un nom d'événement (repère 1) ;
- Saisir éventuellement une description (repère 2).

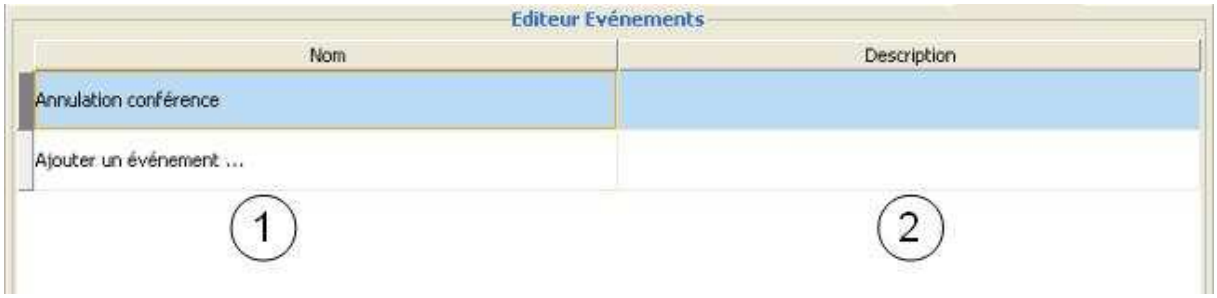

Figure 48 – Outil pour l'édition des événements

3 - Comment supprimer un événement ?

- Sélectionner l'événement que souhaitez supprimer ;
- Raccourci clavier « Supprimer ».

***Note** : quand un événement est supprimé, tous les événements associés aux tâches sont automatiquement supprimés.*

4 - Comment associer un événement déclencheur à une tâche ?

- Dans l'espace de tâches, sélectionner la tâche avec laquelle vous souhaitez associer un événement déclencheur (Figure 49) ;
- Éditer la caractéristique « Déclenchement » en choisissant l'événement.

Figure 49 – Caractéristique pour associer un événement déclencheur à une tâche

OU

- Dans l'espace de tâches, double cliquer sur l'icône indiqué dans la Figure 50 dans le cadre de la tâche.

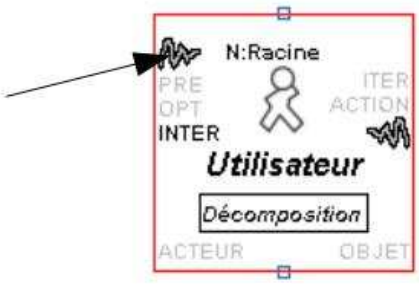

Figure 50 – Caractéristique pour associer un événement déclencheur à une tâche

5 - Comment choisir les événements générés par une tâche ?

- Dans l'espace de tâches, sélectionner la tâche avec laquelle vous souhaitez pouvoir générer des événements ;
- Éditer la caractéristique « Evénements » en double cliquant dessus (Figure 51) (une boîte de dialogue apparaît).

Décomposition	Elémentaire
Acteur(s)	Pas d'acteur associé
Acteur(s) Système(s)	Pas d'acteur système associé
Précondition	true
Itération	[1]
Effets de bord	
Evénement	Pas d'événement associé
Actions	void

Figure 51 – Caractéristique pour associer à une tâche des événements générés

OU

- Dans l'espace de tâches, double cliquer sur l'icône indiqué dans la Figure 52 dans le cadre de la tâche

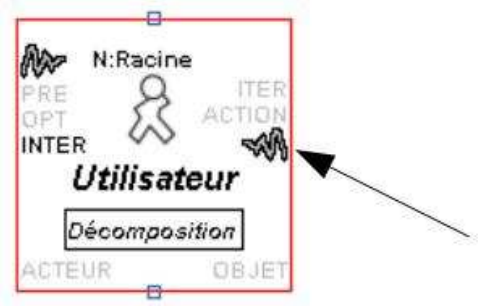

Figure 52 – Caractéristique pour associer à une tâche des événements générés

- Sélectionner la cellule « Ajouter un événement … » dans le tableau du haut (Figure 53);
- Cliquer dessus et sélectionner l'événement que vous souhaitez déclencher (repère 1) ;

Ou

- Sélectionner l'événement que vous souhaitez déclencher dans le tableau du bas (repère 2) et double cliquer dessus.

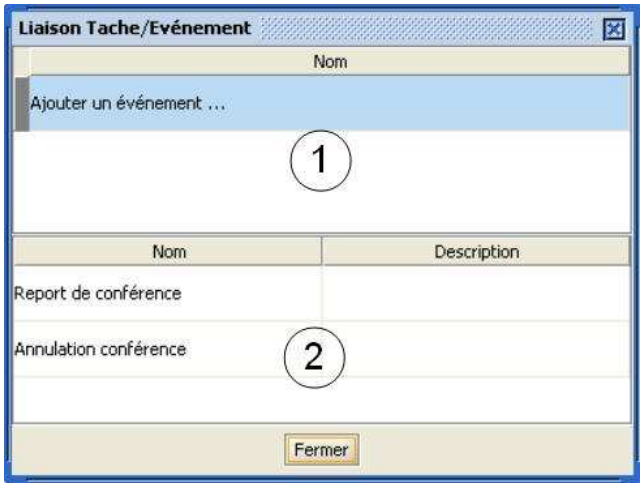

Figure 53 – Outil pour l'édition des événements « générables » par une tâche

***Note** : un événement ne peut être généré plus d'une fois par tâche.*

B.6 - Création/Edition des « libellés »

Concept : Libellés ou labels

Un label est une étiquette d'identification de groupes de tâches. Contrairement au objets précédents, un label n'intervient pas sur le déroulement de l'activité (notamment pas d'influence sur la simulation du modèle K-MAD). Les labels sont utilisés pour aider à la visualisation différentielle des arbres de tâches.

Libellés ou labels : un libellé est décrit par :

- Nom : nom donné au label pour l'identifier ;
- Description : détaille le rôle de ce label ;
- Couleur : associe au label une couleur ;
- Couleur Visible : indique si la couleur du label doit être prise en compte ;
- Visible : indique si les tâches associées doivent être affichées ou pas.

Descriptif / Exigences :

- Nom : texte, obligatoire ;
- Description : texte, facultatif ;
- Couleur : Couleur, facultatif (par défaut BLANC) ;
- CouleurVisible : booléen, facultatif (par défaut « true ») ;
- Visible : booléen, facultatif (par défaut « true »).

***Note** : il ne peut y avoir de doublons dans les noms des labels.*

1 - Comment atteindre le module « libellés » ?

- Cliquer sur l'onglet « Libellé »

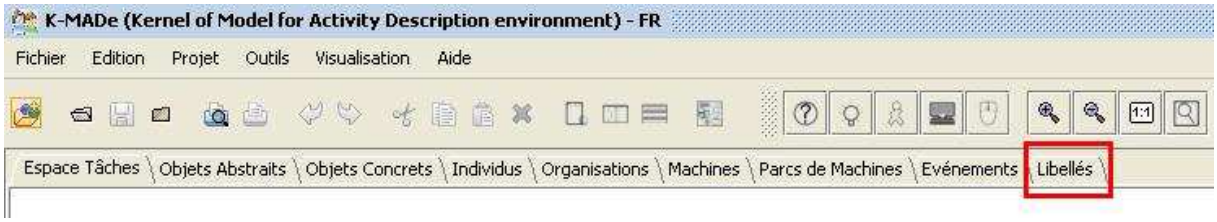

Figure 54 - Onglet pour atteindre le module libellé

2 - Comment créer un libellé ?

- Sélectionner la cellule « Ajouter un libellé … » (Figure 55) ;
- Double cliquer dessus et saisir un nom de libellé (repère 1);
- Saisir éventuellement une description (repère 2);
- Saisir éventuellement une couleur (repère 3);
- Modifier éventuellement le caractère de visibilité de la couleur (repère 4) ;
- Modifier éventuellement le caractère de visibilité des tâches associées à ce libellé (repère 5).

Figure 55 – Outil pour l'édition des libellés

3 - Comment supprimer un libellé ?

- Sélectionner le libellé que souhaitez supprimer ;
- Raccourci clavier « Supprimer ».

Note : lors de la suppression d'un libellé les relations avec les tâches associées sont supprimées.

4 - Comment associer un libellé à une tâche ?

- Dans l'espace de tâches, sélectionner la tâche avec laquelle vous souhaitez associer un libellé (Figure 56) ;
- Éditer la caractéristique « Libellé » en choisissant un libellé à partir de la liste fournie.

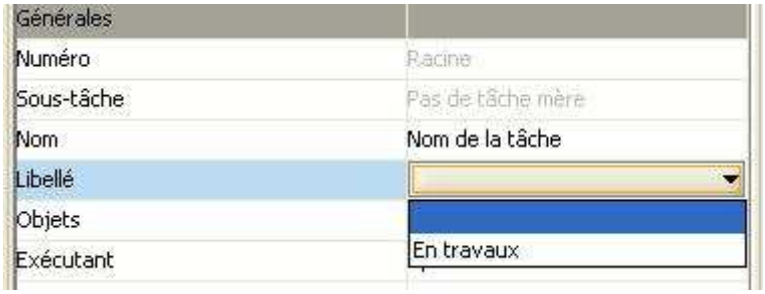

Figure 56 - Caractéristique pour associer un libellé à une tâche

5 - Comment activer la couleur de toutes les tâches associées à des libellés ?

K-MADe fournit une option dans l'espace de tâches qui permet d'activer ou pas la colorisation des tâches associées à des libellés. Cette option est globale et revient à mettre « true » à l'attribut *CouleurVisible* de tous les libellés.

- Activer ou pas le bouton « Libellé Couleur » à partir de la boîte à outils disponible dans l'espace de tâches (Figure 57).

Figure 57 – Outil pour activer la couleur de l'ensemble des libellés

Sur la Figure 58 nous présentons un arbre graphique dont certaines tâches ont été associées à des libellés (celles qui présentent une couleur). L'action afficher les couleurs des libellés est activée.

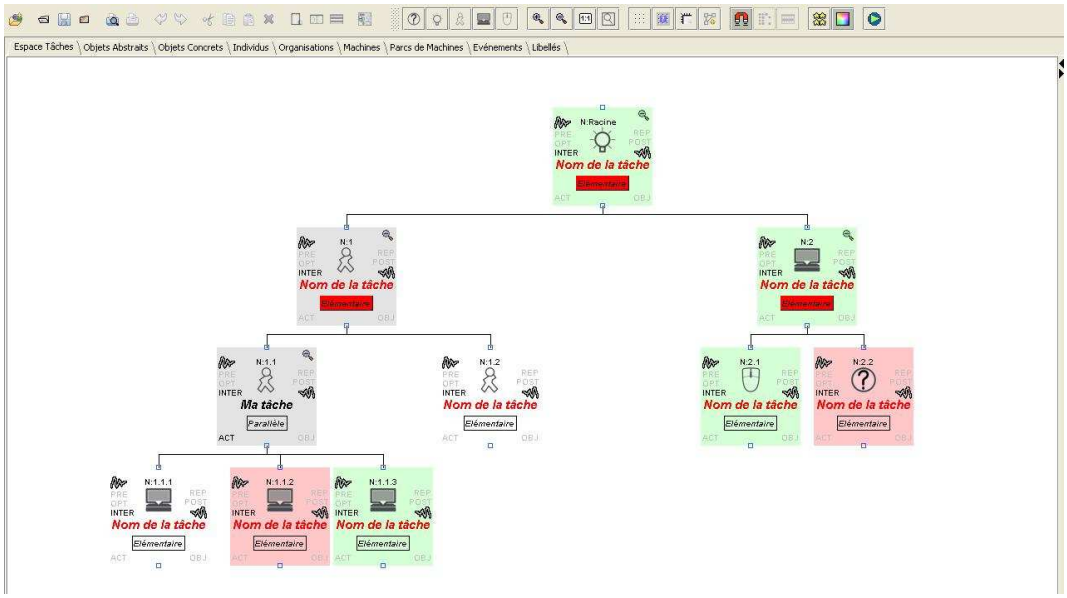

Figure 58 – Activation de l'ensemble des couleurs des libellés.

Au contraire sur la Figure 59, l'action « Libellé Couleur » est désactivée. Par conséquent les tâches ne sont pas colorisées par les libellés.

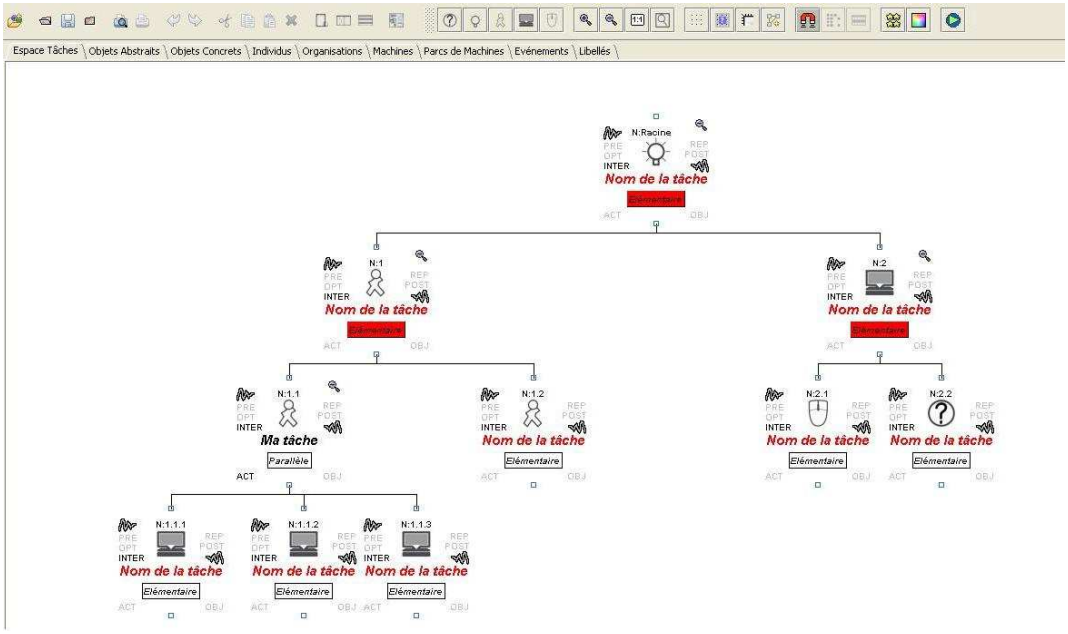

Figure 59 – Désactivation de l'ensemble des couleurs des libellés

6 - Comment activer la visibilité de toutes les tâches associées à des libellés ?

K-MADe fournit une option dans l'espace de tâches qui permet d'afficher ou pas les tâches associées à des libellés. Cette option est globale et revient à mettre « true » à l'attribut *Visible* de tous les libellés.

- Activer ou pas le bouton « Libellé Visible » à partir de la boîte à outils disponible dans l'espace de tâches.

Figure 60 – Outil pour activer la visibilité de l'ensemble des libellés

C - Relation ente tâches et objets de l'état du monde

Cette partie s'intéresse à la construction des expressions pour les caractéristiques suivantes : précondition, effet de bords et itération.

Rappelons que pour la précondition, il s'agit d'une condition qui doit être vraie pour exécuter une tâche. Dans le cas des actions, il s'agit d'une expression qui permet la modification des objets concrets de l'état du monde.

Enfin, l'itération est une expression permettant de rendre une tâche itérative de manière à répéter plusieurs fois une tâche.

Nous commençons par présenter l'outil de construction des expressions de manière générale. En effet, cet outil propose un mode de fonctionnement général identique aux trois types d'expressions.

Enfin nous détaillons pour chacun des types d'expressions les différentes grammaires utilisées en donnant si possible le plus d'exemples possibles.

C.1 - Création/Edition d'une expression

Concept : Expression de manière générale

Les expressions dans le modèle K-MAD permettent de construire les caractéristiques des préconditions, action et itérations. Pour chaque type de caractéristique, une grammaire est disponible.

K-MADe distingue deux informations à renseigner pour les expressions des caractéristiques de tâches :

- La description « textuelle » de l'expression ;
- L'expression « évaluable ».

Information « textuelle » : elle permet à l'analyste de donner dans un langage naturel une description concernant l'expression « évaluable » qu'il souhaite construire.

Descriptif / Exigences : texte, facultatif

Exemple : Existe-il plus de deux missions de type conférences qui débutent après le 20-05-2006 ?

Expression « évaluable » : il s'agit d'une expression construite à partir d'une grammaire et qui peut être évaluée. En d'autres termes, il est possible de vérifier la syntaxe de l'expression et d'en sortir un résultat qui peut être pour une:

- précondition, *true* ou *false* ;
- itération, *true* ou *false* s'il s'agit d'un prédicat, sinon il s'agit d'un variant un entier ;
- action, il n'y a pas de résultat mais une modification de l'état du monde.

Concernant la vérification d'une expression, elle consiste à vérifier qu'il n'existe pas d'erreur dans une expression. Nous trouvons trois types d'erreur :

- Les erreurs lexical : désignent les erreurs dans un nom d'une expression ;
- Les erreurs syntaxiques : désignent les erreurs suite à une incohérence dans la grammaire d'une expression ;
- Les erreurs sémantiques : désignent les erreurs concernant d'une part les problèmes dans les objets issus de l'état du monde et d'autre part des problèmes de typage.

Descriptif / Exigences : expression, obligatoire

Exemple : card($Conference, $departDate > 20052006) > 2. Cette expression fait suite à la description donnée précédemment.

Concept : Expression de type « Littéral »

Les « Littéraux » désignent des expressions de types valeurs constantes et valeurs utilisateurs. Nous avons vu dans la partie liée aux attributs abstraits que K-MAD ne manipulait que des types simples. Par conséquent les valeurs constantes ne sont que des types simples.

Valeurs constantes : les types disponibles sont

- Booléen ;
- Nombre ;
- Texte.

Descriptif / Exigences :

- Booléen : Enumération = {true, false} ;

- Nombre : Ensemble des nombres : entiers naturels et nombres décimaux ;
- Texte : encadré par « ' »

Exemple : 12, true, 'rouge'

Valeurs utilisateurs : il s'agit de valeurs de types simples qui sont initialisées par l'analyste. Les valeurs sont données par l'utilisateur au moment de l'évaluation d'une expression (par exemple pendant la simulation d'un arbre K-MAD). Les types disponibles sont :

- Booléen ;
- Nombre ;
- Texte.

Descriptif / Exigences :

- ?BOOL: valeur utilisateur concernant un booléen ;
- ?NUM : valeur utilisateur concernant un nombre ;
- ?STR : valeur utilisateur concernant un texte.

Exemple : 12.5 < ?NUM

Concept : Expression de type « Opérateur »

Les expressions « Opérateurs » désignent des opérateurs utilisés pour des affectations, pour des comparaisons ou pour des opérations conditionnelles. Leurs disponibilités dépendent du type de caractéristiques éditées.

Nous reviendrons sur les opérateurs pendant la présentation des expressions pour les préconditions, action et itérations.

Concept : Expression de type « Fonction »

Les fonctions permettent de manipuler les objets concrets de l'état du monde. Elles sont différentes selon le type de la caractéristique éditée (précondition, action et itération). Par ailleurs, les fonctions ont une syntaxe assez simple. Elles possèdent au moins un paramètre et certaines fonctions peuvent en posséder au grand maximum deux.

De façon générale, les fonctions manipulent les objets concrets par l'intermédiaire des groupes et des attributs abstraits donnés en paramètre (rappelons qu'un groupe connaît son objet abstrait).

Nous pouvons à ce niveau distinguer deux types de fonctions :

- Les fonctions qui s'appliquent sur l'ensemble des objets concrets d'un groupe que nous appellerons fonctions globales;
- Les fonctions qui s'appliquent sur un seul objet concret que nous appellerons fonctions locales.

Concernant les fonctions dites « locales », l'objet concret peut être obtenu par le type du groupe donné en paramètre de la fonction. Si le groupe est

- une liste : c'est l'objet concret le plus ancien ;
- une pile : c'est l'objet concret le plus récent ;
- unique (singleton) : c'est l'objet lui-même ;
- un ensemble : l'objet concret est choisi par l'analyste.
- un tableau : l'emplacement (numéro de la case du tableau) est choisi par l'analyste

Il existe une autre façon d'obtenir l'objet concret pour certaines fonctions dites « locales » en exploitant un « buffer ». Nous insisterons sur ces aspects lors des présentations des fonctions pour les préconditions, action et itérations.

***Note** : les fonctions peuvent pointer vers des groupes et des attributs abstraits. Pour identifier les groupes et les attributs, leurs noms sont précédés du caractères $. Exemple :* $conférence (groupe conférence), $date_de_depart (attribut abstrait date de départ).

L'outil K-MADe s'appuie sur chaque grammaire des expressions (précondition, action et itération) pour y vérifier la cohérence des expressions saisies par l'utilisateur.

1 - Comment atteindre l'outil de construction d'expressions ?

Selon si une précondition, action ou une itération est éditée la démarche de modification consiste à :

- Sélectionner une tâche ;
- Éditer la caractéristique souhaitée (précondition, action et itération).

Figure 61 – Caractéristiques pour exprimer la précondition, l'itération et les action d'une tâche

OU

- Dans l'espace de tâches, double cliquer sur un des icônes indiqués dans la Figure 62 dans le cadre de la tâche(Repère 1 : préconditions ; Repère 2 : Itérations ; Repère 3 : Action) .

Figure 62 – Caractéristiques pour exprimer la précondition, l'itération et les action d'une tâche

2 - Présentation de l'éditeur d'expressions en général

L'éditeur d'expressions présente deux grandes zones (voir Figure 63).

La première située à droite liste (repère 4) l'ensemble des objets abstraits de l'état du monde. Cette interface permet de parcourir les objets abstraits (groupes et attributs abstraits associés) de manière à les retrouver plus efficacement pour construire une expression.

La seconde située sur la partie centrale propose une interface permettant d'éditer une expression (repère 1, 2 et 3). Plus précisément nous y trouvons :

- Une partie haute pour l'édition, la vérification et l'évaluation de l'expression en construction (repère 1);
- Une partie centrale appelée **calculatrice** fournissant à l'utilisateur un ensemble de fonctions, d'opérateurs et de littéraux (repère 2);
- Une partie basse pour l'affichage des messages d'états et d'erreurs (repère 3).

Toutes ces interfaces seront détaillées en profondeur dans la suite de ce document.

***Note** : l'interface présentée sur la Figure 63 est issue de l'éditeur d'expression de la caractéristique **Précondition**. Les interfaces concernant les caractéristiques **Action** et **Itération** conservent un agencement des composants graphiques assez proche. Nous les étudierons en détail par la suite.*

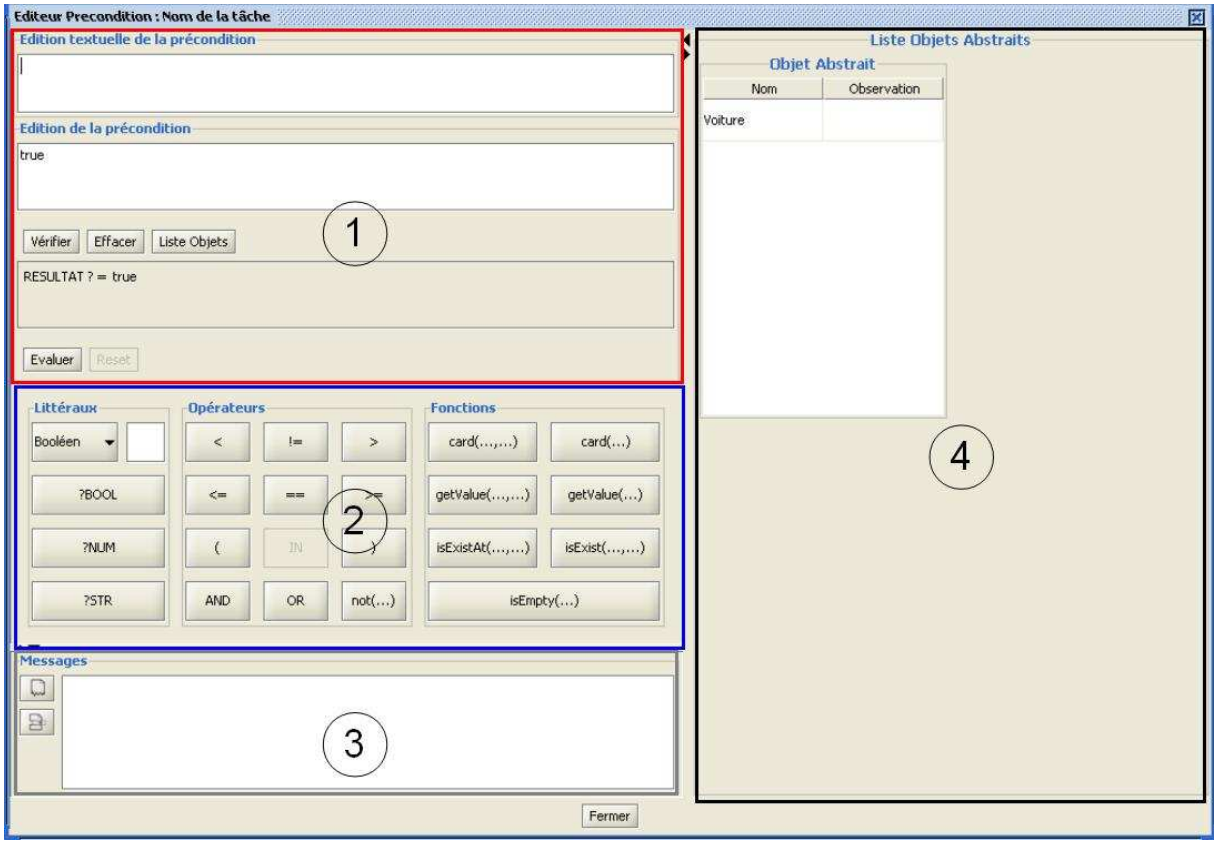

Figure 63 – Outil pour l'édition des expressions de précondition, action et itération..

3 - Quelles sont les fonctionnalités de la calculatrice ?

La calculatrice fournit un ensemble de fonctions, d'opérateurs et de littéraux et dont l'interface de la calculatrice évolue en fonction du type de caractéristique (précondition, action et itération) à éditer. Elle permet également de :

- Vérifier la syntaxe d'une expression en concordance avec la grammaire de la caractéristique éditée ;

- Évaluer l'expression (précondition et itération).

Ou

- Modifier les objets de l'état du monde (action).

4 - Comment éditer une expression d'une caractéristique de tâche ?

L'édition va consister à éditer la description « textuelle » et l'expression « évaluable » (voir Figure 64).

Pour la description « textuelle » :

- Editer directement la zone de texte (repère 1).

***Note 1** : rappelons que la description « textuelle » est une information facultative.*

Concernant l'expression « évaluable » :

- Editer directement l'expression dans la zone d'édition (repère 2).

Ou

- Cliquer sur les éléments de la calculatrice pour construire une expression (repère 3).

Pour supprimer des éléments de l'expression, vous devrez la modifier directement dans la zone d'édition (repère 2).

***Note 2**: K-MADe n'est pas sensible à la casse. Le texte suivant « Conference » est égale à « CONFERENCE ».*

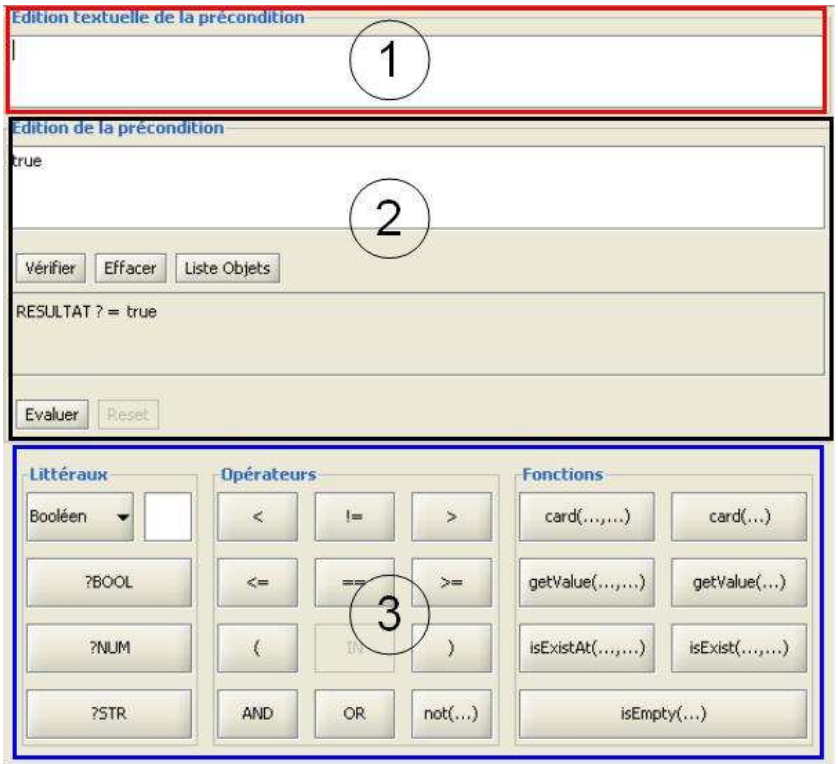

Figure 64 – Outil pour la saisie de la description textuelle et de la construction de l'expression « évaluable ».

5 - Comment vérifier une expression ?

Rappelons que la vérification d'une expression consiste à contrôler qu'il n'existe pas d'erreur dans une expression. Trois types d'erreur sont à distinguer :

- Les erreurs lexicales : désignent les erreurs dans un nom d'une expression ;
- Les erreurs syntaxiques : désignent les erreurs suite à une incohérence dans la grammaire d'une expression ;
- Les erreurs sémantiques : désignent les erreurs concernant d'une part les problèmes dans les objets issus de l'état du monde et d'autre part des problèmes de typage.

Pour vérifier une expression :

- Cliquer sur le bouton « Vérifier » (voir Figure 65). Les messages d'erreur sont affichés dans la partie basse de l'outil d'édition (repère 1 de la Figure 65).

Exemple :

- isEmpy($conferences) => erreur syntaxique ;
- 12 12 => erreur syntaxique ;
- 12 < 'd' => erreur sémantique, problème de typage pour la décomposition de comparaison ;
- isEmpty($vacances) => erreur sémantique, le groupe vacances n'existe pas (voir Figure 65).

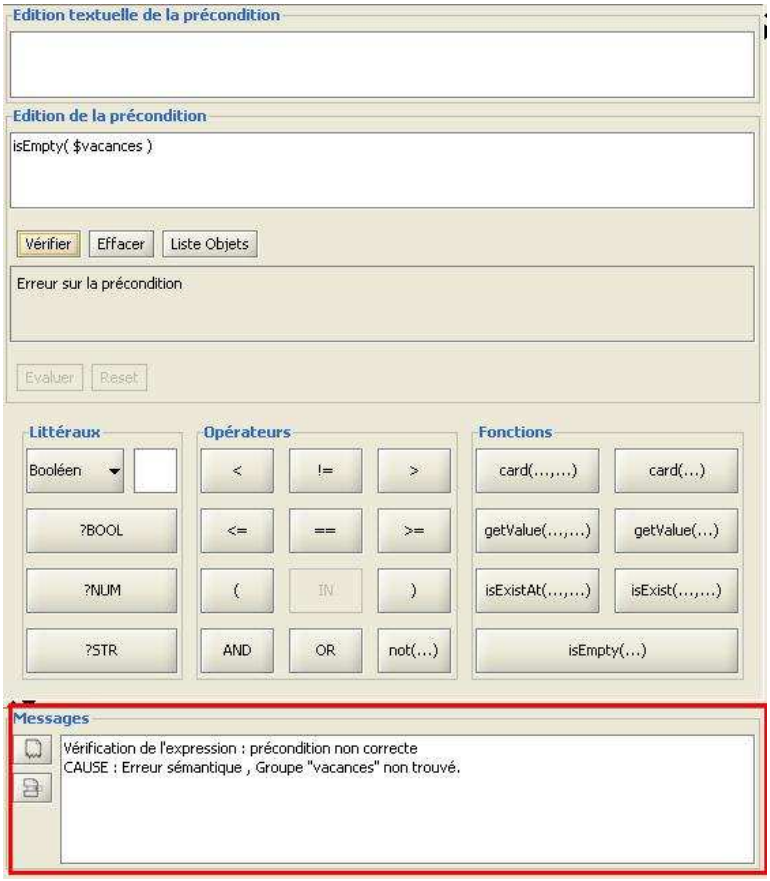

Figure 65 – Exemple d'erreur sémantique. Le groupe « vacances »

6 - Comment enregistrer une expression dans une caractéristique de tâche (pré/post condition ou itération) ?

Suite à une édition d'une expression, son enregistrement n'est effectif qu'après vérification. Le bouton vérification joue le rôle d'enregistrement de l'expression dans la caractéristique.

- Éditer une expression ;
- Cliquer sur le bouton « Vérifier » ;
- Cliquer sur le bouton « Retour » .

***Note** : si une expression est vérifiée comme incorrecte, elle est également enregistrée dans la caractéristique. Notons également que certains services (par ex : simulation) qui exploitent les expressions afin de les évaluer seront accessibles seulement si toutes les expressions sont vérifiées comme correctes.*

7 - Comment initialiser une expression selon le type de caractéristique ?

L'initialisation consiste à effacer le contenu de l'expression en cours d'édition et de placer une valeur par défaut qui dépend du type de la caractéristique (« true » pour une précondition, « void » pour un action et [1] pour une itération)

- Cliquer sur le bouton « Effacer » (Figure 66).

Figure 66 – Action pour initialiser une expression

8 - Comment évaluer une expression ?

L'évaluation consiste à déterminer le résultat effectif d'une expression (un booléen pour la précondition, un booléen ou un entier pour l'itération, une modification de l'état du monde pour les actions).

Pour évaluer une expression :

- Éditer l'expression ;
- Vérifier l'expression (bouton « Vérifier ») ;
- Cliquer sur le bouton « Evaluer ».

A la suite de la vérification, l'expression est placée dans la zone d'évaluation (repère 1 de la Figure 67). L'expression affichée dans cette zone contient des informations supplémentaires comme :

- Des listes des objets concrets pour les fonctions exploitant une groupe ;
- Des champs de texte pour saisir les valeurs utilisateurs.

***Note 1** : l'évaluation ne peut être réalisée que si l'expression a été vérifiée positive.*

***Note 2** : l'évaluation ne peut être réalisée que si toutes les valeurs utilisateurs et/ou tous les objets concrets ont été précisés par l'utilisateur. En effet, une expression peut contenir des valeurs utilisateurs (?NUM, ?BOOL et ?STR) et/ou des objets concrets (un groupe dont le type est un ensemble ou un tableau).*

Figure 67 – Action pour évaluer une expression

9 - Comment choisir un groupe ?

Si une fonction nécessite l'utilisation d'un groupe comme paramètre, vous devez :

- Sélectionner sur la zone de droite (repère 1 de la Figure 68) un objet abstrait ;
- Positionner le curseur dans la zone d'édition de l'expression à l'endroit où vous souhaitez ajouter le nom du groupe ;

Puis

- Sélectionner un groupe (repère 2 de la Figure 68) ;
- Double cliquer sur le groupe sélectionné (le groupe est automatiquement ajouté dans la zone d'édition de l'expression).

Ou

- Saisir directement le nom du groupe dans la zone d'édition (**ne pas oublier le caractère $**).

Figure 68 – Sélection d'un groupe

10 - Comment choisir un attribut abstrait ?

Si une fonction nécessite l'utilisation d'un attribut abstrait comme paramètre, vous devez :

- Positionner le curseur dans la zone d'édition de l'expression à l'endroit où le nom de l'attribut abstrait doit être ajouté ;
- Sélectionner sur la zone de droite (repère 1 la Figure 68) un objet abstrait ;

Puis

- Sélectionner un attribut abstrait (repère 3 la Figure 68) ;

- Double cliquer sur l'attribut abstrait sélectionné (l'attribut abstrait est automatiquement ajouté dans la zone d'édition de l'expression).

Ou

- Saisir directement le nom de l'attribut abstrait dans la zone d'édition (**ne pas oublier le caractère $**).

11 - Comment saisir des valeurs utilisateurs dans une expression ?

Si une expression contient des valeurs utilisateurs, des champs de textes sont éditables dans la zone d'évaluation. L'édition consiste :

- Saisir une valeur (dépendant du type de la valeur utilisateur) pour tous les champs de texte présents dans la zone d'évaluation (Figure 69).

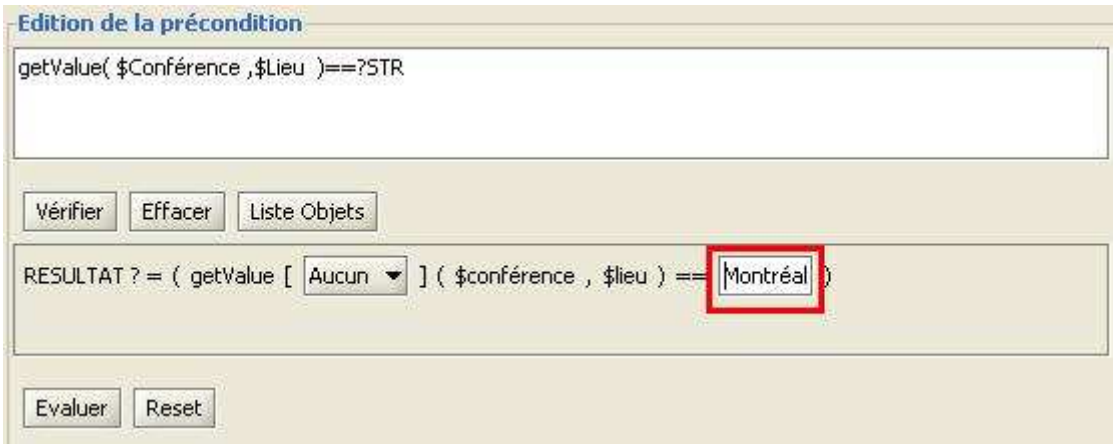

Figure 69 – Expression contenant des valeurs utilisateurs à saisir pour l'évaluation.

***Note** : si des valeurs utilisateurs sont manquantes, l'expression n'est pas évaluée.*

12 - Comment initialiser les valeurs utilisateurs saisies ?

L'initialisation des valeurs utilisateurs consistent à supprimer toutes les valeurs saisies de l'expression contenue dans la zone d'évaluation.

- Cliquer sur le bouton « Reset » (Figure 70).

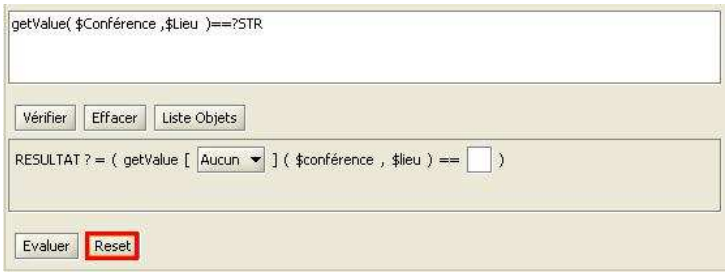

Figure 70 – Action pour initialiser les valeurs utilisateur

13 - Comment choisir des objets concrets pour une fonction donnée ?

Si une expression contient des fonctions qui font appel à des groupes dont le type est ensemble ou tableau, l'utilisateur doit effectuer un choix dans la liste proposé : pour un ensemble, la liste est composée par les noms des objets concrets, pour un tableau la liste est composée par le numéro des emplacements. L'édition consiste :

- Sélectionner un des choix possibles.

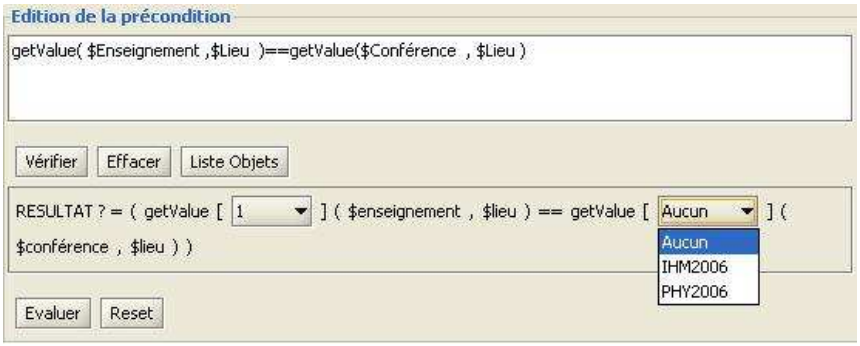

Figure 71 – Expression contenant des objets concrets à sélectionner.

C.2 - Édition de la caractéristique « précondition »

Concept : Expression « précondition »

La caractéristique précondition est une condition qui doit être vraie pour exécuter une tâche. Cette caractéristique est représentée par une expression qui retourne un booléen (soit *true* soit *false*). Une expression ne retournant pas un booléen est considérée comme fausse.

Une expression « précondition » est construite à partir de valeurs constantes (ceux expliquées dans la partie C.1), de valeurs utilisateurs (également expliquées dans la partie C.1), d'opérateurs et de fonctions.

Concept : Opérateurs disponibles pour une expression « précondition »

Nous distinguons trois familles d'opérateurs :

- Les opérateurs de comparaison ;
- Les opérateurs « unaires » : parenthèse et négation ;
- Les opérateurs logiques.

Opérateurs de comparaison : ces opérateurs retournent un booléen. Ils comparent un membre de gauche avec un membre de droite : exp_1 *op* exp_2 où *op* peut être <, >, <=, >=, *!*= ou ==.

Descriptif / Exigences :

- Booléen : <, >, <= et >= : exp_1 et exp_2 doivent être du même type et non booléén. Si exp_1 et exp_2 sont de type texte, c'est l'ordre lexicographique (sans prendre en compte les majuscules) qui est pris en compte ;
- Booléen : != et == : exp_1 et exp_2 peuvent être de type booléen, texte et entier. exp_1 et exp_2 doivent être du même type.

Exemple : 12 < 45 ; true != false, 'Conference' < 'vacances'

Opérateurs « unaires » : ce sont en fait des fonctions unaires que nous avons placé à ce niveau car elles ne fonctionnaient pas sur le même principe que les fonctions manipulant des objets de l'état du monde.

A la différence des autres opérateurs, la fonction parenthèse peut retourner autre chose qu'une expression de type booléenne.

Descriptif / Exigences :

- Entier, Booléen ou Texte : () : elle concerne les parenthèses permettant de regrouper des expressions ;
- Booléen : NOT(Booléen param) : cette fonction permet de retourner le complément de l'expression *param*. **Le type de *param* doit être un booléen.**

Exemple : (12 < 13), NOT(12 < 13)

Opérateurs logiques : ces opérateurs correspondent aux opérateurs de condition. Ils retournent un booléen et effectuent une condition entre un membre de gauche et un membre de droite : *exp_1 op exp_2* où *op* est soit *AND* soit *OR* et où exp_1 *et* exp_2 *sont des expressions de types booléennes*

Descriptif / Exigences :

- Booléen : AND : *exp_1 AND exp_2* . L'expression de gauche et de droite doivent avoir la même valeur ;
- Booléen : OR : *exp_1 OR exp_2*. Au moins une expression parmi les membres de gauche et de droite doit être vraie.

Exemple : (12 > ?NUM) AND (12 < ?NUM)

Concept : Fonctions disponibles pour une expression « précondition »

Sept fonctions sont proposées pour construire une expression « précondition ». Leurs fonctionnalités et leurs syntaxes sont différentes :

- Nombre card(Groupe, <condition logique>) ;
- Nombre card(Groupe) ;
- Nombre, Texte ou Booléen getValue(Groupe, Attribut Abstrait) ;
- Nombre, Texte ou Booléen getValue(Attribut Abstrait) ;
- Booléen isExistAt(Groupe, <condition logique>) ;
- isExist(Groupe,<condition logique>) ;
- isEmpty(Groupe).

Nombre card(Groupe g, <condition logique> cl) : cette fonction retourne le nombre d'objet concret contenu dans le groupe *g* et qui satisfait la condition logique *cl*. Cette condition logique est une expression construite à l'aide d'opérateurs logiques et d'opérateurs de comparaison sur les attributs abstraits de l'objet abstrait considéré (nous avons vu précédemment qu'un groupe connaissait son objet abstrait). Dans ce sens *cl* **doit retourner une expression de type booléenne et doit contenir un ou plusieurs attributs abstraits**.

Descriptif / Exigences : Nombre card(Groupe g, <condition logique> cl), fonction globale

Exemple : card($Conference, $departDate > 20052006) > 2 : existe-il plus de deux missions de type conférences qui débutent après le 20-05-2006 ?

***Note** : cette fonction ne peut être utilisée seule dans une précondition. En effet, cette fonction retourne un entier alors que l'expression de la précondition doit retourner une valeur booléenne.*

Nombre card(Groupe g) : cette fonction retourne le nombre d'objet concret contenu dans le groupe *g*. Contrairement à la fonction précédente il n'y a pas de condition logique.

Descriptif / Exigences : Entier card(Groupe g), fonction globale

Exemple : card($Conference) > 0 : existe-il des missions de type conférence ?

***Note** : les fonctions peuvent se combiner sous couvert du respect du typage de l'opérateur.* (card($*Conference*) == card($Conference, $departDate > 20052006)) AND (card($Conference) > 1)*. Toutes les missions de type conférence satisfont le prédicat* $departDate > 20052006*.*

Nombre, Texte ou Booléen getValue(Groupe g, AttributAbstrait aa) : cette fonction retourne la valeur de l'attribut concret associé à l'attribut abstrait *aa*. Le type de retour de la fonction dépend du type de l'attribut abstrait *aa.* Il s'agit d'une fonction « locale » est l'objet concret est déterminé par le type du groupe *g*.

Descriptif / Exigences : Nombre, Texte ou Booléen getValue(Groupe g, AttributAbstrait aa), fonction locale

Exemple : getValue($Conference, $isInEurope) retourne la valeur booléenne de l'attribut concret *isInEurope*.

***Note 1** : si getValue(…) retourne un booléen cette fonction pourra être utilisée seule dans l'expression de la précondition. Au contraire si la fonction retourne un entier ou du texte, elle ne pourra pas être utilisée seule.*

Nombre, Texte ou Booléen getValue(AttributAbstrait aa) : cette fonction retourne la valeur de l'attribut concret associé à l'attribut abstrait *aa*. Par ailleurs cette fonction ne peut être utilisée qu'à l'intérieur d'une autre fonction. Ainsi, l'attribut abstrait *aa* fait référence à l'objet abstrait défini par le groupe de la fonction englobante. De plus, il s'agit d'une fonction locale et dont l'objet concret est obtenu automatiquement de la fonction englobante.

Descriptif / Exigences : Entier, Texte ou Booléen getValue(AttributAbstrait aa), fonction locale

Exemple : card($Conference, $getValue($departDate) > 20052006). Cette expression donne le même résultat que l'exemple donné pour la fonction *card(Groupe g)*. Le groupe de la fonction *getValue(…)* est donc *$Conference*.

Booléen isExistAt(Groupe g, <condition logique> cl) : cette fonction vérifie s'il existe dans le groupe *g* un objet concret qui satisfait la condition logique *cl*. La condition logique est une expression construite à partir d'opérateurs logiques, d'opérateurs de comparaison et d'attributs abstraits issus de l'objet abstrait référencé par *g*.

Descriptif / Exigences : Booléen isExistAt(Groupe g, < condition logique>), fonction locale

Exemple : isExistAt($Conference, departDate == 20052006). Existe-il un objet concret (choisi selon le type du groupe *$Conference*) dont l'attribut abstrait dapartDate vaut 20052006.

Booléen isExist(Groupe g, <condition logique> cl) : cette fonction vérifie s'il existe dans le groupe *g* au moins un objet concret qui satisfait la condition logique *cl*. À la différence de la fonction *isExistAt(…)* cette fonction s'intéresse à l'ensemble des objets concrets contenu dans le groupe *g*, il s'agit d'une fonction globale. La condition logique est une expression construite à partir d'opérateurs logiques, d'opérateurs de comparaison et d'attributs abstraits issus de l'objet abstrait référencé par *g*.

Descriptif / Exigences : Booléen isExist(Groupe g, Prédicat p), fonction globale

Exemple : isExist($Conference, departDate == 20052006). Existe-il au moins un objet concret dont *departDate* vaut 20052006 dans le groupe *Conference*.

Booléen isEmpty(Groupe g) : cette fonction vérifie si le groupe *g* est vide ou pas, c'est-à-dire si *g* contient ou pas des objets concrets.

Descriptif / Exigences *:* Booléen isEmpty(Groupe g), fonction globale

Exemple : isEmpty($Conference). Vérifie si le groupe *Conference* contient des objets concrets ou pas.

1 - Description de l'outil d'édition d'une expression précondition

Sur la Figure 72 est présentée l'outil d'édition d'une expression précondition. Nous retrouvons les mêmes éléments que ceux présentés dans la partie précédente.

Les opérateurs sont désignés par le repère 1 et les fonctions par le repère 2.

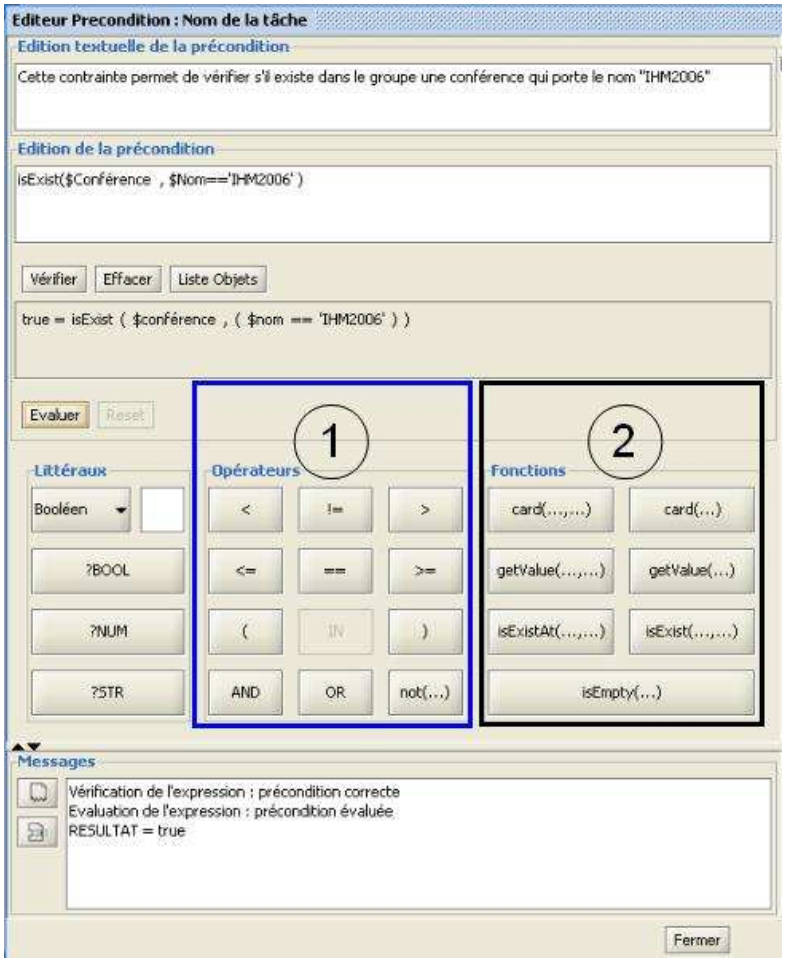

Figure 72 – Outil pour la construction d'une expression précondition

C.3 - Édition d' Action de bord de la tâche

Concept : Expression « action »

L'expression « action » d'une tâche permet d'effectuer des modifications sur les objets de l'état du monde. Plus concrètement, elle permet de modifier/créer et supprimer des objets concrets et modifier les valeurs des attributs concrets.

Une expression « action » est construite à partir de valeurs constantes (ceux expliquées dans la partie C.1), de valeurs utilisateurs (également expliquées dans la partie C.1), d'opérateurs et de fonctions.

Concept : Opérateurs disponibles pour une expression « action »

Nous distinguons trois familles d'opérateurs :

- Les opérateurs d'affectation ;
- Les opérateurs arithmétiques ;
- L'opérateur séquence.

Opérateurs d'affectation : ces opérateurs ne retournent pas de valeur. Ils affectent au membre de gauche une valeur déterminée par le résultat de l'opérateur : exp_1 op exp_2 où op peut être :=, += ou -=.

Descriptif / Exigences :

- := : il s'agit d'une simple affectation. exp_1 et exp_2 doivent être de même type. Le type de *op* est déterminé par le type d'exp_1. Ainsi si exp_1 est de type Texte, le membre de droite exp_2 doit être un Texte (K-MADe retournera une erreur si les types ne sont pas corrects).
- += : il s'agit d'une affectation combinée à une addition entre la valeur précédente de exp_1 et la valeur de exp_2. exp_1 et exp_2 peuvent être de type différent (Nombre ou Texte). L'opération ajoute exp_1 à exp_2 s'il s'agit de nombre. S'il s'agit de texte, elle les concatène.
- -= : il s'agit d'une affectation combinée à une soustraction entre la valeur précédente de exp_1 et la valeur de exp_2. exp_1 et exp_2 doivent être de type Nombre.

Exemple : set($Conference, $name := 'ISOLA'). Modification de l'attribut *$name* par la valeur 'ISOLA'. La fonction *set(…,…)* sera étudiée par la suite.

Opérateurs arithmétiques : ces opérateurs de calcul retournent une valeur. Ils effectuent une opération entre le membre de gauche et le membre de droite.

Descriptif / Exigences :

- + : il s'agit de l'opérateur d'addition ou de concaténation. Les membres peuvent être de type (Nombre ou Texte). Si le membre exp_1 et exp_2 sont des textes, l'opérateur retourne une concaténation de texte. Par contre si exp_1 et exp_2 sont des nombres, l'opérateur retourne la somme de exp_1 et exp_2.
- - : il s'agit de l'opérateur de soustraction. Seuls les membres de type nombre sont autorisés.

Exemple : set($Conference, $name := 'ISOLA' + '2007'). Modification de l'attribut *$name* par la valeur 'ISOLA2007'

Opérateur séquence : cet opérateur désigne la séquence. Il permet d'enchaîner plusieurs fonctions dans la même expression de la postcondition. Il ne retourne pas de valeur.

Descriptif / Exigences : ; : permet d'enchaîner plusieurs fonctions

Exemple : set($Conference, $name := 'ISOLA') ; set($name := 'ISOLA2007'). Appel séquentiel des fonctions *set(…,…)* et *set(…)*.

Concept : Fonctions disponibles pour une expression « action »

Huit fonctions sont proposées pour construire une expression « action ». Leurs fonctionnalités et leurs syntaxes sont différentes :

- ObjetConcret add(Groupe) ;
- ObjetConcret create(Groupe, Texte) ;
- ObjetConcret remove(Groupe) ;
- ObjetConcret replace(Groupe, Groupe) ;
- set(Groupe,<affectation>) ;
- set(<affectation>…) ;
- getValue(Groupe,Attribut Abstrait) ;
- getValue(Attribut Abstrait).

Les fonctions *getValue(Groupe, Attribut Abstrait)* et *getValue(Attribut Abstrait)* ont une utilisation identique à celles présentées dans la partie expression « précondition ».

Par ailleurs les fonctions permettant de construire les expressions « action » se distinguent des fonctions précondition et itération (voir plus tard) par la capacité à retourner au modèle K-MAD l'objet concret manipulé. L'objet concret est stocké dans un buffer du modèle K-MAD. L'avantage d'une telle solution est de pouvoir enchaîner des fonctions en exploitant le même objet concret.

Ainsi les fonctions dites « locale » pour construire les expressions « action » utilisent l'objet concret stocké dans un buffer du modèle K-MAD.

Enfin, précisons que pour chaque appel d'une fonction modifiant l'état des objets du modèle K-MAD (*add*, *create*, *remove*, *replace*, *set*) le modèle effectue une sauvegarde de l'état précédent.

***Note 1** : l'objet concret retourné par une fonction est stocké dans un buffer du modèle K-MAD. Il ne s'agit pas d'une liste ni d'une pile mais d'un type unique (singleton). Ainsi, lorsqu'une fonction retourne un objet concret, K-MAD supprime le précédent objet concret et stock le nouveau dans le buffer.*

***Note 2** : n'utilisez pas une fonction « locale » si le buffer d'objet concret du modèle K-MAD est vide.*

ObjetConcret add(Groupe g) : cette fonction permet d'ajouter l'objet concret, stocké dans le buffer du modèle K-MADe, dans le groupe *g***.** Cette fonction retourne également l'objet concret manipulé, c'est-à-dire le même qui vient d'être ajouté dans le groupe *g*.

Descriptif / Exigences : ObjetConcret add(Groupe g), fonction locale

ObjetConcret create(Groupe g, Texte t) : cette fonction crée et ajoute un objet concret dans le groupe *g*. Le nom du nouvel objet concret est donné par la chaîne *t*. Cette fonction retourne l'objet concret nouvellement créé au modèle K-MAD.

Descriptif / Exigences : ObjetConcret create(Groupe g ; Texte t), fonction locale

ObjetConcret remove(Groupe g) : cette fonction supprime un objet concret contenu dans le groupe *g* et le retourne au modèle K-MAD. L'objet concret à supprimer est choisi en fonction du type du groupe *g*.

Descriptif / Exigences : ObjetConcret remove(Groupe g), fonction locale

ObjetConcret replace(Groupe g_1, Groupe g_2) : cette fonction supprime l'objet concret g_1 et l'ajoute dans le groupe g_2. Cette fonction retourne l'objet concret transféré. L'objet concret à déplacer est choisi en fonction du type du groupe g_1.

Descriptif / Exigences : ObjetConcret replace(Groupe g_1, Goupe g_2), fonction locale

ObjetConcret set(Groupe g, <affectation> a) : cette fonction modifie la valeur d'un attribut concret exprimé dans l'affectation *a*. Cette expression doit être une affectation (:=, += et -=) dont le membre de gauche est un attribut abstrait et le membre de droite une expression qui retourne une valeur. Cette fonction retourne l'objet concret qui a été modifié.

Descriptif / Exigences : ObjetConcret set(Groupe g, <affectation> a). a ::= AttributAbstrait (:= | += | -=) Expression où Expression doit retourner une valeur.

Exemple : set($Conference, $name := 'ISOLA'). Modification de l'attribut *$name* par la valeur 'ISOLA' où 'ISOLA' est une constante texte. L'objet concret est choisi par l'utilisateur.

ObjetConcret set(<affectation> a) : cette fonction modifie également la valeur d'un attribut concret exprimé dans l'expression e. Cette expression doit suivre la grammaire décrite dans la fonction précédente. L'attribut concret modifié appartient à l'objet concret stocké dans le buffer du modèle K-MAD. Cette fonction retourne l'objet concret qui a été modifié.

Descriptif / Exigences : ObjetConcret set(<affectation> a) avec « e ::= AttributAbstrait (:= | += | -=) Expression » où Expression doit retourner une valeur.

Exemple : set($name := 'IHM2007'). Modification de l'attribut *$name* par la valeur *'IHM2007'*. L'objet concret est celui qui a été retourné par la précédente fonction.

1 - Description de l'outil d'édition d'une expression Action

Sur la Figure 73 est présentée l'outil d'édition d'une expression action. Les opérateurs sont désignés par le repère 1 tandis que les fonctions sont désignées par le repère 2.

Le repère 3 permet de visualiser l'historique des modifications du modèle K-MAD suite à un appel d'une fonction. Enfin le repère 4 affiche l'objet concret courant stocké dans le buffer de K-MAD.

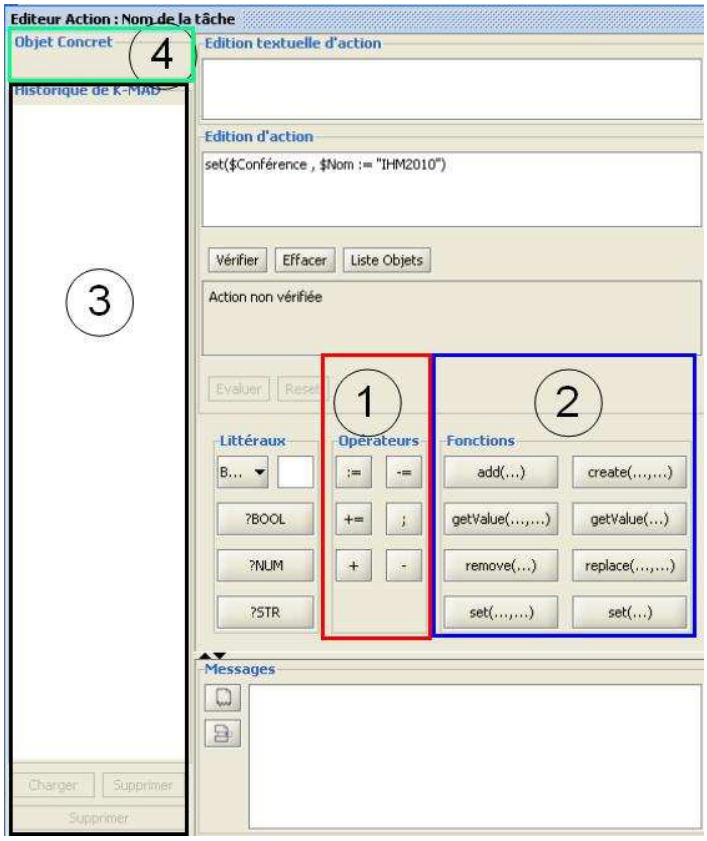

Figure 73 – Outil pour la construction d'une expression action

2 - Charger et supprimer un état de l'historique du modèle K-MAD

EN COURS DE DEVELOPPEMENT

C.4 - Édition d'une itération de tâche

Concept : Expression « Itération »

L'itération représente le caractère répétitif d'une tâche puisqu'une tâche peut être exécutée plusieurs fois. Par exemple pour les missions, il peut s'agir du traitement à posteriori des conférences. La tâche de traitement de retour de mission sera répétée autant de fois qu'il y a de conférences terminées et dans lesquelles l'agent a participées.

L'expression modélisant une itération peut être construite suivant deux formes :

- Un variant qui décroît à chaque exécution de la tâche ;
- Un prédicat qui exprime le fait de continuer ou pas l'itération.

Nous ne présenterons pas les opérateurs utilisés dans la construction d'une itération car il s'agit des mêmes que ceux de l'expression précondition.

***Note** : il est impossible de combiner les deux formes : variant et prédicat.*

Variant : le variant dans une expression itération est un entier qui décroît jusqu'à zéro à chaque répétition d'une tâche.

Descriptif / Exigences : [Entier], un variant est encadré par des crochets. L'entier ne peut être construit qu'à partir d'une expression de type valeur constante.

Exemple : [20]. La tâche sera exécutée 20 fois

***Note** : par défaut l'expression d'une itération est [1]. Lors de l'exécution d'une tâche le variant décroît à zéro*

Prédicat : il s'agit d'une condition booléenne dont la valeur est retournée par les fonctions *while(Booléen e)* ou *notWhile(Booléen e)*. Si les fonctions retournent *true* l'itération continue sinon elle se termine.

Descriptif / Exigences : while(Expression) ou notWhile(Expression)

Exemple : while(12 < ?NUM) : tant que 12 < ?NUM est vrai l'itération continue.

Concept : Fonctions disponibles pour une expression « itération »

Deux fonctions sont proposées pour construire une expression « itération ». Leurs fonctionnalités et leurs syntaxes sont différentes :

- *Booléen while(Booléen e)* ;
- *Booléen notWhile(Booléen e)*.

Pour construire l'expression *e* qui retourne un booléen, K-MAD fournit une grammaire identique à la construction d'une expression précondition.

Ces fonctions ne manipulent pas explicitement des objets concrets et c'est pourquoi elles ne sont pas identifiées comme fonctions locales ou globales.

Booléen while(Booléen e) : cette fonction retourne *true* si e est vraie, sinon elle retourne *false*.

Descriptif / Exigences : Booléen while(Booléen e) :

Exemple : while(card($Conference, $departDate > 20052006) < 2). Tant que la cardinalité des conférences dont la date est supérieure à 20052006 ne dépasse pas deux, l'itération continue.

***Note** : attention à ce que vous voulez exprimer, il peut y avoir un problème de boucle sans fin.*

Booléen notWhile(Booléen e) : cette fonction retourne *true* si e est faux, sinon elle retourne *false*.

Descriptif / Exigences : Booléen notWhile(Booléen e) :

Exemple : notWhile(card($Conference, $departDate > 20052006) < 2). Tant que la cardinalité des conférences dont la date est supérieure à 20052006 dépasse deux, l'itération continue.

***Note** : attention à ce que vous voulez exprimer, il peut y avoir un problème de boucle sans fin.*

1 - Comment se présente l'outil d'édition pour l'itération d'une tâche ?

Sur la Figure 74 est présentée l'outil d'édition d'une expression itération. Le repère 1 désigne un champ de texte permettant de saisir directement une valeur pour un variant. Le repère 2 désignent quant à lui les deux fonctions permettant de construire une itération à partir d'une prédicat.

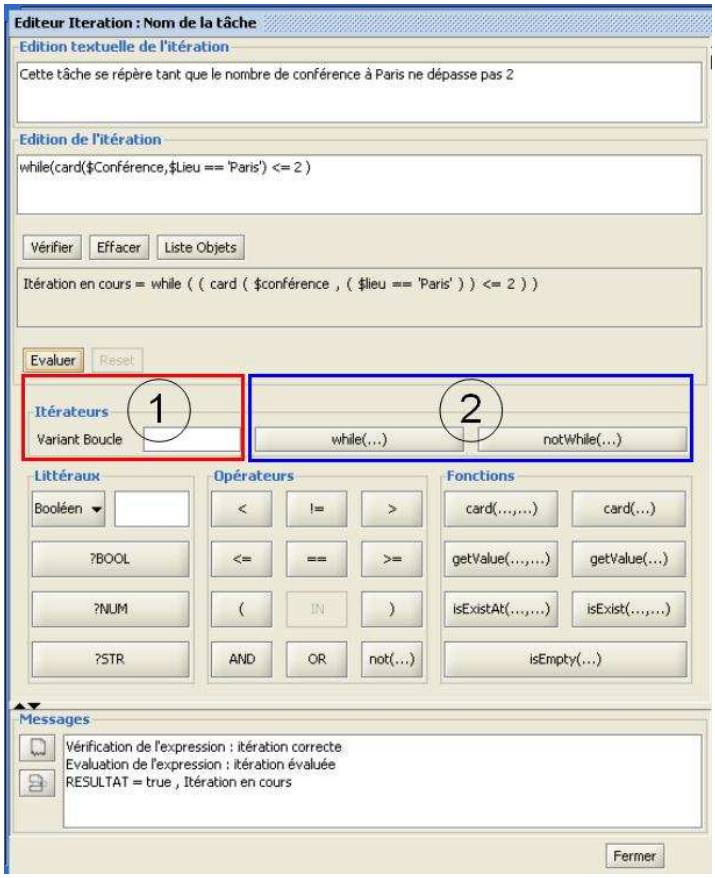

Figure 74 - Outil pour la construction d'une expression itération

CHAPITRE 5

Cohérence du modèle

L'outil de cohérence est un service inclus dans le logiciel K-MADe permettant de vérifier si le modèle défini par l'analyste ne transgresse pas certaines règles établies par la sémantique du modèle K-MAD.

L'outil de cohérence n'interrompt pas les actions de l'analyste au cours de la construction d'un modèle. L'outil de cohérence est à l'initiative de l'analyste et l'informe sur toutes les erreurs rencontrées.

Par ailleurs, certaines erreurs jugées sensibles doivent être résolues pour permettre l'interrogation du modèle (simulation, recherche avancée, statistique, …).

L'outil de cohérence distingue ainsi deux niveaux d'erreurs :

- Les erreurs critiques qui ne permettent pas une interrogation du modèle ;
- Les avertissements qui n'influencent pas l'interrogation du modèle.

Par ailleurs, une erreur peut être de type :

- Hiérarchie, il s'agit d'une erreur liée à la construction de l'arbre de tâches ;
- Expression, il s'agit d'une erreur liée à la construction des expressions contenues dans les caractéristiques : précondition, action et itération.

Concernant les erreurs contenues dans les objets abstraits, objets concrets, utilisateurs, équipements, événements et libellés, les erreurs sont résolues au moment de la construction de ces objets.

On se propose dans cette partie de présenter les différentes erreurs du modèle pouvant être rencontrées et la manière de les résoudre puis nous détaillons l'outil de cohérence. Nous enrichissons en sorte la description du modèle donnée dans le chapitre précédent.

Concept : Erreurs critiques

Les erreurs critiques sont des erreurs qui ne permettent pas une interrogation du modèle. Nous les détaillons ci-dessous :

Pas de tâche unique : cette erreur est provoquée lorsque un modèle K-MAD contient uniquement une seule tâche.

Descriptif / Exigences : critique, hiérarchie, correction : ajouter de nouvelles tâches.

Pas de sous-tâche unique : cette erreur intervient quand une tâche mère possède une seule sous tâche.

Descriptif / Exigences : critique, hiérarchie, correction : ajouter de nouvelles sous-tâches.

Décomposition doit être élémentaire pour une tâche feuille : cette erreur intervient quand une tâche feuille a une décomposition autre qu'élémentaire.

Descriptif / Exigences : critique, hiérarchie, correction : modifier la décomposition de la tâche feuille en élémentaire.

Décomposition non précisée : cette erreur intervient lorsque la décomposition d'une tâche mère n'a pas été définie (décomposition à élémentaire ou inconnue)

Descriptif / Exigences : critique, hiérarchie, correction : modifier la décomposition de la tâche par autre chose que élémentaire ou inconnue

Problème dans l'expression de la précondition : l'expression de la caractéristique précondition contient une erreur (lexicale, syntaxique ou sémantique).

Descriptif / Exigences : critique, expression, correction : éditer l'expression précondition et corriger l'erreur lexicale, syntaxique ou sémantique.

Problème dans l'expression de la postcondition : l'expression de la caractéristique postcondition contient une erreur (lexicale, syntaxique ou sémantique).

Descriptif / Exigences : critique, expression, correction : éditer l'expression postcondition et corriger l'erreur lexicale, syntaxique ou sémantique.

Problème dans l'expression de l'itération : l'expression de la caractéristique itération contient une erreur (lexicale, syntaxique ou sémantique).

Descriptif / Exigences : critique, expression, correction : éditer l'expression itération et corriger l'erreur lexicale, syntaxique ou sémantique.

Concept : Avertissements

Les avertissements sont des erreurs qui même si elles ne sont pas corrigées, n'influencent pas l'interrogation du modèle.

Exécutant non défini : une tâche mère a un exécutant qui n'a pas été défini (la valeur de l'exécutant est inconnu).

Descriptif / Exigences : avertissement, hiérarchie, correction : modifier le type de l'exécutant par autre chose que inconnu.

Exécutant différent : une tâche mère à un exécutant différent que ses sous-tâches.

Descriptif / Exigences : critique, expression,

- si les sous-tâches ont des exécutants différents entre eux, l'exécutant de la tâche mère doit être de type abstrait.
- si les sous-tâches ont des exécutants identiques entre eux, l'exécutant de la tâche mère doit être du même type que l'exécutant des sous-tâches.

1 - Ouvrir l'outil de cohérence

L'outil de cohérence est accessible dans le menu principal « Outils ».

Il peut être ouvert à l'initiative de l'analyste pour vérifier les erreurs du modèle.

Il est ouvert automatiquement si vous tentez d'accéder à un service et s'il existe des erreurs critiques dans le modèle K-MADe en cours d'édition.

***Note** : K-MADe maintient la fenêtre de l'outil « Cohérence » au premier plan. Par ailleurs, toutes modifications apportées au modèle est automatiquement notifiées à l'outil de « Cohérence ». Ainsi la vérification du modèle se fait de manière automatique.*

2 - Description de l'outil de cohérence

Sur la Figure 75 est présenté l'outil de cohérence. Plusieurs informations sont à considérer :

- Le repère 1 désigne le niveaux d'erreur (croix = erreur critique ; interrogation = avertissement) ;
- Le repère 2 désigne l'erreur en elle même, celles décrites précédemment ;
- Le repère 3 désigne le nom de la tâche à qui s'adresse l'erreur ;
- Le repère 4 désigne le type de l'erreur (hiérarchie ou expression) ;
- Le repère 5 localise où se trouve l'erreur.

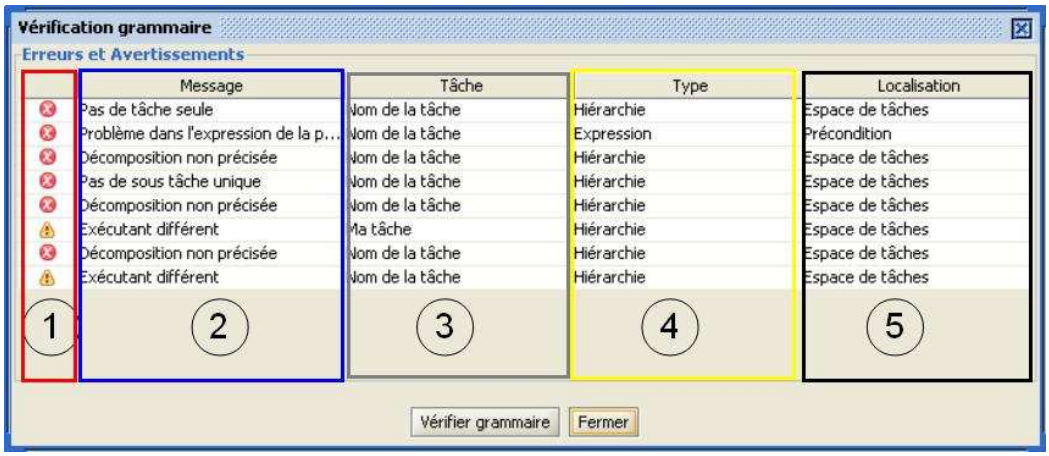

Figure 75 – Outil de cohérence

3 - Corriger une erreur de type expression

Dans le cas où une erreur est de type expression, l'analyste peut atteindre le module d'édition de l'expression erronée.

- Double cliquer sur la ligne contenant une erreur d'expression.

***Note** : en sélectionnant une erreur dans la liste des erreurs, K-MADe localise automatiquement dans l'espace de tâches la tâche associée à l'erreur.*

CHAPITRE 6

Impression du modèle et Recherche textuelle

Ce chapitre a trait aux services liés :

- A l'impression du contenu d'un modèle K-MAD à tous les niveaux (arbre de tâches, fiches utilisateurs, objets du monde) ;
- La recherche textuelle : service qui permet de chercher et remplacer des éléments uniquement dans l'espace de tâches.

A - Impression du modèle K-MAD

Dans un modèle K-MAD, trois types d'informations sont à imprimer :

- Arbre graphique de tâches ;
- Fiches utilisateur ;
- Objets du monde manipulés par l'utilisateur.

1 - Options de mise en page

Sur la Figure 76 sont présentées les options de mise en page :

- Imprimer l'arbre de tâches repère 1 ;
- Modification des marges et du format du papier repère 2 ;
- Mode « paysage » repère 3 ;
- Mode « portrait » repère 4;
- Agrandissement de la vue repère 5;
- Dégrossissement de la vue repère 5;

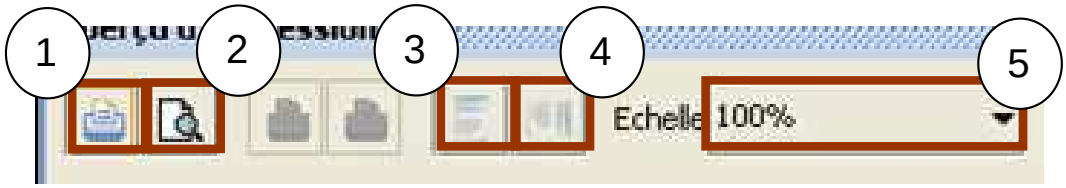

Figure 76 – Options de mise en page

A.2 - Impression de l'arbre de tâches

Cet outil permet d'imprimer la hiérarchie d'un arbre de tâches, en faisant apparaître notamment pour chaque tâche toutes les caractéristiques disponibles dans sa visualisation, c'est-à-dire :

- Le type d'exécutant ;
- Le numéro ;
- Les contraintes liées à l'itération ;
- La décomposition ;
- Le nom de la tâche ;

1 - Ouvrir l'aperçu avant impression

Pour ouvrir l'aperçu avant impression :

- Cliquer sur l'action « aperçu » désignée sur la Figure 77.

Figure 77 – Action pour ouvrir l'aperçu avant impression

2 - Fonctionnement de l'aperçu d'impression d'un arbre de tâches K-MAD

L'aperçu avant impression proposé dans K-MADe permet de configurer l'impression et notamment de paramétrer l'impression de l'arbre de tâches sur plusieurs pages.

Les options liées à l'impression sont désignées par le repère 1 de la Figure 78 tandis que la visualisation de l'arbre est donnée par le repère 2.

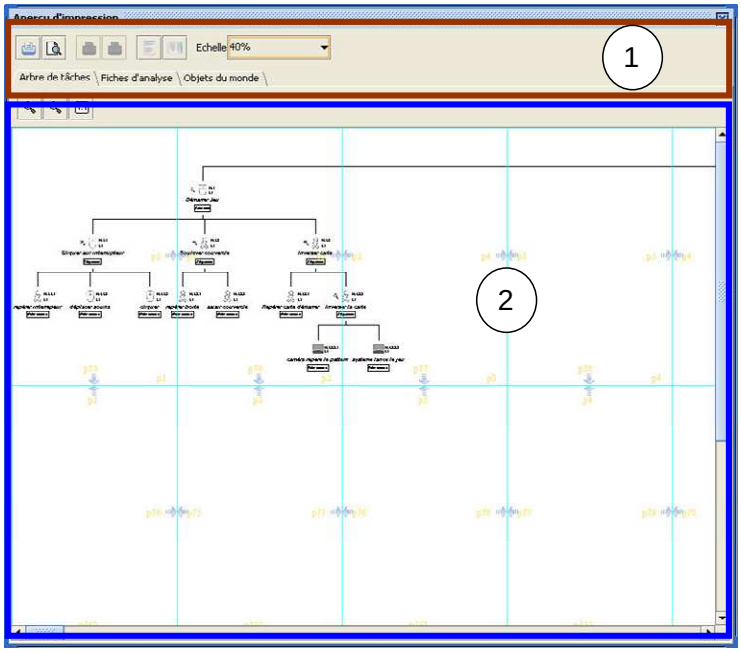

Figure 78 – Aperçu avant impression d'un arbre de tâches K-MAD

Notons que si la dimension d'un arbre de tâches est plus grande que celle d'une page d'impression (la taille de la page peut être modifiée), l'arbre peut être imprimé sur plusieurs pages. Pour cela ; K-MADe exploite un formalisme qui permet de localiser dans le document imprimé les parties de l'arbre de tâches.

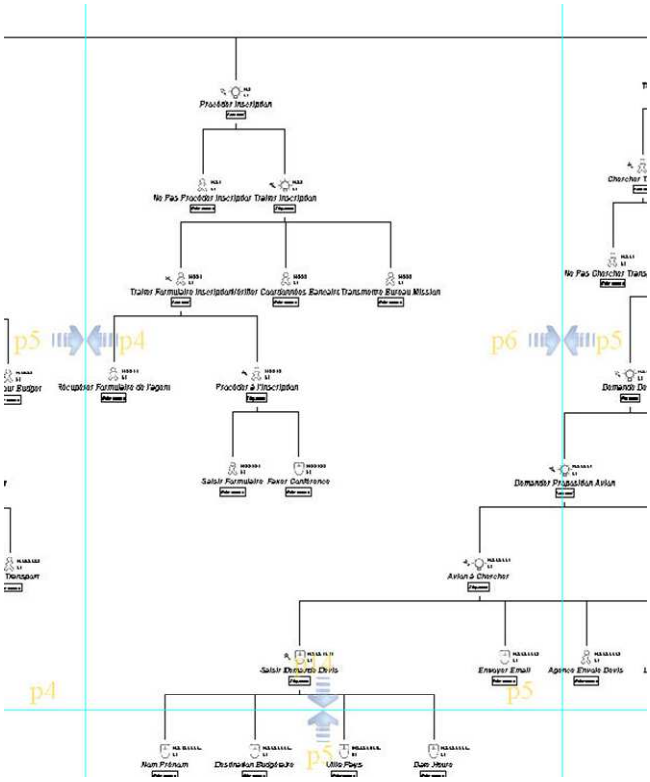

Figure 79 – Capacité d'impression sur plusieurs pages avec la localisation des pages voisines

Sur la Figure 79 est présentée un exemple d'impression de manière à présenter le mécanisme de l'impression multi-pages.

A.3 - Impression des fiches utilisateur

EN COURS DE DEVELOPPEMENT

A.4 - Impression des objets du monde

EN COURS DE DEVELOPPEMENT

B - Recherche textuelle

EN COURS DE REDACTION

CHAPITRE 7

Recherche d'informations sur le modèle

Ce chapitre a trait à l'ensemble des services concernant la recherche d'informations sur le modèle :

- Etablir des statistiques ;
- Fournir des techniques d'interactions pour le parcours d'arbres de grandes tailles dans l'objectif de faciliter la recherche ;
- Interroger le modèle par des critères avancés .

A - Statistique

EN COURS DE DEVELOPPEMENT

B - Visualisation

EN COURS DE DEVELOPPEMENT

C - Recherche avancée

EN COURS DE DEVELOPPEMENT

Chapitre 8

Simulation

L'interconnexion entre les tâches et les objets de l'utilisateur d'une part et de l'aspect formel des objets d'autre part permettent d'animer efficacement des activités spécifiées : c'est le rôle de la simulation. Plus précisément l'objectif de la simulation d'un modèle de tâches K-MAD permet à l'utilisateur de contrôler l'ordonnancement exact des tâches.

K-MADe fournit un outil de simulation réalisant cette fonction. Pour cela la simulation est réalisée étape par étape en proposant à l'utilisateur les tâches pouvant être exécutées selon le contexte du modèle (état des tâches et état du monde). L'utilisateur a la possibilité de modifier les attributs influençant l'ordonnancement afin de visualiser directement les conséquences.

La simulation peut être faite automatiquement (enregistrement et re-jeu de scénario) et permet la simulation des actions sur les objets afin de tenir en compte des conditions (préconditions, utilisateurs et événements déclencheurs).

Nous débutons cette partie par une importante description de la dynamique du modèle K-MADe en s'attardant sur les éléments du modèle qui interviennent directement dans la phase de simulation. L'outil de simulation dispose d'un module d'enregistrement et d'un module de re-jeu. Certaines options de fonctionnement de la simulation sont disponibles dans les deux modules. C'est pourquoi nous présentons à la suite de la description de la dynamique du modèle les options communes puis nous détaillons précisément le fonctionnement du module enregistrement et du module re-jeu de scénarios.

Concept : Déclenchement d'une tâche

Pour qu'une tâche puisse être exécutée, elle doit être atteignable. La décomposition hiérarchique des tâches joue ce rôle. Elle fixe la règle de décomposition des tâches et leurs accessibilités.

Hormis la caractéristique de décomposition, quatre contraintes sont à respecter pour exécuter une tâche :

- l'utilisateur qui exécute la tâche ;
- l'événement déclencheur ;
- le respect de la précondition associée aux objets de l'état du monde ;
- le caractère itératif de la tâche.

Note : nous reviendrons dans la suite de cette partie sur la désignation des états d'une tâche au cours d'une simulation.

Contrainte Utilisateur

Il s'agit de vérifier si l'utilisateur est autorisé ou pas à exécuter la tâche. La caractéristique Acteur contient l'ensemble des utilisateurs autorisé à déclencher la tâche. Il suffit donc de vérifier si l'utilisateur qui souhaite exécuter la tâche est disponible dans la caractéristique Acteur.

***Note**: cette contrainte n'est prise en compte lors de la simulation que si la tâche est élémentaire. L'édition à tous les niveaux du modèle hiérarchique d'un modèle K-MAD est par contre autorisée.*

Contrainte Evénement déclencheur

La caractéristique Déclenchement contient un événement déclencheur. Par ailleurs, la gestion d'un événement déclencheur est assimilable à une ressource. Soit l'événement existe et la tâche le consomme (l'événement est supprimé de la liste des événements générés), soit l'événement n'existe pas et la tâche ne se déclenche pas.

***Note**: cette contrainte n'est prise en compte lors de la simulation que si la tâche est élémentaire. L'édition à tous les niveaux du modèle hiérarchique d'un modèle K-MAD est par contre autorisée.*

Contrainte Précondition

Il s'agit d'une expression construite par rapport aux objets de l'état du monde. Si cette expression est vraie (elle retourne *true*) la contrainte est respectée et la tâche peut ainsi être exécutée.

***Note**: cette contrainte n'est prise en compte lors de la simulation que si la tâche est élémentaire. L'édition à tous les niveaux du modèle hiérarchique d'un modèle K-MAD est par contre autorisée.*

Contrainte Itération

EN COURS DE REDACTION

Concept : Traitement d'une tâche

Le traitement d'une tâche consiste à modifier l'état du monde. Cette modification des concepts manipulés par l'utilisateur concerne les objets concrets par l'intermédiaire des actions et des événements.

***Note**: ces contraintes ne sont prises en compte lors de la simulation que si la tâche est élémentaire. L'édition à tous les niveaux du modèle hiérarchique d'un modèle K-MAD est par contre autorisée.*

Traitement Génération d'événements

Ce traitement est le premier.

Cela concerne une liste d'événements générée au cours de la simulation.

***Note** : Il ne peut y avoir de doublon dans la liste des événements générés. Lorsqu'un événement est émis, il y a une vérification pour savoir s'il existe ou pas dans la liste. S'il n'existe pas, l'événement est ajouté ; s'il en existe un autre l'événement généré est abandonné.*

Traitement Action

Ce traitement fait suite à la génération d'événements.

Il s'agit d'une expression construite pour modifier les objets concrets de l'état du monde (ajout ou suppression d'un nouvel objet concret, modification d'un attribut concret, etc.).

Si une erreur intervient lors de l'évaluation de l'expression (absence de valeurs utilisateurs ou d'une erreur liée à l'exécution d'une fonction), le traitement de la tâche est interrompu. L'état de la tâche ne change pas et le ou les événement(s) généré(s) sont supprimés.

Concept : États d'une tâche lors de la dynamique du modèle

Au cours de la simulation, une tâche peut évoluer dans son état. Cet aspect caractérise ainsi l'évolution de la dynamique de l'arbre hiérarchique.

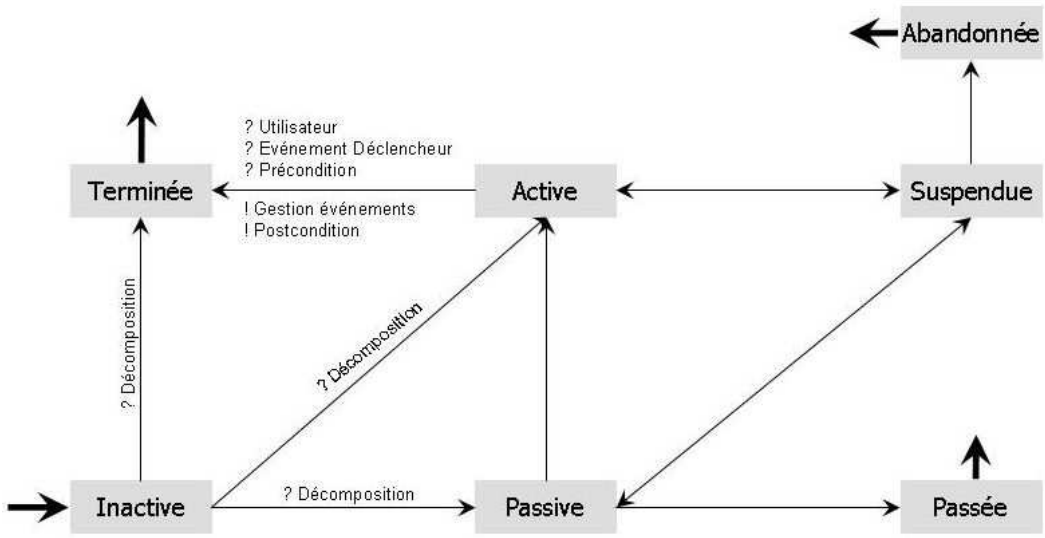

Figure 80 – Automate des différents états d'une tâche

La Figure 80 présente un automate simplifié des différents états d'une tâche au cours de la simulation. La flèche en gras dirigée vers l'état « Inactive » désigne l'état initial tandis que les flèches en gras dirigées vers l'extérieur désignent un état final). Concernant les transitions le caractère ? désigne une condition de déclenchement de la transition et ! une action. **Pour des raisons de simplification nous n'avons pas précisé l'aspect itératif sur le schéma.**

État Inactive

Une tâche dans l'état « Inactive » est une tâche qui n'est pas réalisable. C'est l'état par défaut des tâches. L'analyste ne peut pas interagir avec les tâches se trouvant dans cet état.

Les décompositions sont les seuls à pouvoir changer cet état et à rendre une sous-tâche non inactive.

État Passive

Une tâche dans l'état « Passive » est une tâche accessible, c'est-à-dire utilisable par l'analyste lors de la simulation. Elle est dans un état passif, à ce stade, elle n'est pas encore déclenchée, mais peut l'être par l'analyste.

État Active

Une tâche dans l'état « Active » est une tâche qui a été déclenchée pour être réalisée. Les contraintes qui interviennent à ce niveau et qui doivent être vérifiées sont : l'utilisateur, l'événement déclencheur et la précondition.

Pour les tâches de décomposition élémentaire (tâche feuille), cet état est instantané. Lors de l'exécution d'une tâche feuille son état passe directement à « Terminée » ou « Passive » (le cas d'une itération par exemple).

État Terminée

Une tâche dans l'état « Terminée » signale que la tâche a terminé son exécution. Les traitements relatifs à la tâche sont réalisés : les actions et la génération d'événements.

État Passée

Une tâche dans l'état « Passée » signifie que la tâche ne sera jamais traitée (pas de postcondition et ni de génération d'événements). C'est le cas des tâches facultatives qui peuvent ne pas être exécutées.

> ***Note** : une tâche dont l'exécutant est Système ne peut jamais être dans l'état Passée puisqu'elle ne peut être facultative.*

État Suspendue

Une tâche dans l'état « Suspendue » indique que la tâche n'est plus en cours d'exécution, elle a été interrompue par l'analyste.

> ***Note** : cet état ne peut intervenir que si la tâche est interruptible.*

État Abandonnée

Une tâche dans l'état « Abandonnée » exprime le fait que la tâche a été abandonnée suite à une interruption.

Cet état ne peut intervenir que si la tâche est interruptible et optionnelle (caractéristique nécessité).

> ***Note** : une tâche dont l'exécution est « système » ne peut être optionnelle, par conséquent elle ne peut être abandonnée.*

Concept : Actions lors de la simulation

Les actions permettent à l'analyste d'animer un modèle de tâche K-MAD. Une action est un point d'entrée sur la dynamique du modèle. Elle permet une variabilité dans le déroulement de l'activité. La disponibilité de telle ou telle action est contrainte par l'état en cours des tâches.

Une action s'applique soit à une tâche élémentaire soit à une tâche mère.

Pour chaque action nous donnons l'état précédent à l'action puis l'état suite à l'action. L'état est donné entre (…) et l'action entre […]. Nous précisons également les conditions de disponibilité des actions.

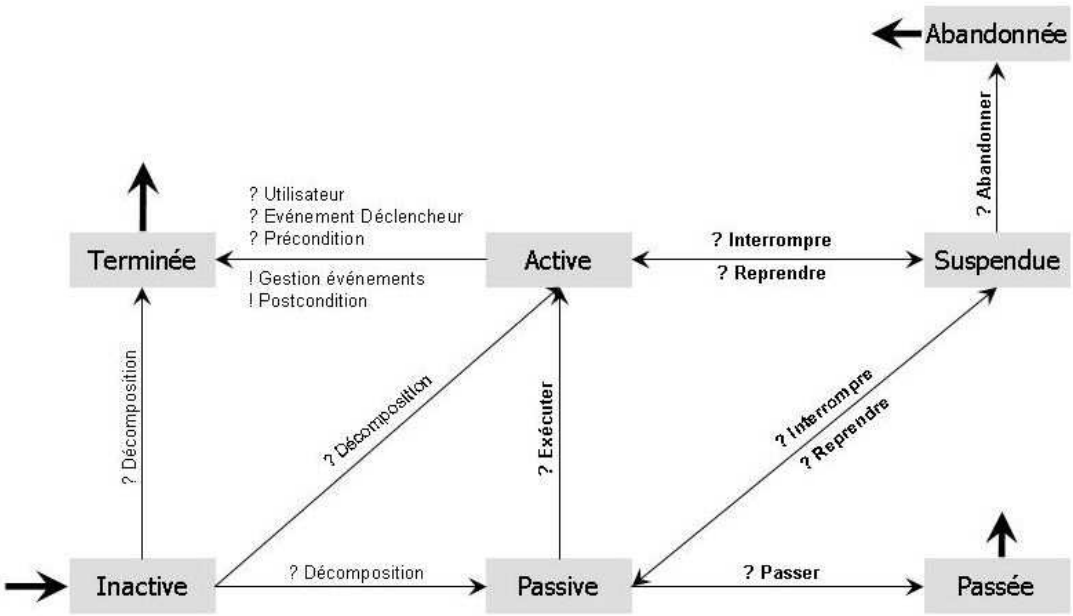

Figure 81 – Automate des différents états d'une tâche incluant également les différentes actions de l'analyste

Sur la Figure 81 nous enrichissons le schéma de la Figure 80 en donnant les possibles actions de l'analyste au cours de la simulation. **Nous avons également simplifié le schéma en éliminant les aspects liés à l'itération**.

Action Exécuter

Cette action permet d'exprimer explicitement, par l'analyste, le souhait d'exécuter une tâche élémentaire à partir du moment qu'elle est dans l'état « Passive ». Suite, à cette action, l'état de la tâche passe à l'état « Active ».

Descriptif / Exigences :

- État (Passive) [exécuter] -> Etat (Active) ;
- Uniquement tâches élémentaires.

Action Passer

Ce choix est relatif à la caractéristique « Nécessité ». Si la tâche est optionnelle, l'action de passer la tâche est proposée à l'analyste. Cette action intervient uniquement quand la tâche est dans l'état « Passive » et aboutie à l'état « Passée ».

Descriptif / Exigences :

- État (Passive) [passer] -> Etat (Passée) ;
- Uniquement les tâches optionnelles (pas d'exécutant Système) ;
- Concerne les tâches élémentaires et les tâches mères ;
- N'intervient pas lorsque la décomposition de la tâche mère est Alternatif (si T_1 est optionnelle et T sa tâche mère à une décomposition de type Alternatif alors T_1 ne peut être passée).

Action Interrompre

Cette action permet de suspendre une tâche (tâche passe dans l'état « Suspendue »). Elle intervient uniquement si la tâche est dans l'état « Passive » et qu'elle peut être Interruptible.

Descriptif / Exigences :

- État (Passive) [Interrompre] -> Etat (Suspendue) ;
- Uniquement les tâches interruptibles ;
- Concerne les tâches élémentaires et les tâches mères.

***Note** : l'interruption d'une tâche ne concerne pas l'interruption du traitement effectif de celle-ci (génération d'événement et d'action). Le traitement est atomique c'est-à-dire que dés l'exécution du traitement, celui-ci ne peut être arrêté et doit se terminer.*

Action Reprendre

Cette action est uniquement disponible si une tâche est dans l'état « Suspendue ». Elle a pour but d'arrêter l'interruption en cours afin que la tâche puisse de nouveau être dans l'état « Passive ».

Descriptif / Exigences :

- État (Suspendue) [Reprendre] -> Etat (Passive) ;
- Uniquement les tâches interruptibles ;
- Concerne les tâches élémentaires et les tâches mères.

Action Abandonner

Cette action est disponible dans le cas où une tâche serait optionnelle et dans l'état « Suspendue ». Le traitement de la tâche ou de ces sous-tâches ne sont pas exécutés.

Descriptif / Exigences :

- État (Suspendue) [Abandonner] -> Etat (Abandonnée) ;
- Uniquement les tâches interruptibles et optionnelle ;
- Ne concerne pas les exécutants Système ;
- Concerne les tâches élémentaires et les tâches mères.

Concept : Processus de simulation d'un arbre de tâches K-MAD

Démarrage d'une simulation

Lors du démarrage d'une simulation, une tâche « Racine » est choisie (explicitement par l'analyste lors de l'enregistrement d'un scénario et implicitement par le scénario à rejouer lors de la phase de re-jeu d'un scénario) et son état est passé à « Passive ».

***Note** : toutes les tâches ne faisant pas partie des sous-tâches de la tâche « Racine » ne seront pas prise en compte lors de la simulation.*

Traitement en cours d'une simulation

Tout au long de la simulation ('jusqu'à la phase de terminaison), les états des tâches sont déterminés à la fois par la décomposition des tâches et par l'état du monde par l'intermédiaire des expressions (précondition, action, itération) des événements et des utilisateurs.

Terminaison d'une simulation

La simulation d'un arbre de tâches K-MAD se termine soit à la demande explicit de l'analyste soit si toutes les tâches du modèle sont dans l'état « Terminée ».

Concept : Décompositions en détail

On s'intéresse dans ici à décrire plus précisément le fonctionnement des décompositions en les présentant avec les différents états des tâches. De manière à simplifier l'explication nous considérons une tâche mère exemple T_0 [DEC] -> T_1, T_2 où DEC désigne la décomposition de la tâche mère et T_1 puis T_2 sont des tâches élémentaires, obligatoire et non interruptible. Par ailleurs, les tâches T_0, T_1 et T_2 sont initialement mises dans l'état inactive. Nous allons donc procéder à la simulation de la tâche mère T_0.

Séquence : T_0 [SEQUENCE] -> T_1, T_2

Les sous-tâches T_1 et T_2 sont exécutées une par une.

Descriptif / Exigences : la simulation de la tâche T_0 conduit aux états initiaux suivants : T_0, T_1 passives et T_2 inactive.

- T_1 est exécutée les nouveaux états sont : T_0 active, T_1 terminée et T_2 passive.
- T_2 est exécutée les nouveaux états sont : T_0, T_1 et T_2 terminées, la simulation est terminée.

Alternatif : T_0 [ALTERNATIF] -> T_1, T_2

Une seule sous-tâche est exécutée : T_1 ou T_2.

Descriptif / Exigences : la simulation de la tâche T_0 conduit aux états initiaux suivants : T_0, T_1 et T_2 passives.

Deux possibilités : soit T_1 est exécutée soit T_2 est exécutée.

Si T_1 est exécutée :

- Les nouveaux états sont : T_0, T_1 et T_2 terminées, la simulation est terminée.

Ou

Si T_2 est exécutée :

- Les nouveaux états sont : T_0, T_1 et T_2 terminées, la simulation est terminée.

Pas d'ordre : T_0 [PAS ORDRE] -> T_1, T_2

Les sous-tâches sont exécutées de manière entrelacée : T_0 puis T_1 ou T_1 puis T_0.

Descriptif / Exigences : la simulation de la tâche T_0 conduit aux états initiaux suivants : T_0, T_1 et T_2 passives.

Deux possibilités : soit T_1 est exécutée soit T_2 est exécutée.

Si T_1 est exécutée en premier :

- T_1 est exécutée les nouveaux états sont : T_0 active, T_1 terminée et T_2 passive ;
- T_2 est exécutée les nouveaux états sont : T_0, T_1 et T_2 terminées, la simulation est terminée.

Ou

Si T_2 est exécutée en premier :

- Les nouveaux états sont : T_0 active, T_1 passive et T_2 terminée ;
- T_2 est exécutée les nouveaux états sont : T_0, T_1 et T_2 terminées, la simulation est terminée.

Variante : dans le cas où T_1 et T_2 sont des tâches non élémentaires, il n'y a pas d'entrelacement des sous-tâches. T_1 et T_2 ne peuvent pas être active en même temps (le cas de la décomposition concurrence).

Nous obtenons le comportement suivant :

- Si T_1 est « Active » et T_2 est non « Terminée » alors T_2 est « Inactive » (aucune action sur T_2 tant que T_1 n'est pas terminée) ;
- Si T_1 est « Terminée » et T_2 non « Terminée » alors T_2 est « Passive » ;
- Si T_2 est « Active » et T_2 non « Terminée » alors T_2 est « Inactive » (aucune action sur T_1 tant que T_2 n'est pas terminée) ;
- Si T_2 est « Terminée » et T_1 non « Terminée » alors T_2 est « Passive » ;
- Si T_1 et T_2 sont « Terminée » alors T_0 est « Terminée » également. La simulation est terminée.

Parallèle : T_0 [CONCURRENCE] -> T_1, T_2

Les sous-tâches sont exécutées en même temps avec un entrelacement des sous-tâches.

Descriptif / Exigences : la simulation de la tâche T_0 conduit aux états initiaux suivants : T_0, T_1 et T_2 passives.

Deux possibilités : soit T_1 est exécutée soit T_2 est exécutée.

Si T_1 est exécutée en premier :

- T_1 est exécutée les nouveaux états sont : T_0 active, T_1 terminée et T_2 passive ;
- T_2 est exécutée les nouveaux états sont : T_0, T_1 et T_2 terminées, la simulation est terminée.

Ou

Si T_2 est exécutée en premier :

- Les nouveaux états sont : T_0 active, T_1 passive et T_2 terminée ;

T_2 est exécutée les nouveaux états sont : T_0, T_1 et T_2 terminées, la simulation est terminée.

Variante : dans le cas où T_1 et T_2 sont des tâches non élémentaires, il y a entrelacement des sous-tâches. T_1 et T_2 peuvent être active en même temps.

Nous obtenons le comportement suivant :

- Si T_1 est « Active » et T_2 est non « Terminée » alors T_2 est « Passive » ou « Active » ;
- Si T_1 est « Terminée » et T_2 non « Terminée » alors T_2 est « Passive » ou « Active » ;
- Si T_2 est « Active » et T_2 non « Terminée » alors T_2 est « Inactive » ou « Active » ;
- Si T_2 est « Terminée » et T_1 non « Terminée » alors T_2 est « Passive » ou « Active » ;
- Si T_1 et T_2 sont « Terminée » alors T_0 est « Terminée » également. La simulation est terminée.

Concept : Comportement d'une tâche itérative

EN COURS DE REDACTION

Concept : Tâche interruptible

La caractéristique interruptible d'une tâche conduit à des états particuliers lors de la simulation. Ces états particuliers interviennent dans le cas où une tâche T interruptible satisfait les conditions suivantes:

- La tâche mère de T a une décomposition de type concurrence ou ordre entrelacé ;
- La valeur de la caractéristique «importance » de T est différente de « Non Déterminée » ;
- Les tâches sœurs de T peuvent avoir des valeurs quelconques concernant la caractéristique « importance ».

Plusieurs cas peuvent se produire :

- Si la tâche T interruptible est un exécutant de type Système et son importance est plus basse que ses tâches sœurs : T est interrompue obligatoirement sans possibilité de reprise ;
- Si la tâche T interruptible est un exécutant de type Système et son importance est plus haute ou égale que ses tâches sœurs : il y a possibilité d'interrompre T avec capacité de reprise ;
- Si la tâche T interruptible est un exécutant autre que Système et son importance est plus basse que ses tâches sœurs : T est interrompue obligatoirement avec capacité de reprise ;
- Si la tâche interruptible est un exécutant de type Système et l'importance est plus haute que ses tâches sœurs : il y a possibilité d'interrompre T avec capacité de reprise .

Concept : Valeurs Utilisateurs

Les valeurs utilisateurs sont des expressions dont le type est fixé (Booléen, Texte, Nombre) et dont les valeurs sont données par l'analyste.

Au cours de la simulation si une tâche doit être exécutée (l'action Exécuter pour une tâche dans l'état Passive), le modèle vérifie si les expressions des caractéristiques précondition, action et itération possèdent oui ou non des valeurs utilisateurs. Dans le cas où les expressions possèdent des valeurs utilisateurs, la simulation ne peut continuer que si

- Toutes les valeurs utilisateurs sont données ;
- Toutes les expressions sont correctes du point de vue sémantique (problème de typage essentiellement).

Les valeurs saisies par l'analyste vont donc permettre de modifier le déroulement de la simulation d'un modèle K-MAD.

Exemple : il se peut qu'une tâche ne puisse être exécutée car sa précondition (12 < ?NUM) n'est pas respectée car l'analyste a donné la valeur 2 pour ?NUM.

1 - Ouvrir l'outil de simulation

L'utilisation du service de simulation impose que le modèle de tâches K-MAD en cours d'édition soit vérifié et validé par l'outil de cohérence. Les erreurs de type avertissement sont autorisées durant la simulation.

L'outil de cohérence est accessible dans le menu principal « Outils ».

2 - Présentation de l'outil de simulation

L'outil de simulation comporte huit grandes zones (voir Figure 82) :

- Zone des caractéristiques d'une tâche (repère 1) ;
- Zone « objet concret » (repère 2) ;
- Zone « événements générés » (repère 3) ;
- Zone « utilisateur » (repère 4) ;
- Zone « espace de tâches » (repère 5) ;
- Zone « enregistrement » ou « re-jeu » d'un scénario (repère 6) ;
- Zone « contrainte » (repère 7) ;
- Zone « messages » (repère 8) .

Dans la suite de ce manuel, nous allons nous intéresser à décrire chaque zone et les possibles interactions. Concernant la zone liée à l'enregistrement ou au re-jeu d'un scénario nous les détaillons en détail dans les parties E.2 et E.3.

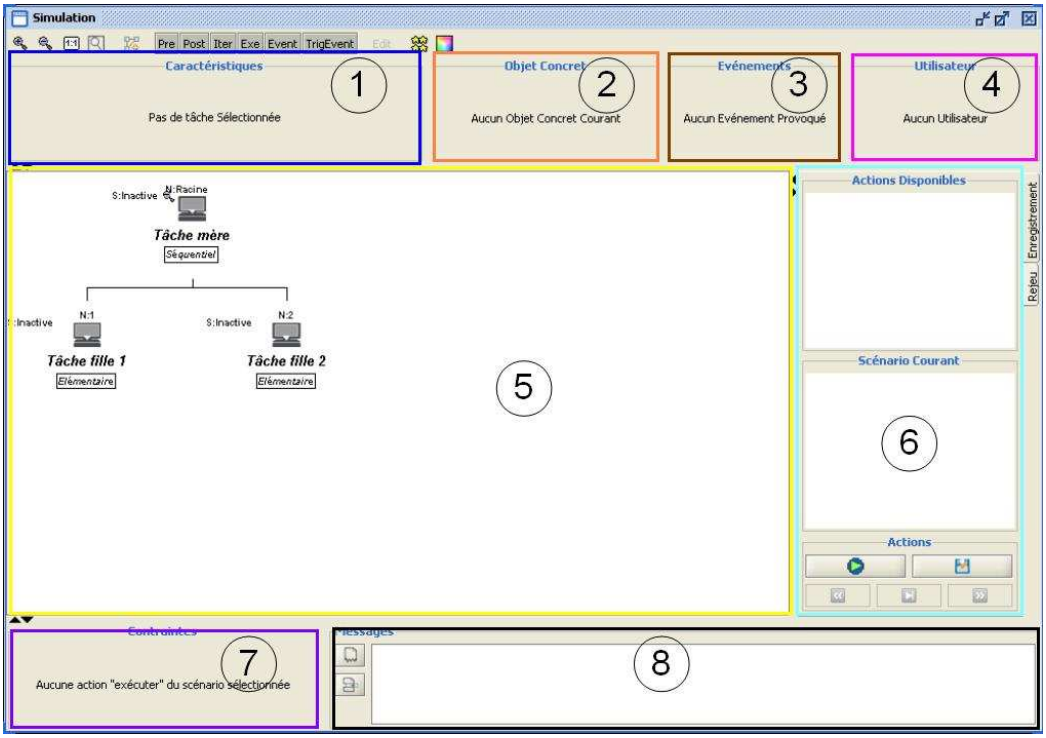

Figure 82 – Aperçu complet de l'outil de simulation

3 - La zone « Espace de Tâches » de la simulation

La zone « Espace de Tâches » présente l'arbre de tâches en cours de simulation. Il n'est pas autorisé à l'analyste de modifier l'arbre et le contenu des tâches. Les accès en modification à l'arbre sont désactivés.

4 - Description graphique d'une tâche au cours de la simulation par rapport à ces caractéristiques ?

Nous donnons sur la Figure 83, une description d'une représentation graphique d'une tâche en cours de simulation.

Il est à noter que la représentation graphique est basée sur celle utilisée pendant l'édition de l'arbre de tâches.

La Figure 83 précise les caractéristiques suivantes :

- Repère 1 : le type d'exécutant ;
- Repère 2 : le numéro (calculé automatiquement) ;
- Repère 3 : les contraintes liées à l'itération ;
- Repère 4 : l'état de la tâche ;
- Repère 5 : le nom de la tâche ;
- Repère 6 : la décomposition

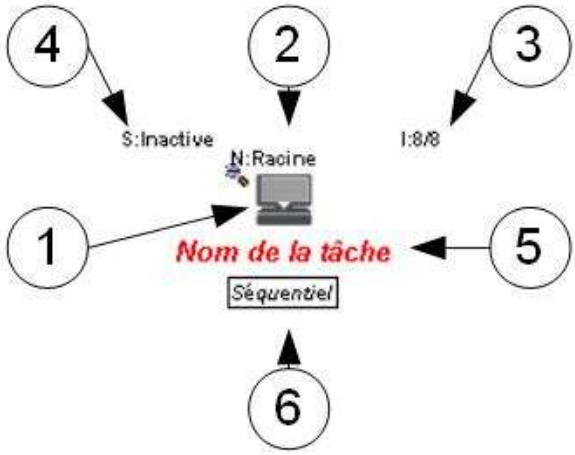

Figure 83 – Représentation graphique de caractéristiques d'une tâche

***Note** : il n'apparaît pas dans cette représentation les ancres qui permettent de relier une tâche à d'autres tâches.*

En plus de l'information de l'état de simulation d'une tâche donnée par le repère 4, K-MADe fournit un codage de couleur permettant d'identifier dans l'arbre complet cet état.

Tâche dans l'état *inactive*

Tâche dans l'état *passive*

Tâche dans l'état *active*

Tâche dans l'état *terminée*

Tâche dans l'état *passée*

Tâche dans l'état *suspendue*

Tâche dans l'état *abandonnée*

5 - La zone « Caractéristiques »

Figure 84 – Zone de caractéristique

La zone de caractéristique présentée sur la Figure 84 (également désignée sur repère 1 de la Figure 82) permet à l'analyste de lire l'ensemble des caractéristiques d'une tâche sélectionnée.

Note : à la différence de l'outil d'édition, les caractéristiques ne sont pas éditables.

6 - La zone « Objet concret »

Cette zone permet de connaître l'objet concret stocké dans le buffer de K-MAD (voir Figure 85).

Le format adopté est le suivant :

- G : nom du groupe ;
- OC : nom de l'objet concret.

Exemple : G : Garage, OC : maclio. Le nom du groupe G est Garage et le nom de l'objet concret est maclio.

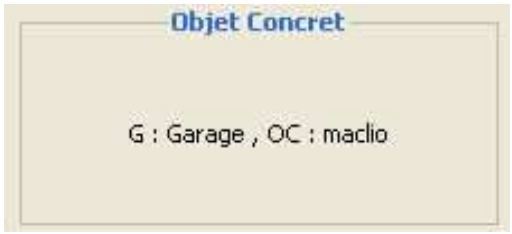

Figure 85 – Zone d'objet concret

7 - La zone « Evénements »

Cette zone permet de connaître l'ensemble des événements qui ont été générés par le modèle K-MAD en cours de simulation (voir Figure 86).

Le format adopté est le suivant :

- E : nom de l'événement ;
- T : nom de la tâche qui a généré l'événement

Exemple : E : PlusDePapier, T : Nom de la tâche. Le nom de l'événement E est PlusDePapier et le nom de la tâche qui a généré l'événement est Nom de la tâche.

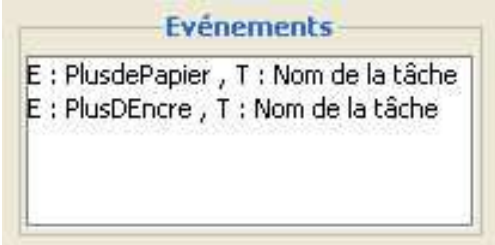

Figure 86 – Zone d'événements générés

8 - La zone « Utilisateur »

Cette zone permet de choisir l'utilisateur courant, c'est-à-dire celui qui doit exécuter les tâches de l'arbre. Avant chaque exécution d'une tâche, l'analyse peut choisir l'utilisateur qu'il désire voir accomplir une tâche (voir Figure 87).

Figure 87 – Zone des utilisateurs

9 - La zone « Contraintes »

Cette zone permet de savoir si une tâche est exécutable ou pas. Les informations ne sont affichées que si l'analyste sélectionne une action de type « exécuter » (voir la partie enregistrement et re-jeu A.1 et A.2). Il s'agit en fait des contraintes concernant le déclenchement d'une tâche, c'est-à-dire :

- Acteur : est-ce le bon utilisateur qui déclenche la tâche ?;
- Précondition : contrainte lié aux objets du monde ;
- Déclenchement : la tâche est-elle déclenchée par un événement.

Les informations sont affichées à l'aide d'une table (voir Figure 88). La première colonne indique le type de contrainte considéré, la deuxième colonne concerne la valeur contenue dans la tâche et enfin la troisième colonne indique si la contrainte est respectée. Si les trois contraintes sont respectées, la tâches peut être exécutée.

Figure 88 – Zone « contraintes »

10 - La zone « Messages »

Cette zone (voir Figure 89) donne des renseignements sur l'exécution d'une tâche. Plus particulièrement sont précisées les informations suivantes :

- Contraintes d'exécution : utilisateur, événement et précondition ;
- Actions : génération d'événements et postcondition.

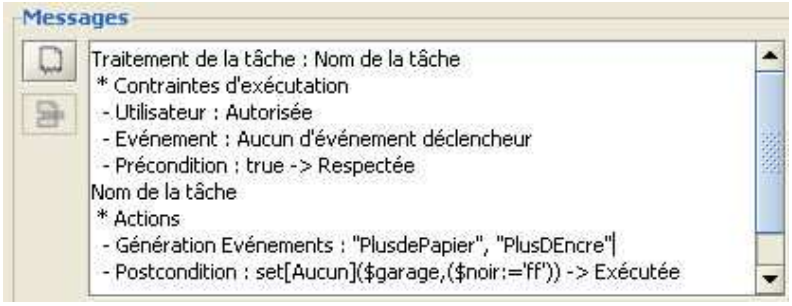

Figure 89 - Zone « messages » suite à l'exécution d'une tâche

11 - Activer/Désactiver les options de visualisation

Le service de simulation utilise des outils de visualisation identiques à ceux utilisés dans la partie édition de l'arbre de tâches (Figure 90). Nous retrouvons donc :

- Agrandissement et réduction ;
- Retour à la taille originale ;
- Vue globale de l'arbre ;
- Aperçu complet de l'arbre de tâches ;
- Libellé Visible et Libellé Couleur pour rendre visible ou affiche les couleurs des libellés.

Figure 90 – Boîte à outils de l'outil de simulation

Par ailleurs, d'autres options sont disponibles, notamment celles permettant d'activer ou de désactiver des contraintes ou traitement qui interviennent pendant la simulation :

- Pre : active ou désactive la prise en compte de la contrainte précondition ;
- Exe : active ou désactive la prise en compte de la contrainte exécutant ;
- Event : active ou désactive la prise en compte de la contrainte événement déclencheur ;
- Iter : active ou désactive la prise en compte de la contrainte itération ;
- Post : active ou désactive le traitement de la postcondition ;
- TrigEvent : active ou désactive le traitement de la génération des événements ;
- Edit : permet l'édition des valeurs utilisateurs.

12 - Édition de valeurs utilisateurs et des objets concrets

Certaines expressions nécessitent la saisie de valeurs utilisateurs et le choix d'objets concrets pour l'exécution de certaines fonctions. L'outil présenté sur la Figure 91 permet la saisie de ces informations.

Lorsqu'une tâche est exécutée et que des valeurs utilisateur ou des objets concrets sont demandés K-MADe affiche automatiquement cet outil. Tant que les expressions ne sont pas sémantiquement vérifiées, la tâche n'est pas exécutée.

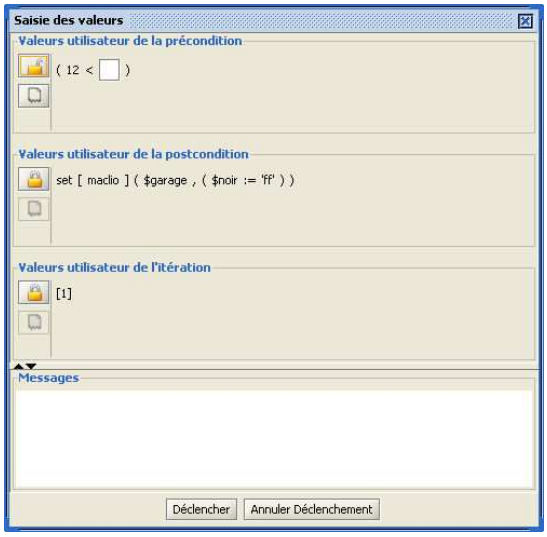

Figure 91 – Outil d'édition des valeurs utilisateurs

A - Manipulation de scénarios

Concept : Scénario

Un scénario peut être vu comme une description précise d'utilisation dans un contexte précis.

Un scénario dans K-MAD repose :

- Sur un parcours particulier dans l'arbre de tâches ;
- Des valeurs particulières (valeurs utilisateurs, objets concrets, événements et utilisateur).

Ainsi un modèle de tâches K-MAD peut aider à l'identification de scénarios intéressants permettant par exemple à l'évaluation de systèmes interactifs.

Cette partie a trait à la manipulation de scénarios. Nous distinguons deux aspects :

- L'enregistrement de scénarios ;
- Le re-jeu de scénarios.

A.1 - L'enregistrement de scénarios

Concept : Enregistrement d'un scénario

L'enregistrement d'un scénario consiste à simuler un arbre de tâches en choisissant, tout au long de ces différents états, les actions à appliquer d'une part et les valeurs particulières d'autre part. Par ailleurs, un enregistrement débute par le choix de la tâche à simuler.

Choix de la tâche à simuler

Avant que la simulation ne débute et par conséquent l'enregistrement du scénario, l'analyste doit choisir la tâche de l'arbre qui sera considérée comme « Racine ».

Choix des actions à appliquer

Tout au long de la simulation l'analyste choisit les actions qui sont disponibles selon l'état du modèle. Rappelons que les actions sont au nombre de cinq :

- Exécuter ;
- Passer ;
- Interrompre ;
- Reprendre ;
- Abandonner.

Choix de valeurs particulières

Les valeurs particulières influencent le déroulement de la simulation. Plus précisément, une simulation peut être influencée par :

- Le choix d'un utilisateur ;
- Une valeur utilisateur (?INT, ?STR et ?BOOL) ;
- Un objet concret particulier (si le type d'un groupe est ensemble) ;
- L'émission d'un événement.

Fin d'un enregistrement d'un scénario

La fin d'un enregistrement peut se terminer d'une part à la demande de l'utilisateur, c'est-à-dire en cours de simulation (il subsiste encore des actions à exécuter) où d'autre part s'il n'y a plus d'action à exécuter.

Dans le cas où il n'y a plus d'action à exécuter, K-MAD distingue deux formes de terminaison :

- La première est une fin complète où la tâche la plus haute de l'arbre est dans l'état terminée ;
- La seconde est une fin incomplète où la tâche la plus haute de l'arbre est dans l'état non terminée.

1 - Présentation de l'outil d'enregistrement

La Figure 92 présente l'outil d'enregistrement. Il se décompose en trois parties :

- Les actions disponibles qui peuvent être appliquées sur le modèle en cours de simulation (repère 1) ;
- Le scénario en cours de construction qui peut être vu comme un historique des actions appliquées sur le modèle K-MAD (repère 2) ;
- Une zone de contrôle (repère 3).

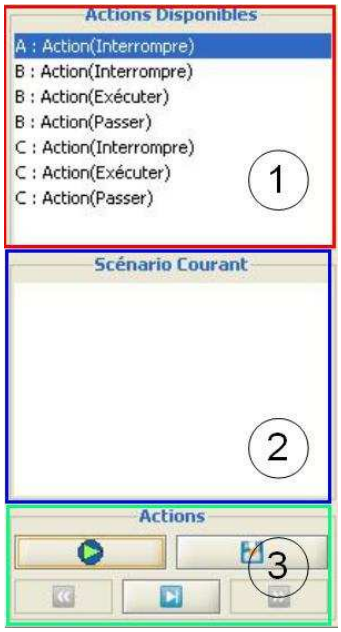

Figure 92 – Zone « d'enregistrement »

2 - Présentation de la zone « Actions Disponibles »

La zone « actions disponibles » liste l'ensemble des actions à un moment donné avec lesquelles l'analyste peut modifier le cours de la simulation (ces actions sont déduites des règles de décomposition).

Le format adopté est le suivant :

- Nom de la tâche
- Action : type de l'action.

Exemple : Tache AB : Action(Interrompre). Le nom de la tâche est Tache AB et le type de l'action est Interrompre.

3 - Présentation de la zone « Scénario Courant »

La zone « scénario courant » liste le scénario en cours de construction. Pour chaque action que l'analyste a effectué sur le modèle K-MAD en cours de simulation, K-MADe les affiche dans cette zone suivant leur ordre d'exécution.

Le format adopté est le suivant :

- Indice qui précise la position de l'action ;
- Nom de la tâche ;
- Action : type de l'action.

Exemple : 1 – Tache A : Action(Exécuter). Première Action exécutée, appliquée à la tâche Tache A dont le type est Exécuter.

4 - Présentation de la zone « Actions »

La Figure 93 présente, l'outil de contrôle concernant l'enregistrement d'un scénario.

- Le repère 1 désigne l'action permettant de débuter une simulation ;
- Le repère 2 désigne, l'outil de sauvegarde du scénario courant ;
- Enfin le repère 3 a trait à l'action d'exécuter une action.

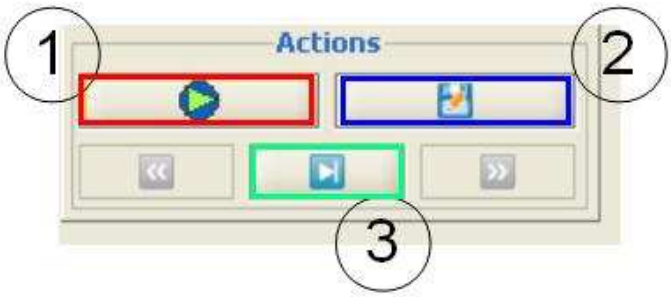

Figure 93 – Zone « Actions pour l'enregistrement »

***Note** : les actions de retour et d'avance sont en cours de développement.*

5 - Débuter un l'enregistrement d'un scénario

- Sélectionner dans l'arbre de tâche, la tâche qui sera considérée comme racine ;
- Cliquer sur l'option « Débuter Simulation » (repère 1 de la Figure 93) ;

6 - Exécuter une action

Deux façons sont proposées pour exécuter une action

- Double cliquer sur l'action directement dans la zone « Actions Disponibles ».

Ou

- Sélectionner l'action dans la liste des « Actions Disponibles » ;
- Cliquer sur le bouton (repère 3) de la zone « Actions ».

7 - Sauvegarder un scénario

- Cliquer sur le bouton (repère 2) de la zone « Actions ».

8 - Quelles sont les informations enregistrées

EN COURS DE REDACTION

A.2 - Le re-jeu de scénarios

Concept : Re-jeu d'un scénario

Le re-jeu d'un scénario consiste à parcourir un chemin particulier donné par le scénario dans le modèle de tâches K-MAD. Plusieurs aspects peuvent donc être vérifiés :

- Atteignabilité de tâches ;
- Atteignabilité d'un état particulier dans les objets du monde.

EN COURS DE REDACTION

1 - Présentation de l'outil

La Figure 94 présente l'outil de re-jeu. Il se décompose en quatre parties :

- Les actions du scénario à rejouer (repère 1) ;
- Les actions disponibles selon l'état du modèle K-MAD en cours de re-jeu (repère 2) ;
- Le scénario en cours de construction (repère 3) ;
- Une zone action pour contrôler le déroulement du re-jeu (repère 4).

Figure 94 – Zone de « re-jeu »

2 - Présentation de la zone « Scénario à Rejouer »

La zone « scénario à rejouer » liste le scénario à rejouer.

Le format adopté est le suivant :

- Indice qui précise la position de l'action par rapport au scénario ;

- Nom de la tâche
- Action : type de l'action.

Exemple : 1 – Tache A : Action(Exécuter). Première Action à exécuter, appliquée à la tâche Tache A dont le type est Exécuter.

3 - Présentation de la zone « Actions Disponibles »

Cette zone permet de connaître les actions disponibles pour les modèle K-MAD en cours de simulation.

Le format adopté est le suivant :

- Nom de la tâche
- Action : type de l'action.

Exemple : Tache AB : Action(Interrompre). Le nom de la tâche est Tache AB et le type de l'action est Interrompre.

4 - Présentation de la zone « Scénario Courant »

Cette zone permet de connaître le scénario en cours de construction. Cet aspect est en cours de développement, il sera possible de voir les valeurs particulières choisies par l'analyste.

EN COURS DE DEVELOPPEMENT

5 - Présentation de la zone « Contrôle »

Sur la Figure 95 est présentée la zone de contrôle du re-jeu d'un scénario. Nous distinguons les actions pour :

- Ouvrir un scénario ;
- Fermer un scénario ;
- Sauvegarder le scénario courant ;
- Arrêter le re-jeu ;
- Jouer le re-jeu ;
- Mettre sur pause le re-jeu ;
- Retour arrière d'une action ;
- Avancer d'une action.

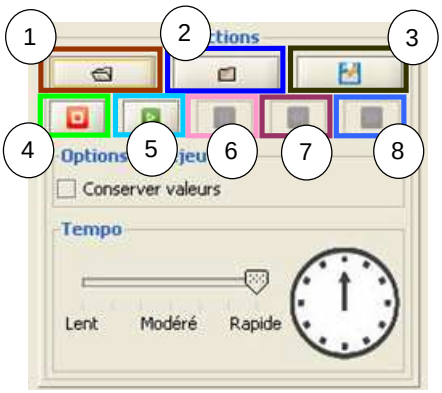

Figure 95 – Zone « Contrôle »

6 - …

EN COURS DE REDACTION

CHAPITRE 9

Préférences de l'outil

Ce chapitre a trait à la configuration de l'outil K-MADe.

A - Général

Ce sous ensemble de paramètres permet de configurer des paramètres généraux.

1 - Description des configurations

La Figure 96 présente l'interface permettant de modifier les paramètres généraux de l'outil K-MADe.

Il est possible de :

- Forcer l'appel le garbage collector
- Changer le pas de zoom
- Configurer l'affichage le Splash screen au démarrage
- Modifier la langue de l'outil. Un redémarrage de l'outil est nécessaire pour que la nouvelle langue soit active. Les langues disponibles sont le français et l'anglais.

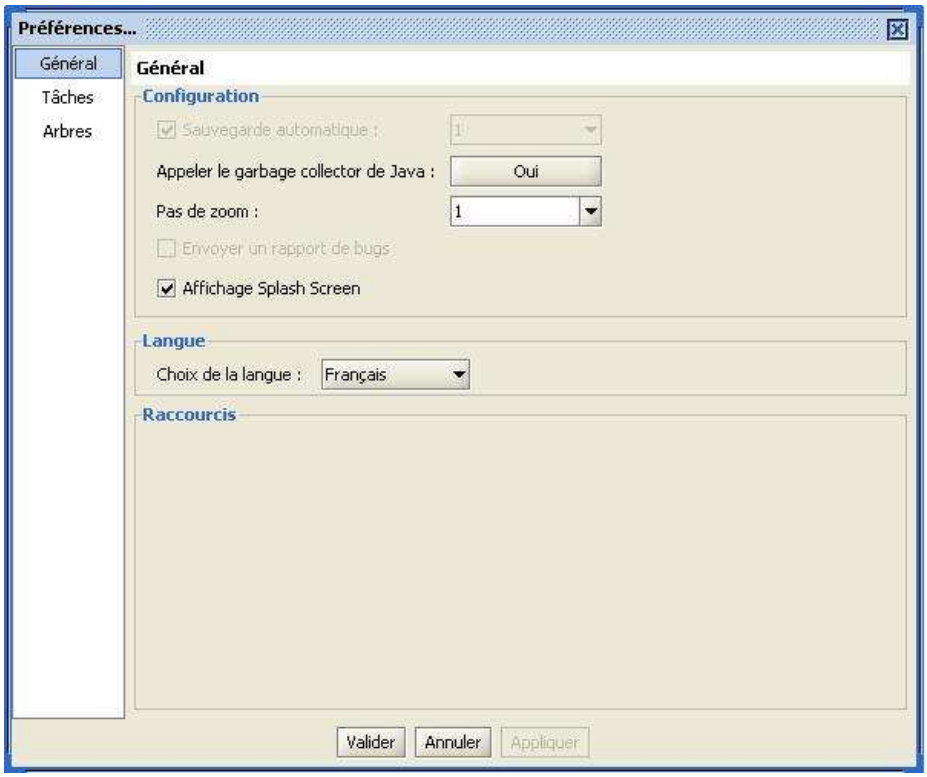

Figure 96 – Interface pour la modification des paramètres généraux

B - Tâches

EN COURS DE DEVELOPPEMENT

C - Arbres

Ce sous ensemble de paramètres permet de modifier l'apparence des arbres de tâches en cours d'édition.

1 - Description de la configuration d'un arbre de tâches

La Figure 97 présente l'interface permettant de modifier les paramètres des arbres de tâches K-MAD.

Il est donc possible de :

- Modifier l'orientation des arbres de tâches (EN COURS DE DEVELOPPEMENT) ;
- Modifier l'espace horizontal entre deux tâches au moment de la re-spatialisation ;
- Modifier l'espace vertical entre des tâches de différents niveaux au moment de la re-spatialisation ;
- Modifier la couleur de sélection des éléments de l'arbre de tâches ;
- Activer / Désactiver l'orthogonalité des liens.

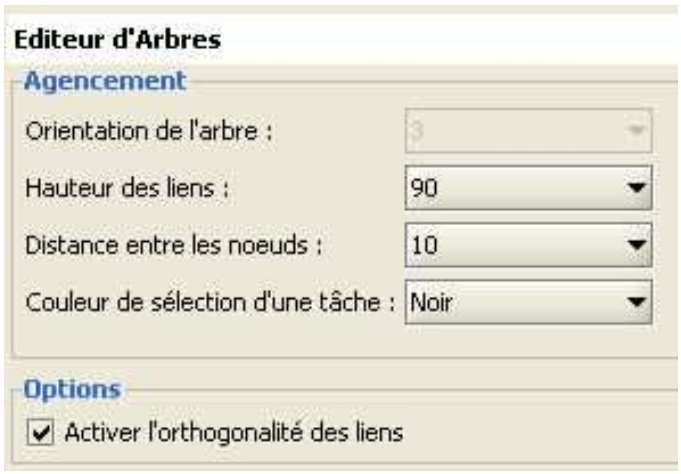

Figure 97 – Interface pour la modification des paramètres des arbres de tâches K-MAD

CHAPITRE 10

Annexes du modèle et de l'outil

Ce chapitre contient un ensemble d'annexes permettant la description du modèle K-MAD et de l'outil associé. Vous trouverez :

- La définition du document XML décrivant un modèle K-MAD ;
- La définition du document XML décrivant un scénario de tâches ;
- Grammaire d'une expression « précondition » ;
- Grammaire d'une expression « action » ;
- Grammaire d'une expression « itération ».

***Note** : les grammaires pour les expressions « précondition », « action » et « itération » sont écrites en utilisant la grammaire de JavaCC (https://javacc.dev.java.net).*

A - Document Type Definition (DTD) du modèle K-MAD

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
      Document Type Definition of K-MAD (Kernel of Model for Activity
Description)
      INRIA Rocquencourt / MErLIn Project
      Authors :
            - BARON Mickael (baron@ensma.fr ou baron.mickael@gmail.com)
            - SCAPIN Dominique (dominique.scapin@inria.fr)
      Version : 1.2
      Date : September 09 2010
-->
<!ELEMENT kmad-model (project, projectinformation, projectinterviews*,
(point | event | actor | user | actorSystem | Individu | Organisation |
Machine | ParcMachines |  task | abstractobject | abstractattribut |
concreteobject | concreteattribut | group | agregat | typevalue | label)*)
>
 <!ATTLIST version version CDATA "2.0">

<!ELEMENT project (id-project-information, id-project-interviews-list?) >
<!ATTLIST project idkmad CDATA #REQUIRED >
<!ATTLIST project classkmad CDATA #REQUIRED >
<!ELEMENT id-project-information (#PCDATA) >
<!ELEMENT id-project-interviews-list (id-project-interview*) >
<!ELEMENT id-project-interview (#PCDATA) >

<!ELEMENT projectinformation (projectinformation-compagny?,
projectinformation-place?, projectinformation-type?, projectinformation-
date?, projectinformation-resources?, projectinformation-motivation?) >
<!ATTLIST projectinformation idkmad CDATA #REQUIRED >
<!ATTLIST projectinformation classkmad CDATA #REQUIRED >
<!ELEMENT projectinformation-compagny (#PCDATA) >
<!ELEMENT projectinformation-place (#PCDATA) >
<!ELEMENT projectinformation-type (#PCDATA) >
<!ELEMENT projectinformation-date (#PCDATA) >
<!ELEMENT projectinformation-resources (#PCDATA) >
<!ELEMENT projectinformation-motivation (#PCDATA) >
```

```
<!ELEMENT projectinterview (projectinterview-actor, projectinterview-
place?, projectinterview-statut?, projectinterview-seniority?,
projectinterview-date?, projectinterview-type?, projectinterview-info?,
projectinterview-director?) >
<!ATTLIST projectinterview idkmad CDATA #REQUIRED >
<!ATTLIST projectinterview classkmad CDATA #REQUIRED >
<!ELEMENT projectinterview-actor (#PCDATA) >
<!ELEMENT projectinterview-place (#PCDATA) >
<!ELEMENT projectinterview-statut (#PCDATA) >
<!ELEMENT projectinterview-seniority (#PCDATA) >
<!ELEMENT projectinterview-date (#PCDATA) >
<!ELEMENT projectinterview-type (#PCDATA) >
<!ELEMENT projectinterview-info (#PCDATA) >
<!ELEMENT projectinterview-director (#PCDATA) >


<!ELEMENT point (point-x, point-y) >
<!ATTLIST point idkmad CDATA #REQUIRED >
<!ATTLIST point classkmad CDATA #REQUIRED >
<!ELEMENT point-x (#PCDATA) >
<!ELEMENT point-y (#PCDATA) >


<!ELEMENT event (event-name, event-description?) >
<!ATTLIST event idkmad CDATA #REQUIRED >
<!ATTLIST event classkmad CDATA #REQUIRED >
<!ELEMENT event-name (#PCDATA) >
<!ELEMENT event-description (#PCDATA) >


<!ELEMENT actor (actor-experience, actor-competence?, id-user) >
<!ATTLIST actor idkmad CDATA #REQUIRED >
<!ATTLIST actor classkmad CDATA #REQUIRED >
<!ELEMENT actor-experience (#PCDATA) >
<!ELEMENT actor-competence (#PCDATA) >
<!ELEMENT id-user (#PCDATA) >

<!ELEMENT actorSystem (actorSystem-experience, actorSystem-competence?, id-
user) >
<!ATTLIST actorSystem idkmad CDATA #REQUIRED >
<!ATTLIST actorSystem classkmad CDATA #REQUIRED >
<!ELEMENT actorSystem-experience (#PCDATA) >
<!ELEMENT actorSystem-competence (#PCDATA) >
<!ELEMENT id-user (#PCDATA) >

<!ELEMENT user (user-name, user-statut?, user-role?, user-imagepath?) >
<!ATTLIST user idkmad CDATA #REQUIRED >
<!ATTLIST user classkmad CDATA #REQUIRED >
<!ELEMENT user-name (#PCDATA) >
<!ELEMENT user-statut (#PCDATA) >
<!ELEMENT user-role (#PCDATA) >
<!ELEMENT user-imagepath (#PCDATA) >

<!ELEMENT Individu (individu-name, individu-statut?, individu-role?,
individu-imagepath?,id-organisation?) >
<!ATTLIST Individu idkmad CDATA #REQUIRED >
<!ATTLIST Individu classkmad CDATA #REQUIRED >
<!ELEMENT individu-name (#PCDATA) >
<!ELEMENT individu-statut (#PCDATA) >
<!ELEMENT individu-role (#PCDATA) >
```

```
<!ELEMENT individu-imagepath (#PCDATA) >
<!ELEMENT id-organisation (#PCDATA) >

<!ELEMENT Organisation (organisation-name, organisation-statut?,
organisation-role?, organisation-imagepath?) >
<!ATTLIST Organisation idkmad CDATA #REQUIRED >
<!ATTLIST Organisation classkmad CDATA #REQUIRED >
<!ELEMENT organisation-name (#PCDATA) >
<!ELEMENT organisation-statut (#PCDATA) >
<!ELEMENT organisation-role (#PCDATA) >
<!ELEMENT organisation-imagepath (#PCDATA) >

<!ELEMENT Machine (machine-name, machine-description?, machine-
isComputer,machine-imagepath?, id-parcMachine?) >
<!ATTLIST Machine idkmad CDATA #REQUIRED >
<!ATTLIST Machine classkmad CDATA #REQUIRED >
<!ELEMENT machine-name (#PCDATA) >
<!ELEMENT machine-description (#PCDATA) >
<!ELEMENT machine-isComputer (#PCDATA) >
<!ELEMENT machine-imagepath (#PCDATA) >
<!ELEMENT id-parcMachine (#PCDATA) >

<!ELEMENT ParcMachines (parcMachines-name, parcMachines-description?,
parcMachines-imagepath?) >
<!ATTLIST ParcMachines idkmad CDATA #REQUIRED >
<!ATTLIST ParcMachines classkmad CDATA #REQUIRED >
<!ELEMENT parcMachines-name (#PCDATA) >
<!ELEMENT parcMachines-description (#PCDATA) >
<!ELEMENT parcMachines-imagepath (#PCDATA) >


<!ELEMENT task (task-name, task-purpose?, task-duration?, id-task-media?,
task-resources?, task-feedback?, task-observation?, task-executant, task-
frequency?, task-compfrequency?, task-importance?, task-modality?, id-task-
eventtrigger?, id-task-events-list?, task-optional, task-interruptible,
task-decomposition, id-task-actors-list?, id-task-subtasks-list?, id-task-
point, id-task-label?, task-precondition, task-descriptionprecondition?,
task-effetsdebord, task-descriptioneffetsdebord?, task-iteration, task-
descriptioniteration?) >
<!ATTLIST task idkmad CDATA #REQUIRED >
<!ATTLIST task classkmad CDATA #REQUIRED >
<!ELEMENT task-name (#PCDATA) >
<!ELEMENT task-purpose (#PCDATA) >
<!ELEMENT task-duration (#PCDATA) >
<!ELEMENT id-task-media (#PCDATA) >
<!ELEMENT task-resources (#PCDATA) >
<!ELEMENT task-feedback (#PCDATA) >
<!ELEMENT task-observation (#PCDATA) >
<!ELEMENT task-executant (#PCDATA) >
<!ELEMENT task-frequency (#PCDATA) >
<!ELEMENT task-compfrequency (#PCDATA) >
<!ELEMENT task-importance (#PCDATA) >
<!ELEMENT task-modality (#PCDATA) >
<!ELEMENT id-task-eventtrigger (#PCDATA) >
<!ELEMENT id-task-events-list (id-task-event*) >
<!ELEMENT id-task-event (#PCDATA) >
<!ELEMENT task-optional (#PCDATA) >
<!ELEMENT task-interruptible (#PCDATA) >
<!ELEMENT task-decomposition (#PCDATA) >
<!ELEMENT id-task-actors-list (id-task-actor*) >
<!ELEMENT id-task-actor (#PCDATA) >
```

```
<!ELEMENT id-task-subtasks-list (id-task-subtask*) >
<!ELEMENT id-task-subtask (#PCDATA) >
<!ELEMENT id-task-point (#PCDATA) >
<!ELEMENT id-task-label (#PCDATA) >
<!ELEMENT task-precondition (#PCDATA) >
<!ELEMENT task-descriptionprecondition (#PCDATA) >
<!ELEMENT task-effetsdebord (#PCDATA) >
<!ELEMENT task-descriptioneffetsdebord (#PCDATA) >
<!ELEMENT task-iteration (#PCDATA) >
<!ELEMENT task-descriptioniteration (#PCDATA) >


<!ELEMENT media (taskmedia-existing, taskmedia-filename?, taskmedia-path?,
taskmedia-startmark?, taskmedia-finishmark?) >
<!ATTLIST media idkmad CDATA #REQUIRED >
<!ATTLIST media classkmad CDATA #REQUIRED >
<!ELEMENT media-existing (#PCDATA) >
<!ELEMENT media-filename (#PCDATA) >
<!ELEMENT media-path (#PCDATA) >
<!ELEMENT media-startmark (#PCDATA) >
<!ELEMENT media-finishmark (#PCDATA) >


<!ELEMENT abstractobject (abstractobject-name, abstractobject-description?)
>
<!ATTLIST abstractobject idkmad CDATA #REQUIRED >
<!ATTLIST abstractobject classkmad CDATA #REQUIRED >
<!ELEMENT abstractobject-name (#PCDATA) >
<!ELEMENT abstractobject-description (#PCDATA) >


<!ELEMENT abstractattribut (abstractattribut-name, abstractattribut-
description?, abstractattribut-typestructure, id-abstractattribut-
abstractobject, id-abstractattribut-type?) >
<!ATTLIST abstractattribut idkmad CDATA #REQUIRED >
<!ATTLIST abstractattribut classkmad CDATA #REQUIRED >
<!ELEMENT abstractattribut-name (#PCDATA) >
<!ELEMENT abstractattribut-description (#PCDATA) >
<!ELEMENT abstractattribut-typestructure (#PCDATA) >
<!ELEMENT id-abstractattribut-abstractobject (#PCDATA) >
<!ELEMENT id-abstractattribut-type (#PCDATA) >


<!ELEMENT concreteobject (concreteobject-name, concreteobject-description?,
id-concreteobject-abstractobject, id-concreteobject-group) >
<!ATTLIST concreteobject idkmad CDATA #REQUIRED >
<!ATTLIST concreteobject classkmad CDATA #REQUIRED >
<!ELEMENT concreteobject-name (#PCDATA) >
<!ELEMENT concreteobject-description (#PCDATA) >
<!ELEMENT id-concreteobject-abstractobject (#PCDATA) >
<!ELEMENT id-concreteobject-group (#PCDATA) >


<!ELEMENT concreteattribut (id-concreteattribut-concreteobject, id-
concreteattribut-abstractattribut, id-concreteattribut-value) >
<!ATTLIST concreteattribut idkmad CDATA #REQUIRED >
<!ATTLIST concreteattribut classkmad CDATA #REQUIRED >
<!ELEMENT id-concreteattribut-concreteobject (#PCDATA) >
<!ELEMENT id-concreteattribut-abstractattribut (#PCDATA) >
<!ELEMENT id-concreteattribut-value (#PCDATA) >
```

```
<!ELEMENT group (group-name, group-description?, id-group-agregat, id-
group-abstractobject) >
<!ATTLIST group idkmad CDATA #REQUIRED >
<!ATTLIST group classkmad CDATA #REQUIRED >
<!ELEMENT group-name (#PCDATA) >
<!ELEMENT group-description (#PCDATA) >
<!ELEMENT id-group-agregat (#PCDATA) >
<!ELEMENT id-group-abstractobject (#PCDATA) >


<!ELEMENT agregat (id-agregat-concreteobjects-list*) >
<!ATTLIST agregat idkmad CDATA #REQUIRED >
<!ATTLIST agregat classkmad CDATA #REQUIRED >
<!ELEMENT id-agregat-concreteobjects-list (id-agregat-concreteobject*) >
<!ELEMENT id-agregat-concreteobject (#PCDATA) >


<!ELEMENT typevalue (typevalue-value) >
<!ATTLIST typevalue idkmad CDATA #REQUIRED >
<!ATTLIST typevalue classkmad CDATA #REQUIRED >
<!ELEMENT typevalue-value (#PCDATA) >


<!ELEMENT label (label-name, label-description?, label-color?, label-
visible?, label-colorvisible?) >
<!ATTLIST label idkmad CDATA #REQUIRED >
<!ATTLIST label classkmad CDATA #REQUIRED >
<!ELEMENT label-name (#PCDATA) >
<!ELEMENT label-description (#PCDATA) >
<!ELEMENT label-color (#PCDATA) >
<!ELEMENT label-visible (#PCDATA) >
<!ELEMENT label-colorvisible (#PCDATA) >
```

B - Document Type Definition (DTD) d'un scénario K-MAD

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
      Document Type Definition of K-MAD Scenario (Kernel of Model for
Activity Description)
      INRIA Rocquencourt / MErLIn Project
      Authors :
            - BARON Mickael (baron@ensma.fr ou baron.mickael@gmail.com)
            - SCAPIN Dominique (dominique.scapin@inria.fr)
      Version : 0.5 Beta
      Date : 01 July 2006
-->

<!ELEMENT scenario (execute | pass | interrupt | resume | cancel)* >
<!ATTLIST scenario name CDATA #REQUIRED >
<!ATTLIST scenario date CDATA #IMPLIED >
<!ATTLIST scenario comment CDATA #IMPLIED >
<!ATTLIST scenario idTask CDATA #REQUIRED >
<!ATTLIST scenario nameTask CDATA #REQUIRED >

<!ELEMENT execute (userExecutingConstraint?, userPreValue*,
userPreConcreteObject*, userPostValue*, userPostConcreteObject*,
userIterValue*, userIterConcreteObject*) >
<!ATTLIST execute idTask CDATA #REQUIRED >
<!ATTLIST execute nameTask CDATA #REQUIRED >

<!ELEMENT userExecutingConstraint EMPTY >
```

```
<!ATTLIST userExecutingConstraint idUser CDATA #REQUIRED >
<!ATTLIST userExecutingConstraint nameUser CDATA #REQUIRED >

<!ELEMENT userPreValue (#PCDATA) >
<!ELEMENT userPreConcreteObject (idUserConcreteObject,
nameUserConcreteObject) >
<!ELEMENT idUserConcreteObject (#PCDATA) >
<!ELEMENT nameUserConcreteObject (#PCDATA) >

<!ELEMENT userPostValue (#PCDATA) >
<!ELEMENT userPostConcreteObject (idUserConcreteObject,
nameUserConcreteObject) >

<!ELEMENT userIterValue (#PCDATA) >
<!ELEMENT userIterConcreteObject (idUserConcreteObject,
nameUserConcreteObject) >

<!ELEMENT pass EMPTY >
<!ATTLIST pass idTask CDATA #REQUIRED >
<!ATTLIST pass nameTask CDATA #REQUIRED >

<!ELEMENT interrupt EMPTY >
<!ATTLIST interrupt idTask CDATA #REQUIRED >
<!ATTLIST interrupt nameTask CDATA #REQUIRED >

<!ELEMENT resume EMPTY >
<!ATTLIST resume idTask CDATA #REQUIRED >
<!ATTLIST resume nameTask CDATA #REQUIRED >

<!ELEMENT cancel EMPTY >
<!ATTLIST cancel idTask CDATA #REQUIRED >
<!ATTLIST cancel nameTask CDATA #REQUIRED >
```

C - Grammaire des préconditions

```
SKIP :
{
  " "
| "\r"
| "\t"
}


TOKEN :
{
      < EXIST : "isExist" >
|     < EXISTAT : "isExistAt" >
|     < EMPTY : "isEmpty" >
|     < NOMBRE : "card" >
|     < GETVALUE : "getValue" >
|     < UNKNOWN_NUMBER : "?NUM" >
|     < UNKNOWN_STRING : "?STR" >
|     < UNKNOWN_BOOLEAN : "?BOOL" >
}

TOKEN:
{
      < PARO : "(" >
|     < PARF : ")" >
|     < COMA : "," >
|     < AND : "AND" >
|     < OR : "OR" >
```

```
|     < EQUAL : "==" >
|     < NOT_EQUAL : "!=" >
|     < INF : "<" >
|     < SUP : ">" >
|     < INF_EQUAL : "<=" >
|     < SUP_EQUAL : ">=" >
|     < NOT : "not" >
|     < EOL : "\n" >
}

TOKEN:
{
      < #DIGIT     : ["0"-"9"] >
|     < #LETTER    : ["A"-"Z","a"-"z",".","é","è", "-", "_", "ç", "à", "@",
"*","µ","£","§","%", "~", "#","{","}","|","`","²"] >
|     < BOOLEAN    : "true" | "false" >
|     < NUMBER : ("-")* ((<DIGIT>)+ | (<DIGIT>)+ "."(<DIGIT>)*) >
|   < NOSPACE    : (<LETTER>) | (<DIGIT>) >
|   < WITHSPACE  : (<NOSPACE> | " ") >
|   < MIDLESPACE : (<NOSPACE>)+ (<WITHSPACE>)* (<NOSPACE>)+ >
|   < FINALSTR   :  <MIDLESPACE> | (<NOSPACE>)+ >
|     < IDENT : "$" <FINALSTR>  >
|     < STRING : "'" (<FINALSTR>)* "'" >
}

NodeExpression expression() : {
      NodeExpression node;
}
{
      < NUMBER > <EOF> { throw new
ParseException(KMADEConstant.NO_FIRST_EXPRESSION_MESSAGE); }
|   < STRING > <EOF> { throw new
ParseException(KMADEConstant.NO_FIRST_EXPRESSION_MESSAGE); }
|   < NUMBER > <NUMBER> { throw new ParseException(); }
|   < UNKNOWN_NUMBER > <EOF> { throw new
ParseException(KMADEConstant.NO_FIRST_EXPRESSION_MESSAGE); }
|   < UNKNOWN_STRING > <EOF> { throw new
ParseException(KMADEConstant.NO_FIRST_EXPRESSION_MESSAGE); }
|     node = conditionalOrExpression() <EOF> {
            if (node instanceof StringConstant || node instanceof
NumberConstant
                        || node instanceof UserNumber || node instanceof
UserString
                        || node instanceof LengthUnaryFunction || node
instanceof LengthFunction) {
                  throw new
ParseException(KMADEConstant.NO_FIRST_EXPRESSION_MESSAGE);
            }
            if (node instanceof GetFunction) {
                  if (!node.isBoolean()) {
                        throw new
ParseException(KMADEConstant.NO_FIRST_EXPRESSION_MESSAGE);
                  }
            }
            return node;
      }
|   <EOF> { throw new ParseException(KMADEConstant.NO_EXPRESSION_MESSAGE);
}
}

NodeExpression conditionalOrExpression() : {
```

```
      NodeExpression left;
      NodeExpression right;
      OrOperator orOp;
}
{
      (left = conditionalAndExpression() ((
      < OR > { orOp = new OrOperator(left); }
  ) right = conditionalAndExpression() {
      if (orOp != null) {
            orOp.setRightNode(right);
            left = orOp;
      }
      })* {
            return left;
      })
}

NodeExpression conditionalAndExpression() : {
      NodeExpression left;
      NodeExpression right;
      AndOperator andOp;
}
{
      left = equalityExpression() ((
      < AND > { andOp = new AndOperator(left); }
  ) right = equalityExpression() {
      if (andOp != null) {
            andOp.setRightNode(right);
            left = andOp;
      }
      })* {
            return left;
      }
}

NodeExpression equalityExpression() : {
      NodeExpression left;
      NodeExpression right;
      EqualityOperator compOp;
}
{
      left = relationExpression() ((
      < EQUAL > { compOp = new EqualOperator(left); }
  | < NOT_EQUAL > { compOp = new DiffOperator(left); }
  ) right = relationExpression() {
      if (compOp != null) {
            compOp.setRightNode(right);
            left = compOp;
      }
  })* {
            return left;
      }
}

NodeExpression relationExpression() : {
      NodeExpression left;
      NodeExpression right;
      RelationalOperator relOp;
}
{
      left = unaryExpressionNotPlusMinus()  ((
```

```
      < INF > { relOp = new InfOperator(left); }
  | < SUP > { relOp = new SupOperator(left); }
  | < INF_EQUAL > { relOp = new InfEqualOperator(left); }
  | < SUP_EQUAL > { relOp = new SupEqualOperator(left); }
  ) right = unaryExpressionNotPlusMinus() {
      if (relOp != null) {
            relOp.setRightNode(right);
            left = relOp;
      }
  })* {
      return left;
  }
}

NodeExpression unaryExpressionNotPlusMinus() :
{ NodeExpression t;}
{
      < NOT > t = primaryExpression() { return new NotUnaryFunction(t); }
|     t = primaryExpression() { return t; }
}

NodeExpression primaryExpression() :
{
      Token myToken;
      NodeExpression node;
}
{
    node = length() { return node; }
|   node = isExist() { return node; }
|   node = isEmpty() { return node; }
|   node = getValue() { return node; }
|   myToken = < STRING > {
            return new
StringConstant(myToken.image.substring(1,myToken.image.length() - 1));
      }
|     myToken =  < NUMBER > {
            return new NumberConstant(new NumberValue(myToken.image));
      }
|   myToken = < BOOLEAN > {
            return new
BooleanConstant(Boolean.parseBoolean(myToken.image));
      }
|   < UNKNOWN_NUMBER > { return new UserNumber(); }
|   < UNKNOWN_STRING > { return new UserString(); }
|   < UNKNOWN_BOOLEAN > { return new UserBoolean(); }
|     <PARO> node = conditionalOrExpression() <PARF> { return new
ParenthesisExpression(node); }
}

NodeExpression getValue() : {
      Token myGroupToken;
      Token myAttrToken;
}
{
      < GETVALUE > < PARO > myGroupToken = < IDENT > < COMA > myAttrToken =
< IDENT > < PARF > {
            try {
                  GroupExpressExpression myGroupRef = new
GroupExpressExpression(myGroupToken.image.substring(1).toLowerCase());
```

```
                  AttributExpressExpression myAttrRef = new
AttributExpressExpression(myAttrToken.image.substring(1).toLowerCase(),
myGroupRef);
                  return new GetFunction(myGroupRef,myAttrRef);
            } catch (ExpressException e) {
                  throw new Error(e.toString());
            }
      }
}

NodeExpression getValueDouble(GroupExpressExpression ref) : {
      Token myGroupOrAttributToken;
      Token myAttrToken = null;
}
{
      < GETVALUE > < PARO > myGroupOrAttributToken = < IDENT >
      (< COMA > myAttrToken = < IDENT >)? < PARF > {
            GroupExpressExpression myGroupRef = null;
            AttributExpressExpression myAttrRef = null;

            try {
                  if (myAttrToken == null) {
                        myGroupRef = ref;
                        myAttrRef = new
AttributExpressExpression(myGroupOrAttributToken.image.substring(1).toLower
Case(), myGroupRef);
                        return new GetUnaryFunction(myAttrRef);
                  } else {
                        myGroupRef = new
GroupExpressExpression(myGroupOrAttributToken.image.substring(1).toLowerCas
e());
                        myAttrRef = new
AttributExpressExpression(myAttrToken.image.substring(1).toLowerCase(),
myGroupRef);
                        return new GetFunction(myGroupRef,myAttrRef);
                  }
            } catch (ExpressException e) {
                  throw new Error(e.toString());
            }
      }
}

NodeExpression isExist() :
{
      Token myIdentToken;
      Token function;
      GroupExpressExpression myGroupRef;
      NodeExpression node;
}
{
      ( function = < EXIST >
|      function = < EXISTAT >)
      < PARO > myIdentToken = < IDENT > {
            try {
                  myGroupRef = new
GroupExpressExpression(myIdentToken.image.substring(1).toLowerCase());
            } catch (ExpressException e) {
                  throw new Error(e.toString());
            }
      }
```

```
      < COMA > node = conditionalOrExpressionFunction(myGroupRef) < PARF >
{
            if (function.kind == EXIST) {
                  return new IsExistFunction(myGroupRef, node);
            } else {
                  return new IsExistAtFunction(myGroupRef, node);
            }
      }
}

NodeExpression isEmpty() :
{
      Token myIdentToken;
}
{
      < EMPTY > < PARO > myIdentToken = < IDENT > < PARF > {
            try {
                  return new IsEmptyUnaryFunction(new
GroupExpressExpression(myIdentToken.image.substring(1).toLowerCase()));
            } catch (ExpressException e) {
                  throw new Error(e.toString());
            }
      }
}

NodeExpression length() :
{
      Token myIdentToken;
      NodeExpression node = null;
      GroupExpressExpression myGroupRef;
}
{
      < NOMBRE > < PARO > myIdentToken = < IDENT > {
            try {
                  myGroupRef = new
GroupExpressExpression(myIdentToken.image.substring(1).toLowerCase());
            } catch (ExpressException e) {
                  throw new Error(e.toString());
            }
      }
      ( < COMA > node = conditionalOrExpressionFunction(myGroupRef) )? <
PARF > {
            if (node == null) {
                  return new LengthUnaryFunction(myGroupRef);
            } else {
                  return new LengthFunction(myGroupRef,node);
            }
      }
}

NodeExpression conditionalOrExpressionFunction(GroupExpressExpression ref)
: {
      NodeExpression left;
      NodeExpression right;
      OrOperator orOp;
}
{
      left = conditionalAndExpressionFunction(ref) ((
      < OR > { orOp = new OrOperator(left); }
  ) right = conditionalAndExpressionFunction(ref) {
      if (orOp != null) {
```

```
            orOp.setRightNode(right);
            left = orOp;
      }
      })* {
            return left;
      }
}

NodeExpression conditionalAndExpressionFunction(GroupExpressExpression ref)
: {
      NodeExpression left;
      NodeExpression right;
      AndOperator andOp;
}
{
      left = equalityExpressionFunction(ref) ((
      < AND > { andOp = new AndOperator(left); }
  ) right = equalityExpressionFunction(ref) {
      if (andOp != null) {
            andOp.setRightNode(right);
            left = andOp;
      }
      })* {
            return left;
      }
}

NodeExpression equalityExpressionFunction(GroupExpressExpression ref) : {
      NodeExpression left;
      NodeExpression right;
      EqualityOperator compOp;
}
{
      left = relationExpressionFunction(ref) ((
      < EQUAL > { compOp = new EqualOperator(left); }
  | < NOT_EQUAL > { compOp = new DiffOperator(left); }
  ) right = relationExpressionFunction(ref) {
      if (compOp != null) {
            compOp.setRightNode(right);
            left = compOp;
      }
  })* {
            return left;
      }
}

NodeExpression relationExpressionFunction(GroupExpressExpression ref) : {
      NodeExpression left;
      NodeExpression right;
      RelationalOperator relOp;
}
{
      left = unaryExpressionNotPlusMinusFunction(ref)  ((
      < INF > { relOp = new InfOperator(left); }
  | < SUP > { relOp = new SupOperator(left); }
  | < INF_EQUAL > { relOp = new InfEqualOperator(left); }
  | < SUP_EQUAL > { relOp = new SupEqualOperator(left); }
  ) right = unaryExpressionNotPlusMinusFunction(ref) {
      if (relOp != null) {
            relOp.setRightNode(right);
            left = relOp;
```

```
      }
  })* {
      return left;
  }
}

NodeExpression unaryExpressionNotPlusMinusFunction(GroupExpressExpression
ref) :
{ NodeExpression t;}
{
      < NOT > t = primaryExpressionFunction(ref) { return new
NotUnaryFunction(t); }
|     t = primaryExpressionFunction(ref) { return t; }
}

NodeExpression primaryExpressionFunction(GroupExpressExpression ref) :
{
      Token myToken;
      NodeExpression node;
}
{
    length() { return null; }
|   node = isExist() { return node; }
|   node = isEmpty() { return node; }
|   node = getValueDouble(ref) { return node; }
|     myToken = < IDENT > {
            try {
                  return new
AttributExpressExpression(myToken.image.substring(1).toLowerCase(), ref);

            } catch (ExpressException e) {
                  throw new Error(e.toString());
            }
      }
|   myToken = < STRING > {
            return new
StringConstant(myToken.image.substring(1,myToken.image.length() - 1));
      }
|     myToken =  < NUMBER > {
            return new NumberConstant(new NumberValue(myToken.image));
      }
|   myToken = < BOOLEAN > {
            return new
BooleanConstant(Boolean.parseBoolean(myToken.image));
      }
|   < UNKNOWN_NUMBER > { return new UserNumber(); }
|   < UNKNOWN_STRING > { return new UserString(); }
|   < UNKNOWN_BOOLEAN > { return new UserBoolean(); }
|     <PARO> node = conditionalOrExpressionFunction(ref) <PARF> { return
new ParenthesisExpression(node); }
}
```

D - Grammaire des actions

```
SKIP :
{
  " "
| "\r"
| "\t"
}
```

```
TOKEN :
{
      < REMOVE : "remove" >
|     < SET : "set" >
|     < ADD : "add" >
|   < CREATE : "create" >
|   < REPLACE : "replace" >
|   < GETVALUE : "getValue" >
|   < UNKNOWN_NUMBER : "?NUM" >
|   < UNKNOWN_STRING : "?STR" >
|   < UNKNOWN_BOOLEAN : "?BOOL" >
|   < VOID_TYPE : "void" >
}

TOKEN:
{
      < PARO : "(" >
|     < PARF : ")" >
|     < COMA : "," >
|     < SEQUENCE : ";" >
|     < ASSIGNMENT : ":=" >
|   < PLUS : "+" >
|   < MINUS : "-" >
|   < PLUS_ASSIGNMENT : "+=" >
|   < MINUS_ASSIGNMENT : "-=" >
|   < EOL : "\n" >
}

TOKEN:
{
      < #DIGIT     : ["0"-"9"] >
|     < #LETTER    : ["A"-"Z","a"-"z",".","é","è", "-", "_", "ç", "à", "@",
"*","µ","£","§","%", "~", "#","{","}","|","`","²"] >
|     < BOOLEAN    : "true" | "false" >
|     < NUMBER : ("-")* ((<DIGIT>)+ | (<DIGIT>)+ "."(<DIGIT>)*) >
|   < NOSPACE    : (<LETTER>) | (<DIGIT>) >
|   < WITHSPACE  : (<NOSPACE> | " ") >
|   < MIDLESPACE : (<NOSPACE>)+ (<WITHSPACE>)* (<NOSPACE>)+ >
|   < FINALSTR   :  <MIDLESPACE> | (<NOSPACE>)+ >
|     < IDENT : "$" <FINALSTR>  >
|     < STRING : "'" (<FINALSTR>)* "'" >
}


NodeExpression expression() : {
      NodeExpression node;
} {
      node = sequentialExpression() <EOF> { return node; }
|   < VOID_TYPE > <EOF > { return new VoidConstant(); }
|     <EOF> { throw new
ParseException(KMADEConstant.NO_EXPRESSION_MESSAGE); }
}

NodeExpression sequentialExpression() : {
      NodeExpression left;
      NodeExpression right;
      NotComma notComma;
} {
      left = primaryExpression() ( <SEQUENCE> { notComma = new
NotComma(left); }
```

```
      right = primaryExpression() {
            if (notComma != null) {
                  notComma.setRightNode(right);
                  left = notComma;
            }
      })* {
            return left;
      }
}

NodeExpression primaryExpression() : {
      NodeExpression node;
} {
      node = remove() { return node; }
|     node = set() { return node; }
|     node = add() { return node; }
|     node = replace() { return node; }
|   node = create() { return node; }
}

NodeExpression remove() : {
      Token myIdentToken;
      GroupExpressExpression myGroupRef;
      NodeExpression node;
} {
      < REMOVE > < PARO > myIdentToken = < IDENT > < PARF > {
            try {
                  return new RemoveUnaryFunction(new
GroupExpressExpression(myIdentToken.image.substring(1).toLowerCase()));
            } catch (ExpressException e) {
                  throw new Error(e.toString());
            }
      }
}

NodeExpression set() : {
      AssignmentOperator nodeAssignment;
      NodeExpression setFunction;
} {
      < SET > < PARO > (
      nodeAssignment = withGroup() {
            setFunction = new
SetFunction(((AttributExpressExpression)nodeAssignment.getLeftNode()).getGr
oupExpressExpression(), nodeAssignment);
      }
|     nodeAssignment = onlyAssignment(null) {
            setFunction = new SetUnaryFunction(nodeAssignment); // null
Group from Attribut
      })
      < PARF > {
            return setFunction;
      }
}

AssignmentOperator withGroup() : {
      Token myIdentToken;
      AssignmentOperator node;
      GroupExpressExpression myExpression;
} {
      myIdentToken = < IDENT > {
            try {
```

```
                  myExpression = new
GroupExpressExpression(myIdentToken.image.substring(1).toLowerCase());

            } catch (ExpressException e) {
                  throw new Error(e.toString());
            }
      } < COMA > node = onlyAssignment(myExpression) {
            return node;
      }
}

NodeExpression add() : {
      Token myIdentToken;
} {
      < ADD > < PARO > myIdentToken = < IDENT > < PARF > {
            try {
                  return new AddUnaryFunction(new
GroupExpressExpression(myIdentToken.image.substring(1).toLowerCase()));
            } catch (ExpressException e) {
                  throw new Error(e.toString());
            }
      }
}

NodeExpression create() : {
      Token myIdentToken;
      Token myNewConcret;
      NodeExpression refNode;
      GroupExpressExpression myExpression;
} {
      < CREATE > < PARO > myIdentToken = < IDENT > {
            try {
                  myExpression = new
GroupExpressExpression(myIdentToken.image.substring(1).toLowerCase());

            } catch (ExpressException e) {
                  throw new Error(e.toString());
            }
      } < COMA > refNode = sumAssignment(myExpression) < PARF > {
            return new CreateFunction(myExpression, refNode);
      }
}

NodeExpression replace() : {
      Token myFirstIdentToken;
      Token mySecondIdentToken;
} {
      < REPLACE > < PARO > myFirstIdentToken = < IDENT > < COMA >
mySecondIdentToken = < IDENT > < PARF > {
            try {
                  return new ReplaceFunction(new
GroupExpressExpression(myFirstIdentToken.image.substring(1).toLowerCase()),
                        new
GroupExpressExpression(mySecondIdentToken.image.substring(1).toLowerCase())
);
            } catch (ExpressException e) {
                  throw new Error(e.toString());
            }
      }
}
```

```
AssignmentOperator onlyAssignment(GroupExpressExpression ref) : {
      Token myIdentToken;
      AttributExpressExpression myIdentNode;
      AssignmentOperator refAOp;
      NodeExpression myExpression;
} {
      myIdentToken = < IDENT > {
            if (ref == null) {
                  myIdentNode = new
AttributExpressExpression(myIdentToken.image.substring(1).toLowerCase());

            } else {
                  try {
                        myIdentNode = new
AttributExpressExpression(myIdentToken.image.substring(1).toLowerCase(),ref
);
                  } catch (ExpressException e) {
                        throw new Error(e.toString());
                  }
            }
      }(
            < ASSIGNMENT > { refAOp = new Assignment(myIdentNode); }
      |   < PLUS_ASSIGNMENT > { refAOp = new PlusAssignment(myIdentNode); }
      |   < MINUS_ASSIGNMENT > { refAOp = new MinusAssignment(myIdentNode);
}
      ) ( myExpression = sumAssignment(ref) ) {
            refAOp.setRightNode(myExpression);
            return refAOp;
      }
}

NodeExpression sumAssignment(GroupExpressExpression ref) : {
      Token operator;
      NodeExpression left;
      NodeExpression right;
} {
      left = primaryAssignment(ref) (
            (operator = < PLUS > | operator = < MINUS > ) right =
primaryAssignment(ref) {
                  if (operator.kind == PLUS) {
                        left = new PlusComputing(left,right);
                  } else {
                        left = new MinusComputing(left,right);
                  }
            }
      )* {
            return left;
      }
}

NodeExpression primaryAssignment(GroupExpressExpression ref) : {
      NodeExpression node;
      Token myToken;
} {
      node = getValueDouble(ref) { return node; }
|   myToken = < STRING > {
            return new
StringConstant(myToken.image.substring(1,myToken.image.length() - 1));
      }
|     myToken =  < NUMBER > {
            return new NumberConstant(new NumberValue(myToken.image));
```

```
      }
|   myToken = < BOOLEAN > {
            return new
BooleanConstant(Boolean.parseBoolean(myToken.image));
      }
|     < UNKNOWN_NUMBER > { return new UserNumber(); }
|   < UNKNOWN_STRING > { return new UserString(); }
|   < UNKNOWN_BOOLEAN > { return new UserBoolean(); }
|     <PARO> node = sumAssignment(ref) <PARF> { return new
ParenthesisExpression(node); }
}

NodeExpression getValueDouble(GroupExpressExpression ref) : {
      Token myGroupOrAttributToken;
      Token myAttrToken = null;
}
{
      < GETVALUE > < PARO > myGroupOrAttributToken = < IDENT >
      (< COMA > myAttrToken = < IDENT >)? < PARF > {
            GroupExpressExpression myGroupRef = null;
            AttributExpressExpression myAttrRef = null;

            try {
                  if (myAttrToken == null) {
                        myGroupRef = ref;
                        myAttrRef = new
AttributExpressExpression(myGroupOrAttributToken.image.substring(1).toLower
Case(), myGroupRef);
                        if (myGroupRef == null) {
                              return new GetUnaryFunction(myAttrRef);
                        } else {
                              return new GetFunction(myGroupRef,myAttrRef);
                        }
                  } else {
                        myGroupRef = new
GroupExpressExpression(myGroupOrAttributToken.image.substring(1).toLowerCas
e());
                        myAttrRef = new
AttributExpressExpression(myAttrToken.image.substring(1).toLowerCase(),
myGroupRef);
                        return new GetFunction(myGroupRef,myAttrRef);
                  }
            } catch (ExpressException e) {
                  throw new Error(e.toString());
            }
      }
}
```

Grammaire des SKIP :

```
{
  " "
| "\r"
| "\t"
}


TOKEN :
{
      < WHILE : "while" >
|     < NOTWHILE : "notWhile" >
}
```

```
TOKEN:
{
      < CROO : "[" >
|     < CROF : "]" >
}

TOKEN:
{
< #DIGIT : ["0"-"9"] >
|     < #LETTER : ["A"-"Z","a"-"z"," ","é","è", "-", "_", "ç", "à", "@",
"$","*","µ","£","§","%", "~", "#","{","}","|","`","²"] >
|     < NUMBER : ("-")* ((<DIGIT>)+ | (<DIGIT>)+ "."(<DIGIT>)*) >
|   < STRING_LITERAL : "(" ( < LETTER >
      | < DIGIT >
      | "'"
      | "$"
      | "?"
      | ">"
      | "<"
      | ","
      | "!"
      | "="
      | "("
      | ")"
      | " " )+ ")" >
}

NodeExpression expression() : {
      NodeExpression node;
} {
      node = iterationExpression() <EOF> { return node; }
|   <EOF> { throw new ParseException(KMADEConstant.NO_EXPRESSION_MESSAGE);
}
}

NodeExpression iterationExpression() : {
      NodeExpression node;
} {
      node = iterationNatural() { return node; }
|   node = iterationFunction() { return node; }
}

NodeExpression iterationNatural() : {
      Token myToken;
} {
      < CROO > myToken = < NUMBER > < CROF > {
            return new NumberVariant(new NumberValue(myToken.image));
      }
}

NodeExpression iterationFunction() : {
      Token myString;
} {
      < WHILE > myString = < STRING_LITERAL > {
            String momo =
myString.image.substring(1,myString.image.length() - 1);
            java.io.StringReader sr = new java.io.StringReader(momo);
          java.io.Reader r = new java.io.BufferedReader(sr);
      Precondition parser = new Precondition(r);
      NodeExpression toto = parser.expression();
            return new WhileUnaryFunction(toto);
```

```
      }
|     < NOTWHILE > myString = < STRING_LITERAL > {
            String momo =
myString.image.substring(1,myString.image.length() - 1);
            java.io.StringReader sr = new java.io.StringReader(momo);
          java.io.Reader r = new java.io.BufferedReader(sr);
      Precondition parser = new Precondition(r);
      NodeExpression toto = parser.expression();
            return new NotWhileUnaryFunction(toto);
      }
}
```

CHAPITRE 11

Exemple(s)

EN COURS DE REDACTION

CHAPITRE 12

Historique et Bibliographie

Les touts premiers travaux en matière de modélisation des tâches pour l'IHM (Scapin, 1988) avaient plusieurs objectifs: considérer la façon dont l'utilisateur se représente sa tâche, et non pas la logique du traitement informatique, ou la tâche "prescrite" ; prendre en compte les aspects conceptuels et sémantiques, et pas seulement les aspects syntaxiques et lexicaux ; arriver à une décomposition structurelle des tâches de façon uniforme ; effectuer une description aussi bien d'un point de vue déclaratif (état des choses) que procédural (façon d'arriver à ces états) ; autoriser la prise en compte du parallélisme et pas seulement du séquentiel (synchronisation des tâches) ; et viser un caractère implémentable. Ces travaux ont été menés dans une problématique d'organisation de la connaissance ergonomique et de définition des moyens de la mettre en oeuvre en évaluation et en conception (Scapin, 1990).

À partir de ces premières idées et diverses exigences, un premier formalisme de description des tâches ("MAD" : Méthode Analytique de description des Tâches) a été proposé (Scapin, D. L. & Pierret-Golbreich, C., 1989, 1990) à l'intersection de l'ergonomie, de l'informatique, et de l'intelligence artificielle. De plus, l'outil EMAD a été développé (Pierret, C., Delouis, I. & Scapin, D. L., 1989) ; Delouis I., Pierret C., 1991).

En parallèle, une méthode de recueil d'information a été définie (Sebillotte S., 1991) ; elle consiste principalement en des entretiens semi-directifs basée sur la technique du « Pourquoi ? » / « Comment ? ». De plus, une méthodologie pratique d'analyse de la tâche en vue d'extraire des caractéristiques pertinentes pour la conception d'interfaces a également été proposée (Sebillotte, S. & Scapin, D. L., 1994; Sebillotte, S. et al., 1994).

Ensuite, les travaux ont été poursuivis selon un processus itératif : élaboration (par exemple : implémentation du modèle dans un formalisme objet, organisation des tâches en différents niveaux d'abstraction), validation sur le terrain (domotique, résolution d'incendies maritimes, contrôle aérien, régulation ferroviaire, etc.) modifications, validation sur le terrain (voir : Hammouche, H., 1995 ; Alonso, B., 1996 ; Fallah, D., 1998).

Les travaux (Gamboa-Rodriguez, F. & Scapin, D. L., 1997; Gamboa-Rodriguez, F., 1998) ont également concerné l'aide à la spécification d'interfaces : construction de deux modèles : un modèle des tâches opérateur (MAD*) et un modèle de conception d'interfaces (ICS). Il s'est agit d'implémenter la formalisation du modèle des tâches, de spécifier le modèle conceptuel d'interfaces et de définir les procédures permettant de passer de l'un à l'autre, en faisant la liaison avec la réalisation pratique d'interfaces (via la boîte à outils ILOG-Views). Les travaux ont conduit à l'implémentation d'un éditeur de description des tâches de l'utilisateur EMAD* et à l'atelier logiciel de conception (ALACIE).

La problématique de l'analyse des tâches et l'état du modèle MAD à ce stade ont été décrits dans Scapin & Bastien (2001).

L'ensemble des travaux précédents ont été menés à l'INRIA (projet de Psychologie Ergonomique pour l'Informatique, puis projet MErLIn, http://www.inria.fr/recherche/equipes/merlin.fr.html), en collaboration, notamment co-encadrement de thèses avec les universités de Paris 6, Paris 11, Paris 5.

Les travaux qui ont amené à la version actuelle du modèle et des outils ont également été menés à l'INRIA. Le nouveau modèle (K-MAD), suite à un examen critique des modèles existants (dont MAD et MAD*), a été conçu par V. Lucquiaud (bourse INRIA) lors de ses des travaux de thèse (encadrement D. L. Scapin, INRIA ; F. Jambon et P. Girard, U. Poitiers) ; ainsi que la première maquette d'outil, à laquelle a contribué D. Autard (ACET INRIA). La version actuelle de l'outil a été développée par M. Baron (post-Doc INRIA).

For the tool: M. Baron, V. Lucquiaud, D. Autard, et D. L. Scapin (2006). K-MADe : un environnement pour le noyau du modèle de description de l'activité. submitted to IHM 2006. Proceedings of the 18th French-speaking conference on Human-computer interaction (Conférence Francophone sur l'Interaction Homme- Machine) ; Montréal ; Canada.

For the model: M. Lucquiaud, V. (2005). Proposition d'un noyau et d'une structure pour les modèles de tâches orientés utilisateurs ; Proceedings of the 17th French-speaking conference on Human-computer interaction (Conférence Francophone sur l'Interaction HommeMachine) ; Toulouse ; France ; pp 83-90 ; 2005.

Bibliographie

Delouis I., & Pierret C. (1991) : *EMAD, manuel de référence*, Rapport interne INRIA.

Gamboa-Rodriguez, F. & Scapin, D. L. (1997). Editing MAD* task descriptions for specifying user interfaces, at both semantic and presentation levels, in Proceedings DSV-IS '97, 4th International Eurographics Workshop on Design, Specification, and Verification of Interactive Systems, Granada, Spain, 4-6 June 1997

Pierret, C., Delouis, I. & Scapin, D. L. (1989). *Un outils d'acquisition et de representation des tâches orienté-objets* (Rapport de recherche N° 1063). Rocquencourt, France : Institut National de Recherche en Informatique et en Automatique.

Lucquiaud, V., Scapin, D. L. et Jambon, F. (2002). *Outils de modélisation des tâches utilisateurs : exigences du point de vue utilisation.* IHM 2002, Poitiers, 26-29 Novembre 2002.

Scapin, D. L. (1988). *Vers des outils formels de description des tâches orientés conception d'interfaces* (Rapport de recherche N° 893). Rocquencourt, France : Institut National de Recherche en Informatique et en Automatique.

Scapin, D. L. et Bastien, J. M. C., (2001). *Analyse des tâches et aide ergonomique à la conception : l'approche MAD** in Systèmes d'information et Interactions homme-machine, C. Kolski (Ed.), Hermés

Scapin, D. L. & Pierret-Golbreich, C. (1990). Towards a method for task description. In L. Berlinguet and D. Berthelette (Eds.), *Work with Display Unit.* Amsterdam *:* Elsevier, 371-380.

Sebillotte S. (1991) : Décrire des tâches selon les objectifs des opérateurs, de l'interview à la formalisation, Le Travail Humain 54,3 p193-223

Sebillotte, S. & Scapin, D. L. (1994). From users' task knowledge to high level interface specification. International Journal of Human-computer Interaction, 6, 1-15.

INDEX

L

M

N

O

P

R

S

T

U

V

TABLE DES ILLUSTRATIONS