Décision multicritère à base de traces pour les applications interactives à exécution adaptative

Hoang Nam Ho

UNIVERSITÉ DE LA ROCHELLE – LABORATOIRE L3I

ÉCOLE DOCTORALE S2IM

SCIENCES ET INGÉNIERIE POUR L'INFORMATION, MATHÉMATIQUES

THÈSE

pour obtenir le titre de

Docteur de l'Université de La Rochelle

Mention: Informatique et Application

Présenté et soutenu par

HO Hoang Nam

Décision multicritère à base de traces pour les applications interactives à exécution adaptative

Thèse dirigée par

Pascal ESTRAILLIER, Mourad RABAH

préparée au Laboratoire L3i

Soutenue le 4 Décembre 2015

Jury:

Rapporteurs :	Patrick Brézillon, Professeur	Université Pierre et Marie Curie
	Karim Sehaba, HDR	Université Lumière – Lyon 2
Directeur :	Pascal Estraillier, Professeur	Université de La Rochelle
Co-directeur :	Mourad Rabah, MCF	Université de La Rochelle
Membres :	Samuel Nowakowski, HDR	Université de Lorraine
	Tristan Cazenave, Professeur	Université Paris-Dauphine

Remerciements

Je tiens tout d'abord à remercier sincèrement Monsieur Pascal Estraillier, mon directeur de thèse, qui m'a donné le goût de la recherche. Il m'a permis de découvrir le domaine de l'Interactivité, plus particulièrement celui des Systèmes Interactifs Adaptatifs tout au long du travail de thèse. Je le remercie pour sa gentillesse ainsi que pour l'aide et les conseils qu'il m'a donnés malgré ses charges académiques et professionnelles. Aujourd'hui, j'achève des travaux de thèse, ce qui n'aurait pas été possible sans lui.

Mes remerciements vont aussi à Mourad Rabah et Samuel Nowakowski qui m'ont co-encadré durant toute cette thèse. Mon travail, notamment la rédaction d'articles de recherche et de ce manuscrit n'aurait pas été le même sans eux. Grâce à eux et aux nombreuses relectures de ce document, mon travail de rédaction a pu être amélioré.

Je tiens à remercier les rapporteurs de ma thèse : Patrick Brézillon pour avoir accepté de plonger dans les profondeurs de ma thèse et Karim Sehaba pour avoir décortiqué son contenu. Je tiens aussi à remercier Tristan Cazenave d'avoir accepté de me faire l'honneur de participer à mon jury de thèse.

Je remercie également de tout mon cœur ma chérie Ty pour m'avoir encouragé, pour avoir vécu au rythme du travail, pour avoir partagé les moments difficiles durant les années de thèse.

Des remerciements particuliers pour mes parents au Vietnam, leur soutien et leur confiance en moi durant les longues années d'études à l'étranger.

Enfin, j'adresse mes plus chaleureux remerciements à tous mes amis qui ont déjà ou pas encore soutenu au laboratoire L3i. Ils ont aussi contribué chacun à sa façon à faire que ce travail soit possible. Je n'oublie pas Kathy, notre charmante secrétaire du laboratoire et je remercie particulièrement les chercheurs de la dynamique POLARIS qui m'ont toujours soutenu et encouragé au cours de la thèse.

Un gros merci à toutes et à tous.

Table des matières

Liste des figures

Liste des tableaux

Partie 1 : Introduction et Positionnement

CHAPITRE 1 CONTEXTE, OBJECTIFS GENERAUX DE LA THESE

SOMMAIRE

Ce chapitre introduit le contexte dans lequel nous avons mené nos travaux de thèse : celui des systèmes interactifs à exécution adaptative. Nous nous focalisons sur les interactions entre le système et les utilisateurs pour bien adapter l'exécution de l'application. Nos travaux ont, en effet, pour objectif principal la recherche d'un optimum parmi un ensemble de solutions existantes en vue d'adapter l'exécution en respectant les buts définis de l'application.

Une application interactive peut être définie comme un ensemble d'actions et d'événements réalisés par l'utilisateur et le système. Dans nos travaux, nous faisons l'hypothèse que l'application est décrite par un ensemble de « situations » [1] et, par conséquent, l'exécution de l'application repose sur un enchainement de « situations ». En positionnant avec l'objectif identifié, nous déduirons ainsi les problématiques à traiter tout au long de la thèse.

1.1 CONTEXTE GENERAL

L'avènement des Technologies de l'Information et de la Communication (TIC) représente une véritable opportunité pour le développement des systèmes informatiques. De la machine de Turing à la machine d'interaction, de nombreux résultats ont déjà été obtenus qui permettent de faciliter la mise en place des systèmes informatiques de plus en plus puissants et flexibles. Les travaux de recherche s'inscrivent à la fois dans le cadre du système et dans celui de l'utilisateur. En effet, les nouveaux systèmes informatiques sont de plus en plus axés sur l'interactivité, où le système ne fonctionne pas seulement par lui-même, mais demande également à un ou à plusieurs acteurs (soit utilisateurs, un agents virtuels) de participer à la réalisation des tâches du système. La conception de ces systèmes prend donc en compte l'ensemble des acteurs qui participent à l'exécution du système. Particulièrement, la conception est centrée sur l'utilisateur en lui proposant un environnement adaptatif, dynamique et recommandant afin de s'adapter en fonction de ses comportements ou de ses besoins. Dans la suite du manuscrit, nous appellerons un tel système : *système interactif adaptatif*.

1.1.1 SYSTEME INTERACTIF ADAPTATIF

Dans un premier temps, nous introduisons les notions centrales de nos travaux à savoir l'interactivité et l'adaptabilité.

1.1.1.1 Notion d'interactivité

Dans ces travaux, nous ne considérerons donc qu'un système est interactif s'il permet les interactions avec un ou plusieurs utilisateurs humains. Une interaction peut-être vue comme une action d'entrée/sortie entre un utilisateur et le système nécessaire à la progression de l'activité de l'application considérée. Dans notre contexte, nous ne considérons donc qu'un système interactif dans lequel les utilisateurs interagissent avec le système par la réalisation d'interactions.

La notion d'interactivité indique la capacité interactive d'un système. Théoriquement, selon [2], l'interactivité peut être définie comme « *le niveau auquel une technologie de communication peut créer un environnement médiateur dans lequel les participants peuvent communiquer (1 vers 1, 1 vers n ou n vers n), de manière à la fois synchrone et asynchrone, et participer aux échanges des messages réciproques. À propos d'utilisateurs humains, elle concerne leur*

aptitude à percevoir l'expérience comme une simulation de communication interpersonnelle et d'augmenter leurs connaissances de télé-présence ».

Un système interactif permet aux utilisateurs d'interagir à travers les interfaces homme-machine pendant le déroulement de l'exécution d'une application interactive.

1.1.1.2 Notion d'adaptation

Dans nos travaux, nous considérons qu'un système est adaptatif s'il a la capacité de changer automatiquement ses caractéristiques selon les besoins et les préférences de l'utilisateur. Selon Jameson [3], « *un système qui peut s'adapter à l'utilisateur est un système interactif qui adapte ses tâches à chaque utilisateur en fonction des informations dont il dispose sur cet utilisateur* ». L'adaptation d'un système interactif se base sur les informations collectées lors des interactions et elle est réalisée pendant que l'utilisateur interagit avec le système [4].

L'adaptation des applications interactives peut être effectuée principalement à deux niveaux : soit au niveau de l'Interface Homme-Machine (IHM), soit au niveau du noyau fonctionnel de l'application indépendant de l'IHM. L'adaptation de l'IHM dénote la capacité d'ajuster ou de changer tout ou partie des interfaces homme-machine centrée sur l'utilisateur. L'adaptation au niveau du noyau fonctionnel se concentre sur le processus d'exécution ou la logique d'exécution définie au cœur de l'application interactive. La Figure 1 décrit une architecture générale d'un système interactif adaptatif.

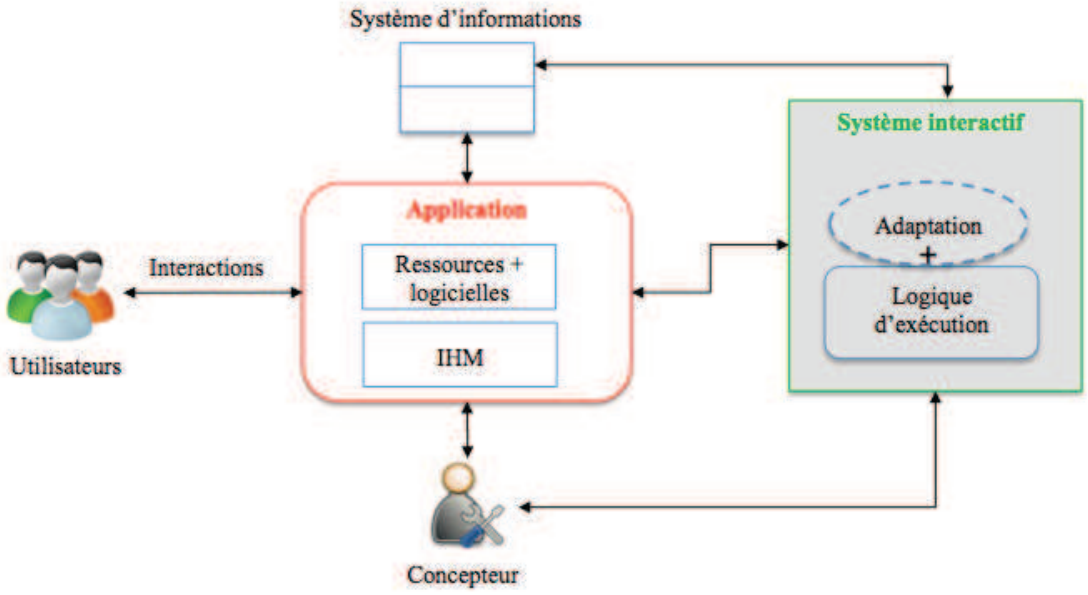

Figure 1. Architecture d'un système interactif adaptatif

Dans notre thèse, nous nous concentrons sur l'adaptation au niveau du noyau fonctionnel. En d'autres termes, nous travaillons sur l'adaptation du processus d'exécution d'une application interactive. Nous introduisons donc *l'exécution adaptative d'une application interactive*.

1.1.2 EXECUTION ADAPTATIVE D'UNE APPLICATION INTERACTIVE

Selon [5], [1], « *l'exécution adaptative est la gestion de l'évolution de l'exécution de l'application par le système en fonction du profil, besoins et comportement de l'utilisateur, en respectant les consignes du concepteur et en tenant compte de l'état des ressources, dans le but de déclencher la meilleure action possible à l'utilisateur* ». Le choix de cette action permet de réduire la différence entre l'état estimé par l'utilisateur et celui espéré par le concepteur afin d'éviter les blocages du déroulement et de maintenir l'intérêt et la motivation de l'utilisateur tout au long de l'exécution.

Les applications interactives sont conçues en définissant leur progression ou leur exécution. Lors de l'exécution, le système interactif peut ajuster ses caractéristiques selon le comportement ou le contexte actuel de l'utilisateur afin de maintenir son attention, son intérêt et d'assurer la continuation du déroulement de l'application. Une question que nous nous posons est : Comment le système peut-il faire l'adaptation lors de l'exécution du système ? L'adaptation se fait au niveau du système lors de l'exécution d'une application, elle intervient sur la logique d'exécution définie par le concepteur. En effet, le déroulement normal sans adaptation se base sur cette logique. Cette dernière donne les règles nécessaires au fonctionnement de l'application. Pourtant, cette logique prédéfinie ne répond pas tout le temps aux besoins de l'utilisateur. La Figure 2 décrit un processus intégré pour mettre en place l'adaptation dans une application interactive.

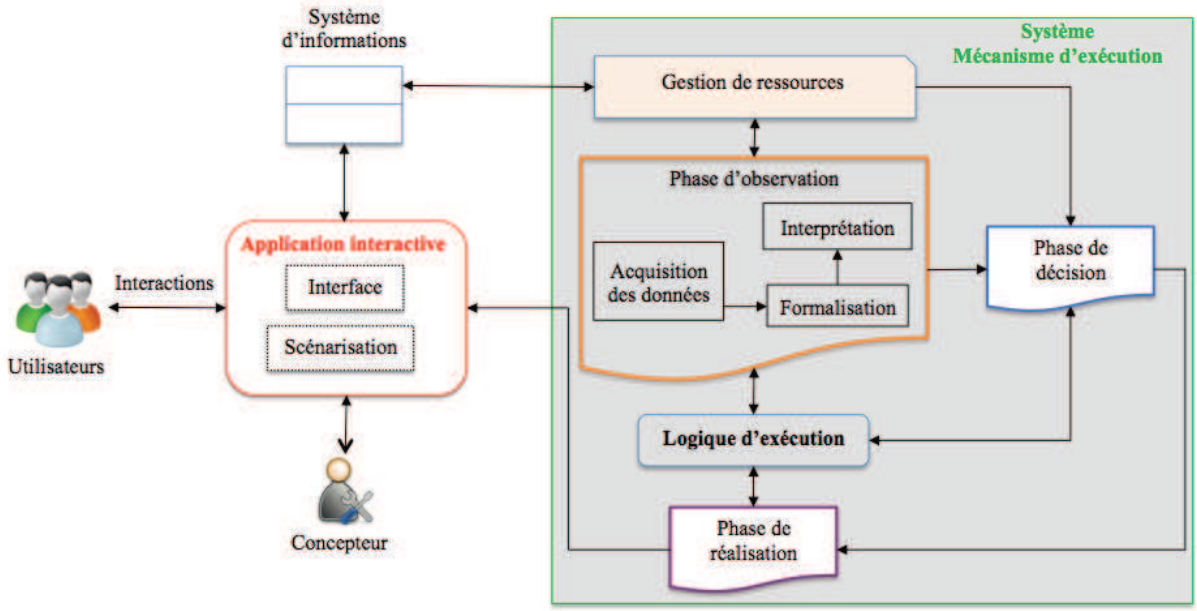

Figure 2. Processus fonctionnel général d'un système gérant une application interactive à exécution adaptative

Nous présentons ci-dessous le déroulement d'une application interactive à exécution adaptative en l'orientant pour répondre à la question posée ci-dessus. Le déroulement est assimilé à un cycle composé des quatre phases suivantes :

- Actions de l'utilisateur : c'est la seule phase qui est réalisée par un acteur humain appelé utilisateur. Elle représente les interactions de l'utilisateur avec le système ;
- Phase d'observation : elle vise à répondre à la question « qu'est-ce qu'on doit observer pour adapter ? ». Elle se divise en trois tâches :

- l'acquisition des données qui collecte des informations de l'utilisateur par un système d'observation ;
- la formalisation des données collectées sous des formes appropriées, et,
- l'interprétation des données formalisées.

- Phase de décision : elle prend en charge les méthodes de sélection des actions à exécuter. Elle s'occupe de la scénarisation les actions de l'utilisateur en donnant la décision pour la suite de la trame scénaristique de l'application ;
- Phase de réalisation : elle concerne la réalisation des actions sélectionnées ou définies dans la phase de décision. Elle permet de réaliser des actions et de mettre en place les objets nécessaires sur l'interface de l'utilisateur.

Ce cycle doit reboucler tout au long du déroulement de l'application comme la montre la Figure 2. Tout au long de l'exécution, l'utilisateur interagit avec l'application. Les trois autres phases sont situées dans le composant « Système – Mécanisme d'exécution ». Le processus décrit dans cette figure est à la base de plusieurs architectures de conception d'un système interactif adaptatif.

Nous avons présenté le processus de l'adaptation. C'est le mot-clé de notre travail et la mission principale que nous allons traiter tout au long de cette thèse. La section suivante va présenter en détail notre objectif général.

1.2 OBJECTIFS GENERAUX

D'après le cycle indiqué dans la section 1.1.2, nous trouvons que le cœur de l'adaptation est concentré en deux phases : celle de l'observation et celle de la décision. Afin de mettre en place l'adaptation dans une exécution d'une application interactive, il nous faut répondre à deux questions correspondant aux deux phases concernées :

- Qu'observons-nous lors de l'exécution de l'application ?
- Comment prendre la meilleure décision ?

L'objectif de cette thèse visent donc à répondre à ces deux questions, qui peuvent se résumer ainsi : **Comment mettre en œuvre l'adaptation lors d'une exécution d'une application interactive ?**

La section suivante va nous décrire en détail le cadre spécifique de cette thèse qui est la scénarisation de l'exécution d'une application interactive par un nouveau formalisme, introduit dans des travaux antérieurs du laboratoire [1] qui s'appelle *situation*. En tenant compte de cette hypothèse, nous orientons notre travail d'adaptation en proposant les approches correspondantes à deux problématiques identifiées pour atteindre l'objectif défini de la thèse.

1.3 SCENARISATION DES APPLICATIONS INTERACTIVES EN SITUATIONS

L'exécution d'une application interactive peut être vue comme une suite d'actions et d'événements réalisés par les utilisateurs et le système.

L'exécution adaptative dans une application interactive permet d'adapter les comportements en fonction du contexte et des objectifs à atteindre. En effet, au cours de l'exécution de

l'application, l'utilisateur peut avoir de nombreuses façons pour atteindre les objectifs définis. Elles ne sont pas toujours optimales, parce qu'une façon peut être optimale dans un contexte avec un utilisateur donné, mais ne le sera pas dans un autre contexte avec un autre utilisateur. Il est donc nécessaire de gérer l'adaptation pendant l'exécution d'une application interactive.

La scénarisation d'une application interactive a généralement déjà des structures scénarisées. Pourtant, ces structures ne sont pas suffisantes pour structurer les scénarios complexes et libres au sens que les interactions ou les actions sont imprévisibles. En plus, elles ne sont pas conçues pour supporter la gestion de cohérence ainsi que l'adaptation à l'intérieur de l'application. Dans notre contexte spécifique, nous cherchons une structure qui permet de piloter facilement le scénario de l'application et de faciliter l'adaptation tout au long de l'exécution. Cette structure devra tenir compte des acteurs, du système et de la gestion de cohérence de l'application. Pour cette raison, nous utilisons un concept qui considère la *situation* comme un gabarit d'interaction qui fait évoluer l'exécution de l'application.

Le concept de *situation* est présenté et détaillé par Pham dans [1], [6]. La *situation* représente l'élément structurant du scénario. L'auteur a défini deux types de situations : les situations élémentaires et les situations composées. Ainsi, une situation peut être découpée en plusieurs situations de plus bas niveau, ou englobée par une situation composée d'un niveau plus haut. Une situation est élémentaire si et seulement si elle est au niveau le plus bas selon le degré de granularité et elle ne peut plus être découpée en situations plus fines. Un scénario est une situation composée le plus globale. Dans le contexte de cette thèse, nous n'abordons que la situation élémentaire. Nous allons appeler situation au lieu de situation élémentaire Tout au long de la thèse, nous utiliserons le terme *situation* pour désigner la situation élémentaire telle qu'elle a été définie dans les travaux de Pham.

La Figure 3 est une illustration des éléments d'une situation. Une situation est un composant où les acteurs interagissent en utilisant des ressources locales dans un contexte précis pour atteindre un ou plusieurs objectifs. À côté des données internes, il existe aussi des données externes venant de différentes sources telles que les variables globales du système d'information, les ressources physiques,… Il faut gérer toutes ces informations pour bien utiliser dans la représentation d'une situation.

Nous n'abordons que la situation élémentaire (les travaux de Pham introduisent les structures de situations composées). Ce point de vue n'est finalement pas restrictif dans la mesure où les situations composées sont constituées de regroupements de situations élémentaires. Nous considérons que l'exécution d'une application interactive produit une séquence de situations élémentaires réalisées liée à la dynamique de l'exécution de la situation. Nous allons rappeler trois définitions liées aux trois notions utilisées dans la thèse. Concernant les autres composants d'une situation, elles sont décrites dans l'Annexe E.

Un *contexte global* : il représente une vision globale en se basant sur l'état du monde externe (environnement, ressources, *etc*), l'état des acteurs et leurs profils associés.

Une *pré-condition* est un ensemble de conditions devant être vérifiées afin de commencer une nouvelle situation. Ce sont des valeurs pré-requises quand le système veut suivre la logique d'exécution du concepteur. Une pré-condition est constituée d'un ensemble de conditions basiques. Chacune porte une valeur qui permet de quantifier ou mesurer les états des participants contribuant dans cette situation ainsi que l'état du système.

Une *post-condition* est le résultat attendu après avoir exécuté la situation. Elle représente les conditions de sortie d'une situation. Au cours de l'exécution de la situation, les états des participants ou du système changent effectivement. Lorsque les conditions sont vérifiées la sortie de la situation permet d'enregistrer tous les changements survenus au niveau des valeurs des états impliqués. La post-condition est organisée aussi en blocs indépendants de conditions basiques.

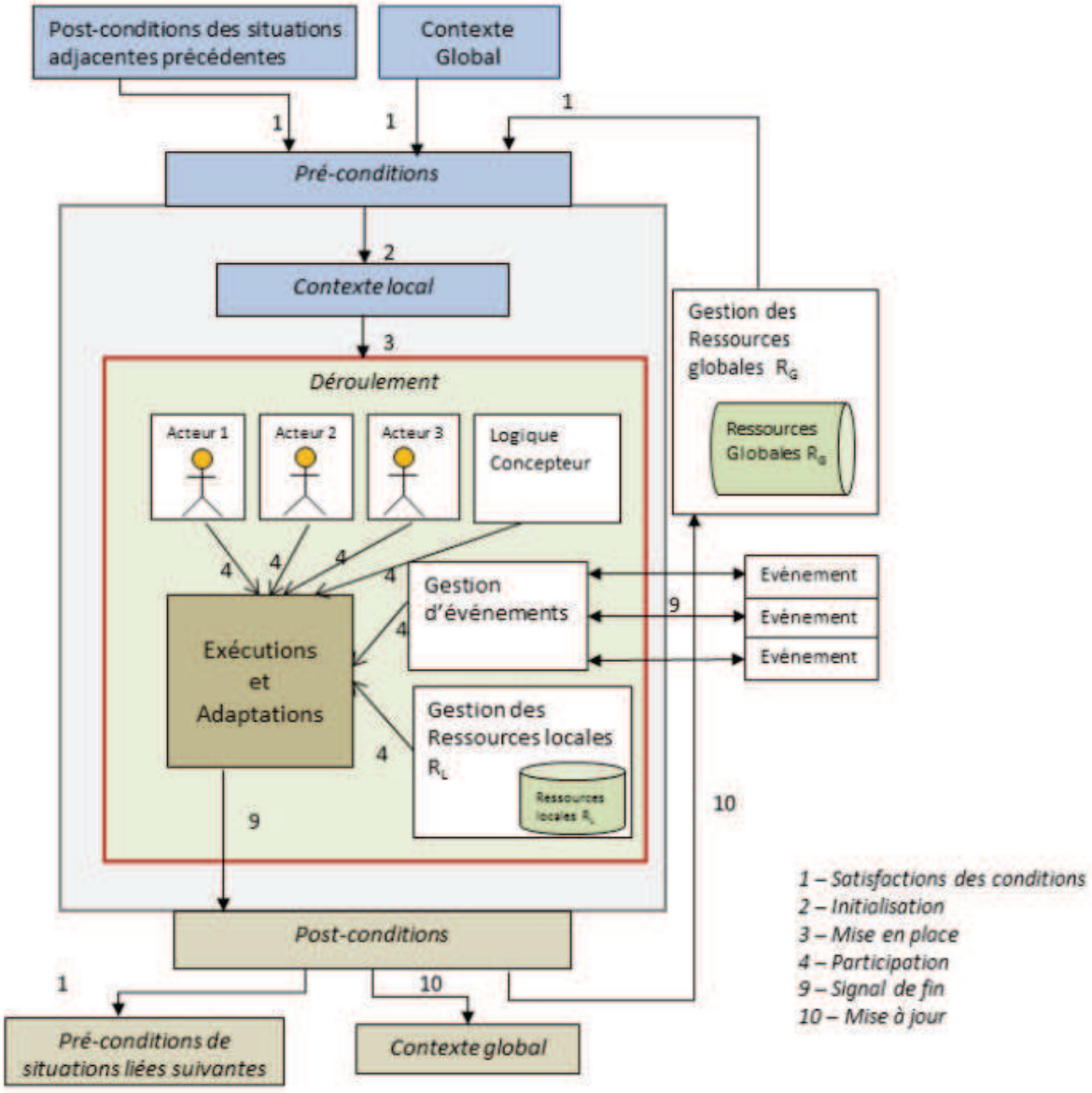

Figure 3. Elément d'une situation [1]

Nous proposons donc de structurer et de formuler le déroulement d'une application interactive adaptative en termes de *situations* qui seront les briques de base permettant de décrire la trame scénaristique de l'application. Ainsi, l'utilisateur d'une application interactive exécute et participe à des situations successives jusqu'à atteindre un objectif prédéfini par le concepteur. L'exécution d'une situation est subordonnée à la satisfaction de pré-conditions. À l'issue de son exécution, des post-conditions permettent d'en établies.

1.4 PROBLEMATIQUES

Dans ce cadre, nous cherchons une approche visant à permettre une optimisation de la réalisation d'un scénario représenté par un enchaînement de situations.

Ce scénario pouvant être soumis à des perturbations, nous proposons dans cette thèse une méthode permettant d'optimiser l'enchaînement de situations composant une application interactive. Ceci repose sur une logique menant à la réalisation des objectifs fixés par le concepteur de ce scénario en s'adaptant au comportement de l'utilisateur, et en tenant compte des contraintes liées à la disponibilité des ressources nécessaires à l'exécution.

Il s'agit de concevoir des méthodes et des outils permettant d'effectuer et/ou d'accompagner le choix de la situation à exécuter parmi un ensemble de situations potentiellement exécutables. Pour cela, nous avons fondé une partie de nos travaux sur le domaine des systèmes d'aide à la décision. Ces systèmes vont nous permettre d'optimiser les actions : identifier la situation qui permet de continuer l'exécution de l'application et respecter un ou plusieurs objectifs de l'application. La situation ainsi proposée par le système de décision est obtenue en évaluant ses pré-conditions, mais aussi parce qu'elle satisfait les objectifs globaux liés au choix du concepteur, au comportement observé et interprété de l'utilisateur mais aussi à l'état des ressources du système.

Cette décision repose cependant sur une multitude de critères. Dans ce cas, l'aide à la décision doit prendre en compte tous ces critères pour choisir une solution optimale dans l'ensemble des solutions possibles à l'instant *t*. Ceci nous a amené à envisager des *systèmes d'aide à la décision multicritère*.

Par ailleurs, l'observation et l'évaluation de l'exécution d'une application interactive sont souvent basées sur l'analyse de grands volumes de données porteuses d'informations contextualisées collectées pendant l'exécution. Au cours de l'exécution de l'application, pour renforcer le système d'information avec des éléments facilitant l'analyse et l'adaptation de la logique d'exécution, nous introduisons des données complémentaires appelées traces. Un système permettant de gérer les traces, appelé *système à base de traces* définit la façon d'observer, de collecter et d'analyser des traces laissées par les utilisateurs précédents. Un système à base de trace permet au système d'aide à la décision d'utiliser les traces pertinentes pour pouvoir décider en intégrant les informations des exécutions précédentes de l'application interactive. L'utilisation des traces sert à affiner le choix du système d'aide à la décision multicritère. La solution trouvée permet d'adapter la logique définie par le concepteur en conformité avec le contexte spécifique actuel.

Nous avons déjà identifié les deux problématiques principales de nos travaux : les *systèmes d'aide à la décision multicritère* et les *systèmes à base de traces* correspondant aux deux questions posées dans 1.2. La section 1.5 suivante va présenter brièvement les approches envisagées pour répondre à aux problématiques déterminées.

1.5 APPROCHES

Nous détaillons l'architecture nécessaire à la mise en place de l'adaptation pendant l'exécution d'une application interactive à base de situations. La Figure 4 décrit globalement notre approche.

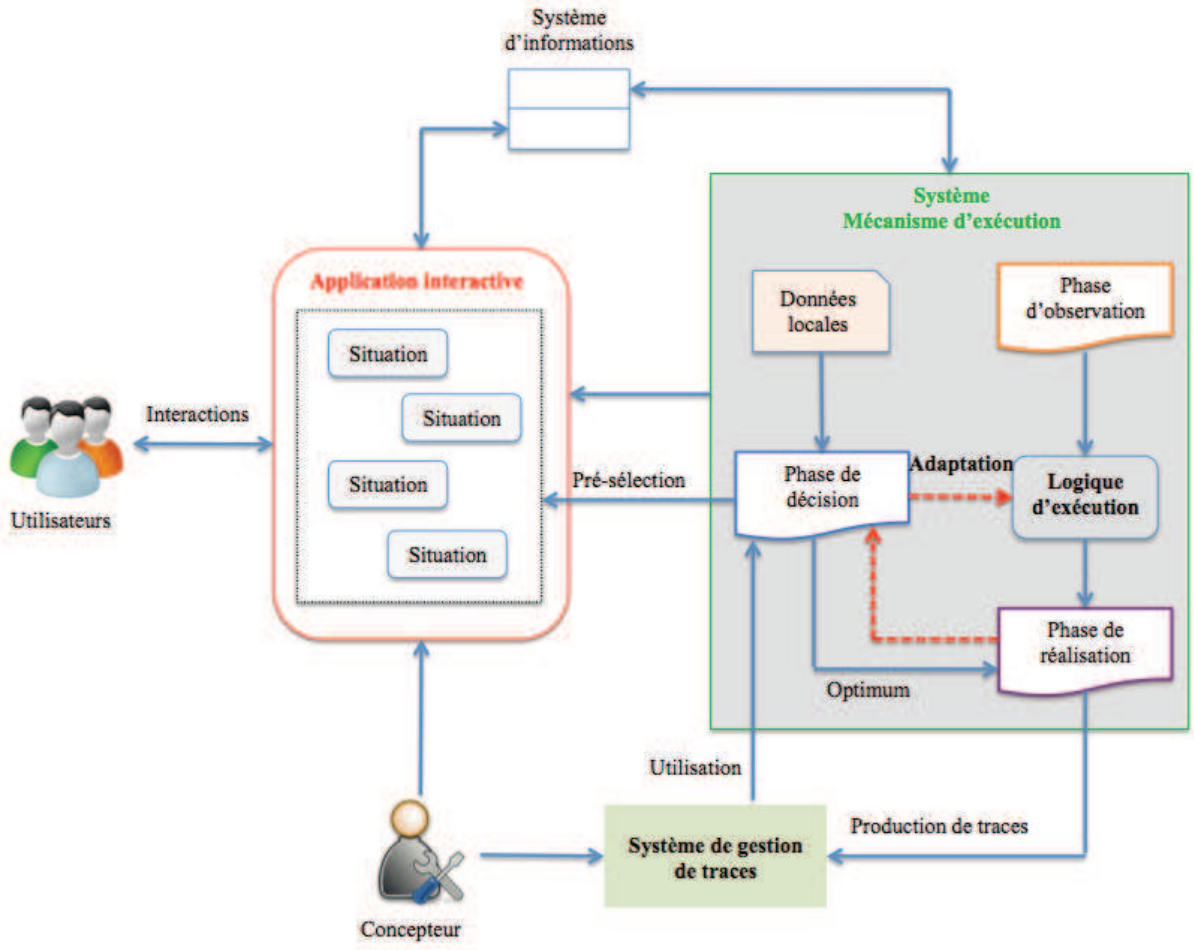

Figure 4. Architecture de notre approche

Lorsqu'un ou plusieurs utilisateurs commencent à exécuter une application interactive utilisant un scénario à base de situations, ils suivent la logique définie dans le scénario en respectant une logique d'exécution qui a été définie par le concepteur. À un moment donné, le scénario peut rencontrer un blocage lors de son déroulement dû au contexte local ou à un souci provenant du système (par exemple l'incohérence de données ou l'apparition de quiproquos [1]) ou il ne sait pas comment avancer le déroulement de l'application en choisissant la meilleure situation. Dans ce cas, une phase de décision intégrée dans le système est appelée pour pouvoir traiter et donner une solution optimale en tenant compte des objectifs. Puis, la solution obtenue est envoyée à la phase de réalisation pour assurer la continuité de l'exécution. L'utilisateur va continuer à suivre la logique d'exécution dans le scénario. Le système d'aide à la décision donnera un support à l'utilisateur pour prendre la meilleure solution pendant le déroulement de l'application.

Par ailleurs, lors de l'exécution des situations, il y a de nombreuses informations générées, par les interactions et le système lui-même, appelées traces. Elles décrivent exactement ce qui s'est passé dans les exécutions précédentes. Elles contiennent la trace des actions passées des utilisateurs. Nous proposons de réutiliser ces traces afin de guider les nouveaux utilisateurs dans les prochaines exécutions. C'est pour cela que nous intégrons à notre architecture un système de gestion de traces permettant d'observer et de gérer les traces.

Les deux solutions proposées nous permettent de résoudre le problème d'adaptation lors d'une exécution d'une application interactive. En d'autres termes, notre mécanisme d'adaptation

devra fonctionner en *temps réel* et il pourra *utiliser les traces* pour renforcer le calcul. Par ailleurs, selon la Figure 4, nous constatons qu'il dispose de trois acteurs (physique ou non) qui contribuent à une exécution adaptative : le concepteur, l'utilisateur et le système. Chacun d'entre eux peut avoir différentes stratégies pour adapter la logique d'exécution de l'application. Nous définissons donc un autre besoin de nos travaux qui est *la prise en compte des différentes stratégies* dans le mécanisme d'adaptation attendu. Ainsi nos trois besoins pour un mécanisme d'adaptation sont les suivants :

- fonctionner en temps réel ;
- réutiliser les traces ;
- prendre en compte les différentes stratégies.

Nous avons également abordé le système d'aide à la décision dans l'approche principale de la thèse comme un mécanisme d'adaptation approprié pour résoudre notre problème. L'analyse détaillée de ce choix sera menée dans le Chapitre 2 en tenant compte des trois besoins suivants : pondération des poids des critères, détermination des alternatives et définition de l'algorithme de décision. En effet, un système d'aide à la décision comporte lui-même des caractéristiques particulières. Tous les systèmes de décision ont considéré un ensemble de critères à satisfaire lors de la prise de décision. Concrètement, un système de décision a besoin de la priorité des critères ou des poids associés à chaque critère mais il a aussi besoin de l'ensemble des alternatives (les solutions possibles) pour la prise de décision et de l'algorithme permettant de calculer le choix. En outre, en reprenant le dernier besoin mentionné pour le mécanisme d'adaptation attendu ci-dessus (prise en compte des différentes stratégies), nous allons considérer un quatrième besoin dans notre système de décision : la prise en compte des différents choix. Ainsi, pour un système de décision, nous avons besoins :

- des poids des critères ;
- des alternatives à prendre de décision ;
- de l'algorithme de décision ;
- de l'agrégation des différents choix.

La section 2.2.2 dans le Chapitre 2 tiendra compte de ces quatre besoins d'un système de décision pour analyser les méthodes existantes.

1.6 TRAVAUX DE LA THESE

L'objectif de cette thèse vise à concevoir une architecture et une méthodologie intégrée dans un système interactif permettant d'adapter le déroulement de l'exécution en se basant sur les traces générées. Pour cela, nous proposons les contributions suivantes :

- Concevoir un mécanisme de gestion de traces dans les applications interactives à base de situations ;
- Définir un processus de décision multicritère à base de traces ;
- Proposer des algorithmes de décision à base de traces adaptés à la mise en œuvre ce processus.

1.6.1 QUESTIONS POSEES DANS LA THESE

Pour atteindre ces contributions, nous allons caractériser la thèse en plusieurs questions auxquelles nous allons répondre :

- Sur la décision :
  - De quoi décidons-nous ?
  - Quelles techniques utiliser pour prendre la décision ?
  - De quels types de données avons-nous besoin pour décider ?
  - Est-ce que le choix final tient compte des différentes logiques de l'application : celle du concepteur, de l'utilisateur et du système ?
- Sur les traces :
  - Quel est le modèle de trace ?
  - Quels types de données récupérons-nous lors de l'exécution ?
  - Comment construire un système à base de traces pour les applications interactives à base de situations ?
- Sur l'évaluation des résultats :
  - Quelle étude de cas choisirons-nous pour illustrer nos contributions ?
  - Comment évaluer nos contributions ?

1.6.2 PLAN DU MANUSCRIT

Le manuscrit de la thèse se compose de 4 parties comme la montre la Figure 5.

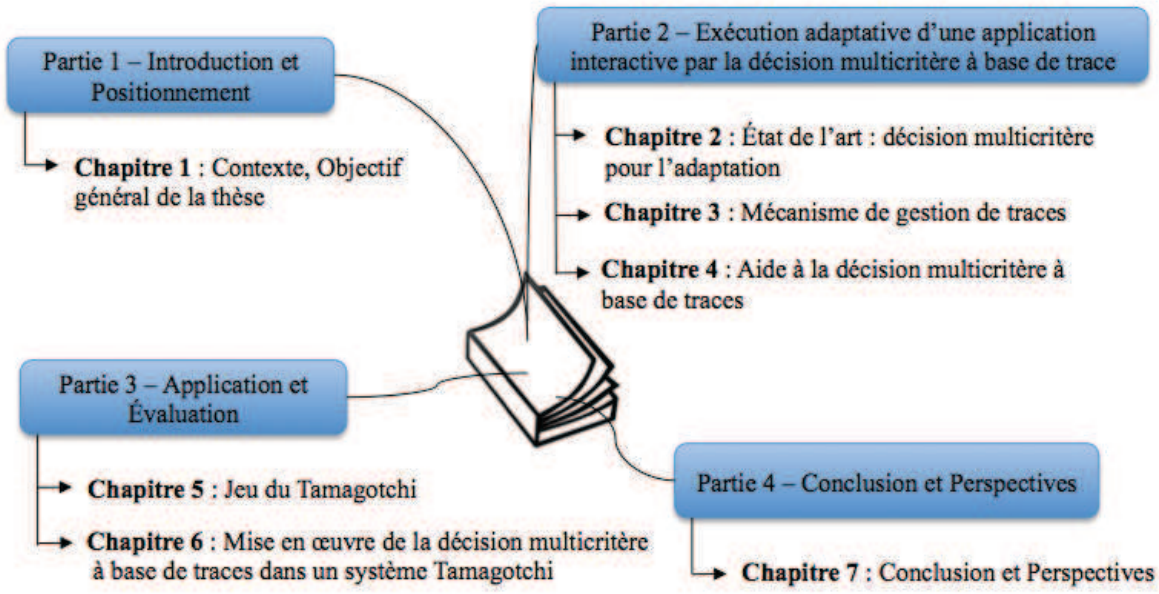

Figure 5. Plan du manuscrit

Après le premier chapitre où nous avons présenté le contexte de la thèse, les problématiques de recherche, les approches de solutions, nous consacrons la deuxième partie du manuscrit à l'exécution adaptative d'une application interactive par la décision multicritère à base de traces.

La partie 2 est constituée de trois chapitres. Le chapitre 2 va présenter en détail l'état de l'art des mécanismes pour mettre en place l'adaptation dans une application interactive à base de situations. La théorie de la décision multicritère sera choisie en raison des hypothèses et de nos besoins d'adaptation. Les théories existantes seront analysées dans ce chapitre pour déduire quelles sont les lacunes des systèmes d'aide à la décision multicritère existant dans la littérature. Malgré l'efficacité des systèmes de décision, nous trouvons que nous pouvons affiner le choix en intégrant dans le calcul des habitudes et comportement passés des anciens utilisateurs. Par conséquent, le chapitre 3 vise à expliquer comment obtenir ces informations par l'introduction des mécanismes de gestion de traces. En combinant un système d'aide à la décision à un système de gestion des traces, nous proposons un nouveau processus d'aide à la décision multicritère à base de traces dans le chapitre 4 de la thèse.

La troisième partie du manuscrit est consacrée à l'expérimentation et la validation de nos propositions. Nous présenterons dans le chapitre 5, une étude de cas autour du principe du jeu Tamagotchi[1]. C'est un jeu simple qui a pour but d'entretenir et de maintenir en vie un animal virtuel appelé Tamagotchi. Les actions de l'utilisateur seront contextualisées au moyen des situations. L'utilisateur doit exécuter les situations au fur et à mesure pour jouer avec le Tamagotchi. Dans le chapitre 6, nous présentons les expérimentations de l'application de nos contributions sur le système Tamagotchi. Les modèles de traces et le processus d'aide à la décision multicritère à base de traces seront expérimentés et validés sur ce cas d'étude.

Enfin, dans la quatrième partie du manuscrit, nous conclurons nos travaux de recherches et décrirons les perspectives scientifiques intéressantes à explorer dans la continuité de la thèse.

[1] https://fr.wikipedia.org/wiki/Tamagotchi

Partie 2 : Exécution adaptative d'une application interactive par la décision multicritère à base de traces

CHAPITRE 2 ETAT DE L'ART : DECISION MULTICRITERE POUR L'ADAPTATION

SOMMAIRE

Les applications décrites au moyen des situations doivent être conçues pour bien gérer la cohérence des données lors de l'exécution de l'application en assurant que leur exécution s'adapte à un contexte précis. Autrement dit, le système interactif peut s'ajuster selon les réactions ou les comportements de l'utilisateur dans le but de maintenir son exécution jusqu'à la fin.

La première partie de ce chapitre répondra au problème de l'adaptation dans un système interactif adaptatif en catégorisant les mécanismes d'adaptation. Nous présentons un processus de décision multicritère qui sera appliqué pour résoudre notre problème d'adaptation lors d'une exécution d'une application interactive. La section suivante détaille les différentes approches de décision. Pour finir ce chapitre, nous identifions les verrous en nous basant sur l'état de l'art et nos besoins. Nous nous intéressons finalement à ces verrous en donnant les propositions principales pour les lever.

2.1 INTRODUCTION

Une application interactive, par définition, est une application qui interagit avec son environnement. Ce type d'application est capable de percevoir l'environnement extérieur à travers des périphériques classiques tels que : le souris, le clavier, *etc.* et aussi à travers des nouveaux périphériques, par exemple le caméra, le capteur d'audio, *etc*. Dans ce type d'applications, nous nous concentrons sur celles pour lesquelles l'exécution de l'application doit suivre une trame préétablie par le concepteur [7]. L'application est dite scénarisée parce qu'elle contient un scénario qui gère la progression de son exécution. Pour cela, le concepteur définit le déroulement de l'application par l'enchainement des actions en une trame scénarisée, appelé scénario.

Nous nous intéressons à une classe d'applications particulière : les applications interactives à exécution adaptive qui exploitent les mécanismes d'adaptation durant leurs exécution de l'application. Comme nous avons dit dans le Chapitre 1, l'adaptation dans notre contexte consiste à adapter l'exécution d'une application à base de situations par une analyse des choix (des situations) à lancer au niveau de l'application afin que son déroulement suive une structure prédéfinie par le concepteur et en tenant compte des objectifs et des préférences de l'utilisateur. Ce type de structure peut être vue comme la planification des situations successives ; l'organisation de l'enchainement des situations. Dans ce cadre, l'adaptation est de chercher à comprendre et d'orienter l'interaction de l'utilisateur pour permettre une évolution adaptative du déroulement de l'application. Les résultats des travaux des thèses de Delmas [8] et Dang [9] portent sur la logique du concepteur. Ils visent à construire un scénario adapté en fonction du contexte, du comportement de l'utilisateur et de l'état des ressources. L'utilisateur doit ensuite suivre le déroulement en se basant sur le scénario obtenu (voir l'Annexe F).

Dans le contexte de la thèse, nous abordons le problème des situations émergentes. Nous considérons un univers riche dans lequel l'utilisateur peut évoluer de manière libre et autonome. Dans ce cadre, nous cherchons à adapter l'exécution d'une application interactive à base de situations en choisissant une situation à exécuter dans le prochain déroulement de l'application indépendamment d'un éventuel graphe de situations préétabli par le concepteur ; graphe représentant le scénario nominal). C'est pour cela que nous intervenons sur l'aspect de choix (notamment soit les techniques de recommandation, soit les techniques de décision dans la section 2.2) pour répondre à notre problème d'adaptation.

2.2 MECANISME D'ADAPTATION

Dans cette classe d'applications, nous allons chercher à savoir comment adapter une application en temps réel et quelles sont les techniques d'adaptation existantes ? Le but de l'adaptation lors d'une exécution est de trouver la meilleure action ou la meilleure ressource à partir d'un ensemble d'actions ou de ressources disponibles. Afin de résoudre ce type de problème, deux approches émergent à savoir la recommandation et la décision. Ces approches permettent de trouver ou de classer les actions ou les ressources en tenant compte du contexte ou de l'état actuel du système ou de l'utilisateur. Nous allons donc approfondir les deux types d'adaptation suivants :

- Adaptation par la recommandation ;
- Adaptation par la prise de décision.

2.2.1 ADAPTATION PAR LA RECOMMANDATION

La **recommandation** est une forme spécifique de filtrage de l'information visant à présenter des contenus (ou ressources) susceptibles d'intéresser l'utilisateur. Généralement, elle sert à identifier et à proposer un item ou un ensemble d'items dont l'utilisateur a besoin dans un contexte défini par un profil utilisateur [10]. Dans le cadre d'applications interactives à base de situations, l'adaptation par la recommandation a pour objectif de fournir à un utilisateur une liste ordonnée de situations possibles en tenant compte de ses préférences et de ses objectifs. Ce travail s'appuie sur les données courantes de l'utilisateur ainsi que sur celles déjà existantes (provenant également des autres utilisateurs) pour recommander une situation appropriée à l'utilisateur.

Il existe un grand nombre d'approches de recommandation. Plusieurs facteurs permettent de catégoriser les méthodes de recommandation [11]:

- La connaissance de l'utilisateur (son profil) ;
- Le positionnement d'un utilisateur par rapport aux autres (réseaux d'utilisateurs) ;
- La connaissance des items à recommander ;
- La connaissance des différentes classes d'items à recommander.

De ces facteurs sont produits divers types de recommandations dont les plus utilisés dans la communauté sont la recommandation basée sur le contenu, la recommandation sociale et les approches hybrides.

2.2.1.1 Recommandation basée sur le contenu

L'approche basée sur le contenu s'appuie sur des évaluations effectuées par un utilisateur sur un ensemble d'items donné. L'objectif est alors de comprendre les motivations l'ayant conduit à juger comme pertinent ou non un item donné [12] [11]. En d'autres termes, cette méthode analyse un ensemble d'items identifiés précédemment par l'utilisateur pour construire un modèle des intérêts de l'utilisateur en se basant sur les caractéristiques des items. Le processus de recommandation consiste à apparier les attributs d'un item avec les attributs des items à recommander. Le processus de recommandation est divisé en deux étapes dont chacune représente un composant :

- Analyse de contenu : cette étape a pour but de prétraiter et extraire des informations pertinentes, de les structurer et de les représenter dans une forme cible appropriée (un vecteur de mots clés, *etc*.) ;
- Composant du filtrage : cette étape analyse les items choisis par l'utilisateur et lui recommande des items pertinents en réalisant l'appariement entre les attributs de l'item et chacun des items à recommander. Cette pertinence est faite par des méthodes de calcul de similarité. Cette mesure prend une valeur entre 0 et 1 qui est aussi le résultat de recommandation des items donnés.

La recherche sur les systèmes de recommandation basés sur le contenu applique de nombreuses techniques de différents domaines. Il existe deux techniques principales : la recherche d'information et la fouille de données.

Dans la recherche d'information, le système doit apparier la description d'un item (les attributs) et les items à choisir pour recommander quels sont les items les plus pertinents. Les items recommandés à l'utilisateur sont représentés par un ensemble de caractéristiques. Pour faciliter la modélisation de ces caractéristiques, le modèle vectoriel est un modèle simple. Dans ce modèle, chaque item et utilisateur sont représentés par un vecteur dont chaque composante correspond à une valeur dans un contexte identifié.

Par ailleurs, il existe aussi des recommandations basées sur le contenu utilisant des approches de fouille de données. Par exemple, des classifieurs bayésiens, le clustering, les arbres de décision et les réseaux de neurones. Ces techniques sont différentes de celles qui concernent la recherche d'information parce qu'elles calculent la prédiction en se basant sur un modèle d'apprentissage qui a été construit par l'extraction des données antérieures.

2.2.1.2 Recommandation sociale

Une approche classique de la recommandation sociale est l'approche par filtrage collaboratif qui s'appuie sur les évaluations de l'utilisateur et celles d'autres utilisateurs dans son réseau. L'idée principale est que l'évaluation d'un utilisateur u pour un item i est susceptible d'être similaire à celle d'un autre utilisateur u^* si u et u^* ont évalué les items de la même façon comme dans la Figure 6.

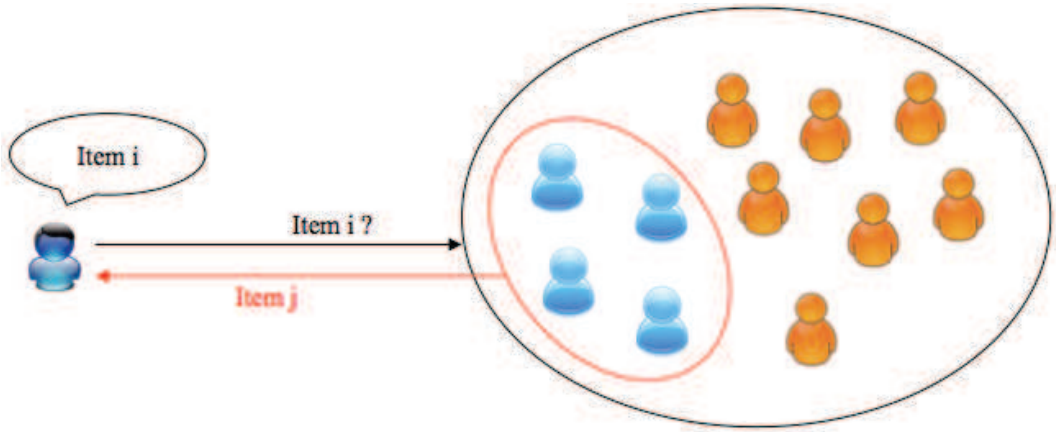

Figure 6. Exploitation les préférences des autres pour prédire le comportement d'un individu donné

L'avantage de cette approche est qu'elle peut surmonter le problème d'indisponibilité de contenu des items ; il s'agit des items dont les contenus ne sont pas disponibles ou difficiles à caractériser. Un autre avantage concerne l'aspect social qui décrit la qualité des items observés par les utilisateurs antérieurs alors que la recommandation basée sur le contenu repose uniquement sur le contenu des items qui peut être un mauvais indicateur. Les méthodes de filtrage collaboratif peuvent être catégorisées en deux grandes types : à base de mémoire et à base de modèles [13].

Le système de recommandation à base de mémoire s'appuie sur les évaluations de l'utilisateur pour apprécier un item. Les évaluations des utilisateurs pour les items stockés dans le système sont analysées directement pour prédire les évaluations des items qui ne sont pas encore évalués. Il existe deux techniques :

- Le filtrage collaboratif basé utilisateur calcule l'évaluation d'un utilisateur u pour l'item i en analysant les évaluations attribuées à i par les différents utilisateurs étant les plus similaires à u. Le calcul de similarité est souvent appliqué tels que la similarité cosinus [14] ou la corrélation Pearson [15] ;
- Au lieu de mesurer la corrélation entre des utilisateurs, le filtrage collaboratif basé items utilise les évaluations obtenues pour mesurer la similarité entre les items.

Contrairement à la recommandation à base de mémoire, celle à base de modèle utilise les évaluations pour construire un modèle de prédiction. Ce modèle sera utilisé pour prédire les évaluations des utilisateurs pour de nouveaux items. Ce type de recommandation offre la possibilité de construire le modèle de prédiction hors ligne puis de l'utiliser en ligne pour effectuer la recommandation.

2.2.1.3 Approche hybride

D'après [16], il s'agit de combiner les deux méthodes de recommandation basée sur le contenu et sur le filtrage collaboratif pour mettre en place des approches hybrides permettant d'effectuer la recommandation. Il y a quatre manières de combiner ces approches :

- Mettre en œuvre ces deux méthodes séparément et combiner les valeurs de prédiction ;
- Extraire des caractéristiques de l'approche basée sur le contenu et les intégrer dans l'approche de filtrage collaboratif ;
- Extraire des caractéristiques de l'approche de filtrage collaboratif et l'intégrer dans l'approche basée sur le contenu ;
- Construire un modèle unifié qui hérite des caractéristiques de l'approche basée sur le contenu et celle sur le filtrage collaboratif.

2.2.2 ADAPTATION PAR LA PRISE DE DECISION

Cette section est destinée à l'étude générale des mécanismes de prise de décision qui sont utilisés à la mise en place de l'adaptation pour les systèmes interactifs adaptatifs. Nous allons présenter l'objectif de la prise de décision et ainsi que la catégorisation de la prise de décision avant de décrire des différentes méthodes d'adaptations existantes où ils s'agissent de la prise de décision.

La **prise de décision** vise à aider les utilisateurs dans des procédures de traitement de l'information ou de choix à partir d'un ensemble de choix disponibles. D'après [17], « la prise de décision est l'activité de celui qui, prenant appui sur des modèles clairement explicités mais non nécessairement complètement formalisés, aide à obtenir des éléments de réponses aux questions que se pose un intervenant dans le processus de décision, éléments concourant à éclairer la décision et normalement à prescrire, ou simplement à favoriser un comportement de nature à accroitre la cohérence entre l'évolution du processus d'une part, les objectifs et le système de valeurs au service desquels cet intervenant se trouve placé d'autre part ». La décision est également un processus qui utilise les données disponibles à un instant donné afin de permettre une prise de décision sur un problème traité. Pour aller plus loin, la décision permet de proposer un choix optimal relativement à un objectif à atteindre. Lors de la décision, il faut tenir compte des critères associés aux objectifs de l'application. L'optimisation la plus simple consiste à optimiser un seul critère, appelé monocritère, qui n'est souvent pas le reflet de la réalité. La plupart des prises de décision doivent prendre compte différents critères permettant d'évaluer un objectif d'une application. Dans ce cas, nous parlerons d'optimisation multicritère, ou plus généralement, de prise de décision multicritère. Il existe deux contextes de décision qui sont : l'aide à la décision et la décision automatique [18].

- L'aide à la décision : c'est le contexte où le décideur essaie de dialoguer avec un système pour prendre une décision. La décision n'est pas immédiate mais construite itérativement en combinant ses préférences. Le temps nécessaire à l'élaboration du modèle de décision complet dépend strictement du type d'application.
- La décision automatique : dans ce cas, le système analyse les données en se basant sur un modèle donné pour prendre une décision en un temps donné.

Si nous sommes dans n'importe quel contexte qui a besoin de décision, nous pouvons distinguer plusieurs types de décision selon les résultats attendus :

- Choix : donner une ou plusieurs solutions appropriées ;
- Tri : classer des alternatives dans des catégories différentes, par exemple : Catégorie d'acceptation, catégorie de rejet,…
- Classement : déterminer l'ordre des alternatives (liste de priorité) ;
- Score : associer un score à chaque alternative.

2.2.2.1 Définitions

Dans un premier temps, nous rappelons les définitions des principaux termes employés.

Objectif

Un objectif est un but ou un résultat que l'utilisateur va atteindre

Alternative

Une alternative désigne un objet ou un choix sur lequel opérera le processus de décision.

Critère

Un critère est une fonction définie sur l'ensemble des alternatives, qui permet de juger les alternatives.

Poids des critères

Chaque critère est associé à une valeur indiquant son importance par rapport aux autres critères. Cette valeur est appelée le poids du critère.

Décision multicritère

La prise de décision multicritère (Multi-Criteria Decision Making) constitue une branche de la recherche opérationnelle [19] donnant plusieurs méthodes et approches de recherches. Elle vise à trouver une solution parmi un ensemble d'alternatives possibles.

Décideur

Le décideur est généralement une personne ou un assistant virtuel qui est supposé connaître le problème de décision multicritère. Le décideur se base sur sa connaissance pour exprimer des relations de préférences entre les alternatives déterminées.

Nous définissons quelques notations comme suit :

$A = \{a_1, a_2 \dots, a_n\}$ représente l'ensemble des alternatives dont chaque alternative a_i est formalisée par p attributs x : $a_i = (x_1^i, x_2^i, \dots, x_p^i)$.

$\succ$ est la relation de préférence sur A, par exemple $\succ (a_1, a_2)$ indique que a_1 est préféré à a_2. Si la relation est binaire, $\succ$ prendra une valeur soit 0, soit 1. Si la relation est floue, elle prendra une valeur dans [0, 1].

$C = \{C_1, C_2, \dots, C_m\}$ est l'ensemble des critères.

$g_h : X \mapsto \mathbb{R}, h \in C$ est une fonction d'évaluation du critère C_h sur une alternative de A.

Classification des Méthodes de décision multicritère

La problématique de décision multicritère est souvent présente dans la vie pratique. Du simple choix d'un achat d'un produit à la sélection d'une profession, la question demeure la même : commet faire le bon choix en tenant compte de toutes les contraintes existantes et des critères identifiés qui contribuent au processus de décision ? Ainsi, un processus de décision multicritère peut être représenté par la Figure 7.

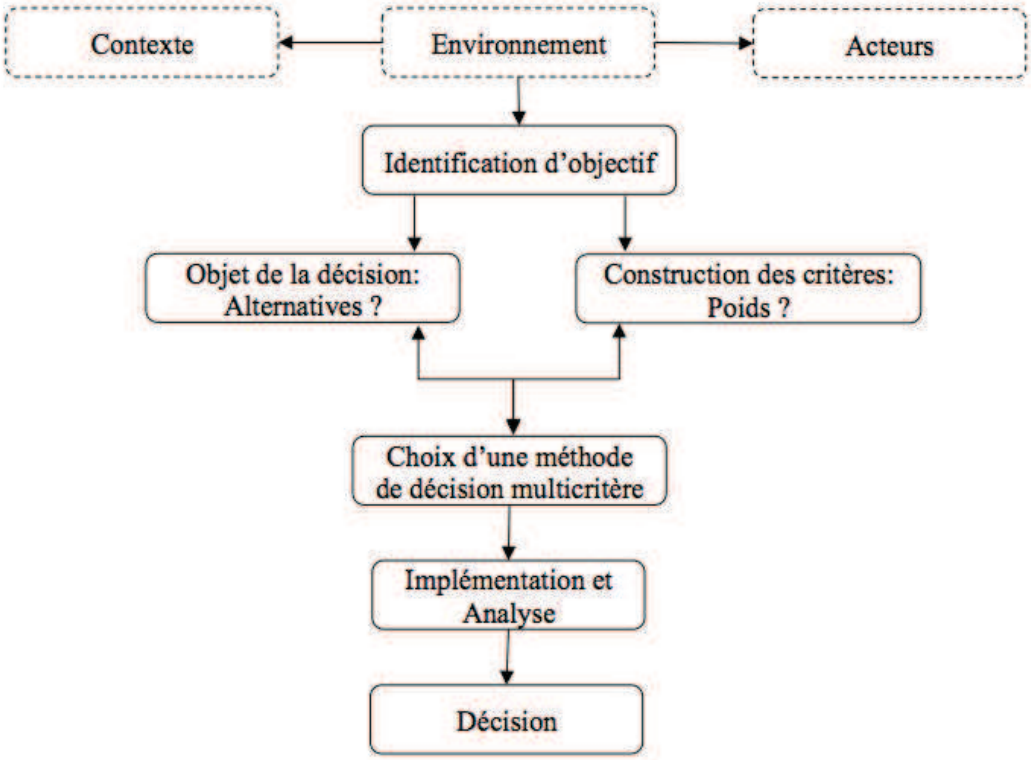

Figure 7. Processus de décision multicritère

Comme la montre dans la Figure 7, nous sommes dans un environnement qui a besoin d'un processus de décision multicritère. Cet environnement consiste en un contexte précis et des acteurs qui contribuent à son exécution. Ce processus est décrit par 6 étapes principales :

- Identifier l'objectif global de la démarche ;
- Dresser la liste des alternatives ;
- Dresser la liste des critères à prendre en considération ;
- Choisir une méthode de décision multicritère ;
- Implémenter et Analyser;
- Décider.

La problématique de décision multicritère est d'autant plus complexe qu'elle a engendré de multiples travaux de recherches, qui ont donné naissance à de multiples méthodes de décision multicritère. Chaque type de décision correspond à différentes méthodologies. Cependant, la plupart des méthodes demandent d'évaluer et de comparer des alternatives selon des critères pour choisir une meilleure alternative qui correspond à un contexte précis. Ce sont deux étapes obligatoires pour résoudre le problème. En nous basant sur ces deux étapes, nous distinguons deux catégories d'approches de décision multicritère qui sont : approche agréger puis comparer et approche comparer puis agréger.

2.2.2.2 Approche « agréger puis comparer »

Le principe de cette approche est d'évaluer d'abord chaque alternative selon *m* critères. Puis, nous appliquons une fonction d'agrégation pour calculer l'évaluation totale de chaque

alternative. Enfin, une fonction de comparaison est utilisée pour déterminer la relation de préférence entre les différentes alternatives. L'approche consiste à calculer :

$$\succ (a_1, a_2) = P\left(\mathcal{E}\big(g_1(a_1), g_2(a_1), \dots, g_m(a_1)\big), \mathcal{E}\big(g_1(a_2), g_2(a_2), \dots, g_m(a_2)\big)\right) \quad (1)$$

Avec :

$\mathcal{E} : \mathbb{R}^m \mapsto \mathbb{R}$ une fonction d'agrégation de *m* critères sur une alternative

$P(a_1, a_2)$ une fonction de comparaison entre deux alternatives a_1 et a_2.

Nous retrouvons cette approche dans les méthodes de pondération et celles d'utilité de multi-attributs. Ce sont les deux premières approches qui ont été proposées pour résoudre le problème de décision multicritère. Elles font la phase d'agrégation des alternatives, puis elles se basent sur le résultat d'agrégation pour comparer les alternatives. La description de ces deux approches se trouve dans les sections 2.2.2.2.1 et 2.2.2.2.2.

2.2.2.2.1 Méthodes de pondération

Il existe différentes méthodes basées sur le principe de pondération.

WSM (Weighted Sum Model)

Elle est probablement la méthode la plus utilisée dans la communauté de décision. En supposant que nous avons *n* alternatives et *m* critères, la meilleure alternative est celle qui satisfait l'expression suivante [20] :

$$A^*_{WSM} = max \sum_{h=1}^{m} e_{ih}.w_h \qquad i = 1, 2, \dots, n \quad (2)$$

Avec :

A^*_{WSM} est le poids de WSM de la meilleure alternative
e est la valeur d'évaluation de l'alternative i pour le critère C_h
w_h est le poids de l'importance du critère C_h

La pertinence de chaque solution est égale à la somme des productions calculées dans la formule ci-dessus. Après avoir estimé le poids total de chaque alternative, le choix se portera sur celle ayant la valeur maximale. Cette approche ne peut s'appliquer que dans le cas monodimensionnel (nous n'avons qu'une seule unité de mesure pour tous les critères).

WPM (Weighted Product Model)

C'est une méthode similaire à WSM. Chaque solution est comparée avec les autres solutions par la formule [21], [22] :

$$R(a_r, a_t) = \prod_{h=1}^{m} \left(\frac{e_{rh}}{e_{th}}\right)^{w_h} \quad (3)$$

Avec :

$R(a_r, a_t)$ est la relation entre deux solutions a_r et a_t

e_{ih} est la valeur d'évaluation de l'alternative i en terme du critère C_h

w_h est le poids de l'importance du critère C_h

Si la valeur $R(a_r, a_t)$ est supérieure à 1, la solution a_r sera plus désirable que a_t. La solution pertinente est celle qui est supérieure ou égale aux autres solutions. La différence de cette approche est la multiplication dans le modèle au lieu d'utilisation l'addition. WPM est considérée comme une modification de WSM et donc elle peut surmonter l'inconvénient de WSM (aspect monodimensionnel). WPM permet d'éliminer l'unité de mesure grâce à sa nouvelle structure de multiplication. Elle peut être appliquée dans les 2 cas : mono et multidimensionnel.

WSM et WPM sont les premières méthodes qui indiquent comment associer un score à une alternative selon m critères. Dans ces deux approches, le résultat de la décision dépend des poids associés à chaque critère. De plus, ces deux approches ne proposent pas une méthode de calcul des poids des critères et elles ne précisent pas comment obtenir l'ensemble des alternatives.

2.2.2.2.2 Théorie de l'utilité multi-attributs (MAUT)

Dans cette partie, nous introduisons une fonction numérique dite **fonction d'utilité**. Elle permet de résoudre le problème de décision multicritère par une approche MAUT (Multi Attribute Utility Theory) [23]. Avant d'aller plus avant, nous aimerions expliquer en quoi consiste **l'utilité** ? Selon le dictionnaire français Larousse, **l'utilité** est l'aptitude d'un bien à satisfaire un besoin ou à créer les conditions favorables à cette satisfaction[2]. Donc, l'idée de base est de construire une fonction d'utilité dans un contexte de décision donné qui permet d'associer des scores ou des utilités aux alternatives auxquelles le décideur est confronté. Ces utilités permettront de classer les alternatives possibles.

Définition de la fonction d'utilité

Reprenons la relation $\succ$. Si nous avons deux alternatives a_i et a_j, si a_i est préférée à a_j alors nous écrivons $a_i \succ a_j$. Dans ce cas, le résultat qui est déterminé par le décideur est l'alternative a_i.

En fait, il existe différentes formes possibles pour cette relation. Elles varient en fonction de la relation de préférence entres les alternatives comme le montre le tableau suivant :

Tableau 1. Relation de préférences entre deux alternatives

[2] http://www.larousse.fr/dictionnaires/francais/utilité/80821

Expression formelle	Signification
$a_i \succ a_j$	a_i est préféré ou indifférence à a_j
$\neg (a_i \succ a_j) \wedge \neg (a_j \succ a_i)$	a_i et a_j sont incomparables
$(a_i \succ a_j) \wedge \neg (a_j \succ (a_i)$	a_i est préféré strictement à a_j

Avec ce type de représentation, la relation est souvent très difficile à manipuler. Donc, le terme d'**utilité** devient un concept qui simplifie et facilite la manipulation de cette représentation ainsi que le calcul. La fonction :

$$u : X \mapsto \mathbb{R} \tag{4}$$

Affecte, à l'ensemble des alternatives, un nombre réel indiquant l'**utilité** de l'alternative considérée. Plus cette valeur est élevée, plus l'alternative est préférée par le décideur. Formellement, nous écrivons :

$$\forall a_i, a_j \in X, a_i \succ a_j \Leftrightarrow u(a_i) \geq u(a_j) \tag{5}$$

La fonction d'utilité u doit prendre en compte les connaissances des alternatives pour réaliser le calcul. Des différents types d'informations, nous déduisons les différentes fonctions d'utilités *u*.

Fonction d'utilité multicritère

En réalité, la plupart des problèmes ont besoin d'une décision multicritère. Une fonction d'utilité d'un problème multicritère est une combinaison d'utilité de chaque critère. Nous définissons une fonction *U* sur un ensemble d'alternatives *X* ayant *m* critères comme suit :

$$\begin{aligned} &\forall\, a_1 = (x_1^1, x_2^1, \dots, x_p^1), a_2 = (x_1^2, x_2^2, \dots, x_p^2) \in X \\ &a_1 \succ a_2 \Leftrightarrow U(x_1^1, x_2^1, \dots, x_p^1) \succ U(x_1^2, x_2^2, \dots, x_p^2) \end{aligned} \tag{6}$$

Nous allons construire et décomposer une fonction d'utilité multicritère. Nous présentons ci-dessous deux méthodes principales.

- Décomposition additive

$$U(x_1, x_2, \dots, x_p) = \sum_{i=1}^{p} u_i(x_i) \tag{7}$$

- Décomposition multiplicative

$$U(x_1, x_2, \dots, x_p) = \prod_{i=1}^{p} u_i(x_i) \quad (8)$$

La nature de la décomposition est simple, pourtant le plus difficile est de déterminer une fonction de calcul de l'utilité pour chaque critère. Cette fonction varie en fonction de l'application et de ses critères. Après avoir déterminé la fonction d'utilité, nous calculons la valeur d'utilité de chaque alternative sur tous les critères. L'alternative qui porte la valeur maximale est celle qui sera choisie comme le choix final de la décision.

Nous avons présenté les deux types de méthodes dans l'approche « agréger puis comparer ». Nous allons aborder la deuxième famille d'approches de la décision multicritère à savoir « comparer puis agréger ».

2.2.2.3 Approche « comparer puis agréger »

Cette approche utilise une fonction de comparaison P afin de comparer consécutivement l'évaluation de deux alternatives pour chaque critère. Puis, elle combine ces valeurs pour déterminer quelle est la relation existant entre les alternatives a_1 et a_2 par une fonction d'agrégation $\mathcal{E}$. Cette deuxième catégorie consiste à faire :

$$\succ (a_1, a_2) = \mathcal{E}\left(P_1\big(g_1(a_1), g_1(a_2)\big), P_2\big(g_2(a_1), g_2(a_2)\big), \dots, P_m\big(g_m(a_1), g_m(a_2)\big)\right) \quad (9)$$

Ce type d'approche nécessite plusieurs méthodes de décision multicritère. L'AHP (Analytic Hierarchy Process) est une des méthodes de décision multicritère dans ce type d'approche. Elle compare les différents critères avant de les agréger afin d'obtenir le choix final. Par ailleurs, les méthodes de sur-classement sont les représentantes principales de cette approche. ELECTRE [24] et PROMETHEE [25] sont deux méthodes très connues dans le domaine de décision multicritère. Ces deux types d'approches donnent des résultats de sur-classement, par exemple a_i surclasse a_j indiquant a_i est préférée à a_j.

Nous allons décrire chacune de ces méthodes dans la partie suivante de la thèse.

2.2.2.3.1 Méthode d'AHP

L'AHP (Analytic Hierarchy Process) est une méthode proposée par Saaty [26]. Elle se base sur la décomposition d'un problème de décision multicritère en un système hiérarchique [27]. Cette approche tire sa force de sa similitude avec le mécanisme de décision de l'être humain, à savoir décomposition, jugement et synthèse.

Cette approche demande dans un premier temps de bien identifier l'objectif ainsi que les critères qui contribuent à l'atteinte de cet objectif. Puis, un modèle hiérarchique comme Figure 8 est construit en trois niveaux : le premier est l'objectif, le deuxième concerne les critères à évaluer, le troisième donne les solutions définies pour résoudre le problème.

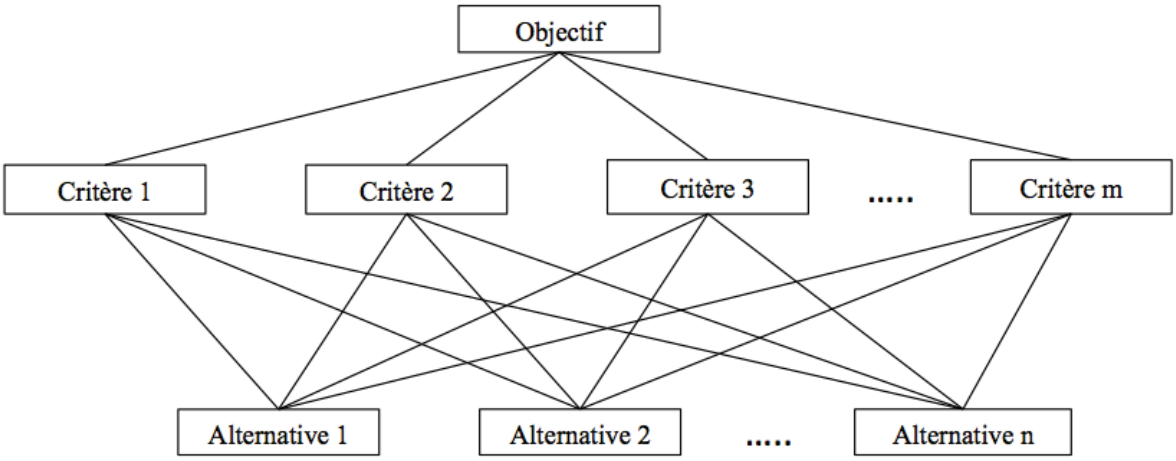

Figure 8. Modèle hiérarchique du problème

Ensuite, une matrice de comparaison deux à deux est construite. Nous comparons l'importance relative de tous les éléments appartenant à un même niveau de la hiérarchie pris deux par deux, par rapport à l'élément du niveau immédiatement supérieur. Pour chaque comparaison, nous devons choisir le critère le plus important et exprimer son jugement quant à son importance. Pour le mettre en œuvre, AHP utilise une approche de comparaison comportant une échelle allant de 1 (même niveau d'importance entre deux critères) à 9 (un critère est absolument plus important qu'un autre). La description détaillée de cette échelle est présentée dans la Figure 9. Le décideur se base sur ce principe pour déterminer le niveau d'importance entre les critères. Nous obtenons à la fin de cette étape une matrice de taille $m \times m$.

Intensity of Importance	*Definition*	*Explanation*
1	Equal Importance	Two activities contribute equally to the objective
2	Weak or slight	
3	Moderate importance	Experience and judgement slightly favour one activity over another
4	Moderate plus	
5	Strong importance	Experience and judgement strongly favour one activity over another
6	Strong plus	
7	Very strong or demonstrated importance	An activity is favoured very strongly over another; its dominance demonstrated in practice
8	Very, very strong	
9	Extreme importance	The evidence favouring one activity over another is of the highest possible order of affirmation
Reciprocals of above	If activity *i* has one of the above non-zero numbers assigned to it when compared with activity *j*, then *j* has the reciprocal value when compared with *i*	A reasonable assumption

Figure 9. Niveau d'importance relative entre les critères [27]

Nous passons à l'étape suivante consistant à déterminer les valeurs de priorités. La détermination des priorités des éléments de chaque matrice se fait par le calcul des vecteurs propres de la matrice. À partir de cette matrice, nous calculons le vecteur de priorité des m critères et aussi le poids de chaque critère [26].

Nous procédons de la même manière pour la comparaison deux à deux des alternatives par rapport à chaque critère. Cette phase nous donne une matrice de taille $n \times n$. Nous calculons le vecteur de priorité comme précédemment.

Nous allons ensuite vérifier la cohérence des jugements que nous venons d'obtenir dans les matrices de comparaison deux à deux. AHP introduit un paramètre spécial, appelé « **ratio de consistance** » CR (Consistency Ratio). Le ratio de consistance est le rapport de l'indice de consistance CI (Consistency Index) à l'indice de consistance aléatoire RI (Random Index).

$$CR = \frac{CI}{RI} \tag{10}$$

tel que :

$$CI = \frac{\lambda_{max} - n}{n - 1} \tag{11}$$

où *n* est le nombre de lignes ou colonnes de la matrice de comparaison deux à deux considérée, λ_{max} est la valeur propre maximale de la matrice de comparaison deux à deux. Concernant le paramètre RI, Saaty [16] a fourni des valeurs pour des matrices de comparaison deux à deux de tailles différentes. Selon Saaty [28], une matrice de comparaison deux à deux est consistante si $CR < 0{,}1$. Nous devons calculer le ratio de consistance pour chaque matrice afin de vérifier leur consistance. Dans le cas où une inconsistance est trouvée, le décideur devra reformuler ses jugements.

Si toutes les matrices que nous avons construites sont consistantes, nous faisons une synthèse des valeurs pour chaque alternative afin de donner le résultat de la décision. À partir de la matrice de comparaison des critères deux à deux, nous obtenons un vecteur de priorité contenant la liste des poids des critères $\{w_1, \dots, w_m\}$. À partir des matrices de comparaison des alternatives deux à deux pour chaque critère, nous obtenons aussi un vecteur de priorité représentant les priorités des alternatives selon le critère considéré, tel que $\langle P_h(a_i), \dots, P_h(a_n)\rangle$ qui est la liste de priorité de n alternatives sur le critère C_h. La meilleure alternative selon l'AHP est celle qui satisfait l'expression :

$$A^* = \max_{i=1,\dots,n} \{AHP(a_i)\} \tag{12}$$

Avec

$$AHP(a_i) = \sum_{h=1}^{m} P_h(a_i).w_h \quad (13)$$

Cette approche intègre la pondération des critères, mais nécessite l'intervention du décideur pour construire la matrice de comparaison deux à deux.

2.2.2.3.2 Méthode ELECTRE

L'origine de cette méthode remonte à 1965. Une équipe de recherche a travaillé sur le problème de décision multicritère. Pour résoudre ce problème, la méthode MARSAN (Méthode d'Analyse, de Recherche, et de Sélection d'Activités Nouvelles) a été proposée. Cette méthode utilise la somme des poids des critères pour sélectionner la nouvelle activité [29]. MARSAN présente un certain nombre de limitations qui ont été levées par la méthode ELECTRE I [30].

ELECTRE I permet d'identifier le sous-ensemble d'alternatives offrant le meilleur compromis des critères possible. Pour ce faire, la méthode introduit deux mesures, l'**indice de concordance** et **l'indice de discordance**. Nous considérons un ensemble A de *n* alternatives et m critères, nous définissons pour chaque critère une fonction d'évaluation g_h (où h = 1 à m). Pour chaque critère, nous avons un poids w_h qui indique l'importance du critère. L'indice de concordance pour deux alternatives a_i et a_j est noté par $C(a_i, a_j)$. Compris entre 0 et 1, il mesure la pertinence de l'assertion « a_i surclasse a_j », comme le montre la formule suivante :

$$C(a_i, a_j) = \frac{\sum_{\forall h:\, g_h(a_i) \geq g_h(a_j)} w_h}{Pds} \quad avec\ Pds = \sum_{h=1}^{m} w_h \quad (14)$$

L'indice de discordance $D(a_i, a_j)$ est défini par :

$$D(a_i, a_j) = 0\ si\ \forall h:\, g_h(a_i) \geq g_h(a_j) \quad (15)$$

Sinon

$$D(a_i, a_j) = \frac{1}{\delta} \times max_h\left[g_h(a_j) - g_h(a_i)\right] \quad (16)$$

avec δ indiquant la différence maximale entre le même critère pour deux alternatives données.

La relation de sur-classement pour ELECTRE I est construite par la comparaison des indices de concordance et de discordance à des seuils limites de concordance φ_C et de discordance φ_D. Ainsi, a_i surclasse a_j, si :

$$a_i \, S \, a_j \;\Leftrightarrow C(a_i, a_j) \geq \varphi_C \;\; et \;\; D(a_i, a_j) \leq \varphi_D \tag{17}$$

Nous obtenons alors un résultat qui contient les relations de sur-classement entre les alternatives. En nous basant sur cette liste, nous déduisons l'ensemble des alternatives pertinentes possibles.

La méthode ELECTRE I est à l'origine de plusieurs méthodes. ELECTRE I a fait ses preuves dans plusieurs domaines d'application [31]. Elle a évolué et donné naissance à une version non officielle, ELECTRE Iv. Cette version prend en compte la notion de seuil de veto. Une autre version dite ELECTRE IS est apparue par la suite [32]. C'est la version actuelle de la méthode ELECTRE pour le type de décision « choix ».

À la fin des années soixante, un nouveau problème de décision se pose, à savoir la planification des médias, ou la définition d'un plan de publicité. Comment établir alors un système adéquat de classement pour les périodiques (revues, journaux, etc.) ? Ceci a conduit à ELECTRE II [33], [34]. Quelques années plus tard, une nouvelle méthode de classement des actions a été élaborée : ELECTRE III [35], [36]. Sa nouveauté consiste en l'utilisation de pseudo-critères [37] et les relations de sur-classement binaires floues. ELECTRE IV [38] qui permettent de classer les actions sans utiliser les poids de l'importance relative entre les critères. C'est la seule approche ELECTRE qui n'utilise pas les coefficients.

À la fin des années soixante-dix, une nouvelle technique de tri des actions dans des catégories prédéfinies a été développée, la procédure trichotomie [39] basée sur un arbre de décision. Ainsi, afin d'aider à la prise de décision, accorder ou pas un prêt, une méthode spécifique, ELECTRE A, a été conçue et appliquée. Enfin, ELECTRE TRI s'est appuyée sur les travaux antérieurs [40].

Pour chaque type de décision, ELECTRE propose une méthode particulière qui s'adapte selon le contexte de l'application. Chacune est définie par des approches mathématiques ou des modèles formels. Parmi les différents types d'ELECTRE, nous voyons qu'ils prennent la décision parmi un ensemble d'alternatives déterminé sur des critères multiples. Bien que le processus de décision avec ELECTRE formalise bien la décision, elle présente l'inconvénient d'utiliser les poids des critères. Par conséquent, malgré l'apport de cette méthode, le décideur doit quantifier les poids des critères ce qui n'est pas toujours une tâche aisée. Par ailleurs, le résultat de cette méthode n'est que le choix, le classement ou le tri obtenu selon l'analyse implicite du mécanisme de décision. Dans le cas où l'utilisateur ou un autre expert a une autre préférence, ELECTRE ne peut pas prendre en compte cet aspect.

2.2.2.3.3 Méthode PROMETHEE

PROMETHEE est l'acronyme de « Preference Ranking Organisation METHod for Enrichment Evaluations ». PROMETHEE représente une famille de méthodes d'aide à la décision multicritère. Les éléments de base de cette méthode ont été présentés par Jean-Pierre Brans en 1982 [41], [42]. Cette méthode a été utilisée avec succès dans de nombreuses recherches dans le domaine de la décision. Elle est semblable à la méthode ELECTRE, dans le sens où elle vise à construire une relation de sur-classement dans l'espace des alternatives, par contre, elle en diffère par la nature de cette relation. Contrairement à la relation de sur-classement de la méthode d'ELECTRE qui est binaire (dans le sens où une relation soit surclasse ou non), la

relation obtenue par la méthode PROMETHEE est une relation de sur-classement valuée. c'est-à-dire une alternative surclasse une autre en associant une valeur qui indique une intensité de préférence numérique. De ce fait, le principe de PROMETHEE reprend presque celui de la théorie de l'utilité multi-attributs. En fait, MAUT associe à chaque alternative une valeur d'utilité numérique. Ces valeurs numériques nous permettent de classer ou d'identifier les relations de sur-classement entre deux ou plusieurs alternatives.

PROMETHEE I (classement partiel) et PROMETHEE II (classement complet) ont été présentées pour la première fois en 1982 lors d'une conférence à Québec, Canada. Nous allons donc présenter en détail le principe de ces deux PROMETHEE. Quelques années plus tard, JP Brans et B. Mareschal ont développé PROMETHEE III (classement basé sur les intervalles) et PROMETHEE IV (cas continu) [43]. En 1992 et 1994, JP Brans et B Mareschal ont suggéré deux extensions: PROMETHEE V (MCDA comprenant les contraintes de segmentation) [44] et PROMETHEE VI (représentation du cerveau d'humain) [45].

La méthode PROMETHEE fonctionne comme suit :

PROMETHEE appartient à la catégorie « comparer puis agréger ». Elle compare d'abord l'évaluation des deux alternatives sur chaque critère. Comme nous sommes dans un contexte de multicritère, il nous faut d'abord déterminer une matrice d'évaluation comme montré dans le Tableau 2.

Tableau 2. Tableau d'évaluation des alternatives

Alternative	**Critère 1**	**….**	**Critère *m***
Alternative 1	$g_1(a_1)$	….	$g_m(a_1)$
….	….	….	….
Alternative *n*	$g_1(a_n)$	….	$g_m(a_n)$

Le tableau d'évaluation est, dans tous les cas, la base des méthodes de décision multicritère. Le décideur pondère ensuite chaque critère selon sa perception. Le travail d'estimation des poids des critères ne se base sur aucun facteur. Cela ne fait appel qu'à l'expérience, l'intuition et aux priorités du décideur. PROMETHEE ne considère que la structure préférence dans la comparaison par paire de critère. La différence entre deux alternatives sur un critère est calculée par :

$$d_h(a_i, a_j) = g_h(a_i) - g_h(a_j) \tag{18}$$

d_h est la différence entre deux alternatives a_i et a_j

Une fois que la différence entre deux alternatives est calculée selon un critère, PROMETHEE considère cette valeur pour évaluer le degré de préférence de ces deux alternatives. Si la différence est petite, le décideur donne un degré de préférence négligeable voire même ne donne pas de préférence. Plus la différence est grande, plus le degré de préférence augmente. Il nous faut examiner les degrés de préférence dans un intervalle entre 0 et 1. Le décideur détermine alors une fonction de préférence pour chaque critère :

$$P_h(a_i, a_j) = F_h[d_h(a_i, a_j)] \qquad \forall\, a_i, a_j \in A \tag{19}$$

$$0 \leq P_h(a_i, a_j) \leq 1$$

La fonction F_h permet au décideur d'estimer la relation de préférence entre deux alternatives sur un critère h. L'auteur a défini 6 types de fonctions de préférences comme la montre la Figure 10 :

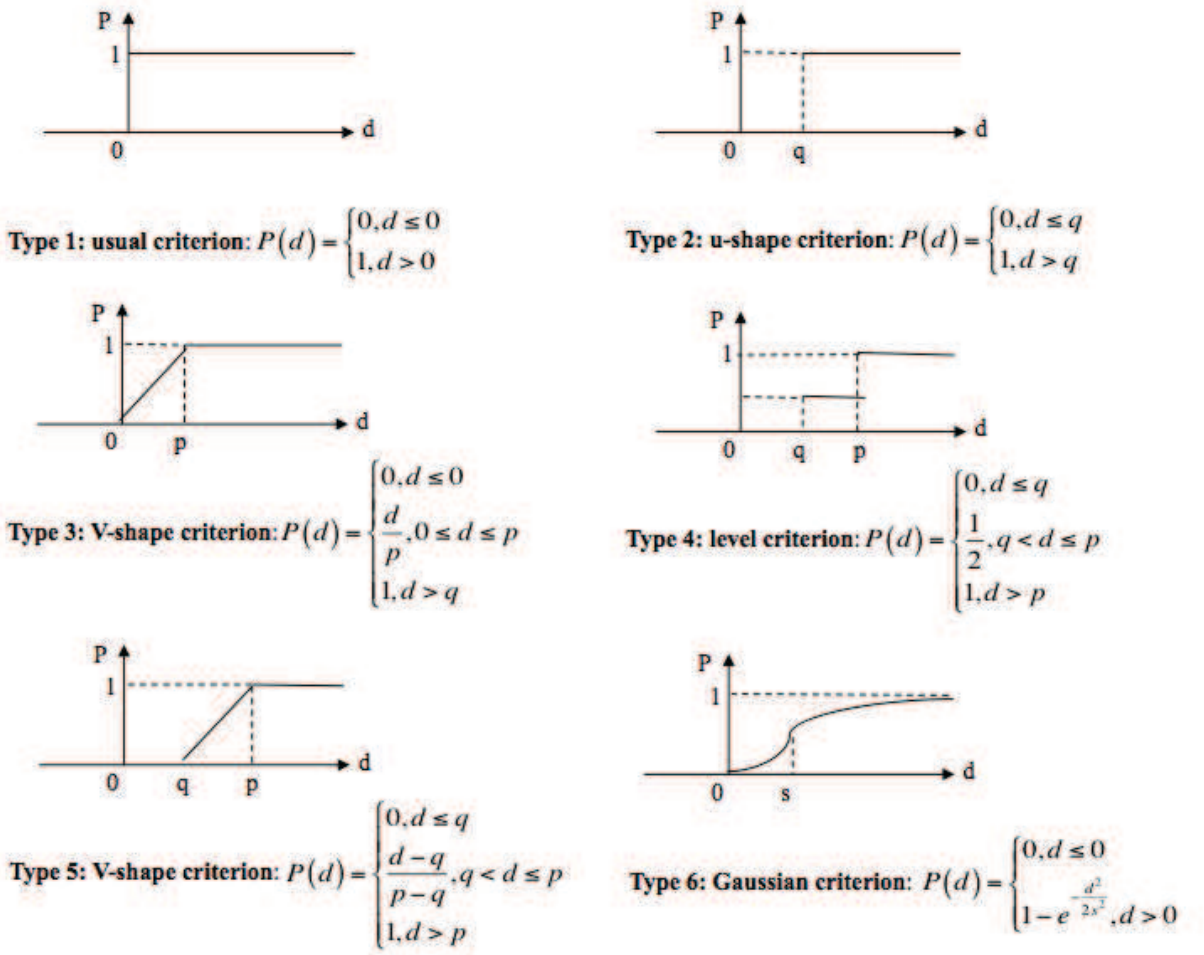

Figure 10. Six fonctions de préférence [46]

Lorsqu'une fonction de préférence a été associée à chaque critère, toutes les comparaisons entre toutes les paires d'alternatives peuvent être effectuées pour tous les critères.

$$\begin{cases}\pi(a_i, a_j) = \sum_{h=1}^{m} P_h(a_i, a_j) \times w_h \\ \pi(a_j, a_i) = \sum_{h=1}^{m} P_h(a_j, a_i) \times w_h\end{cases} \quad (20)$$

$\pi(a_i, a_j)$ représente le degré de préférence de a_i par rapport à a_j pour m critères. Afin de positionner chaque alternative par rapport à toutes les autres alternatives, deux valeurs sont calculées :

$$\begin{cases} \phi^{+}(a_i) = \dfrac{1}{m-1} \times \sum_{x \in A} \pi(a_i, x) \\ \phi^{-}(a_i) = \dfrac{1}{m-1} \times \sum_{x \in A} \pi(x, a_i) \end{cases} \tag{21}$$

Chaque alternative a_i doit être comparée avec (*m-1*) différentes alternatives dans l'ensemble *A*. Le flux positif de préférence $\phi^{+}(a_i)$ estime la manière dont l'alternative a_i est préférée à toutes les autres alternatives tandis que le flux négatif de préférence $\phi^{-}(a_i)$ quantifie la manière dont a_i n'est pas préférée à toutes les autres alternatives.

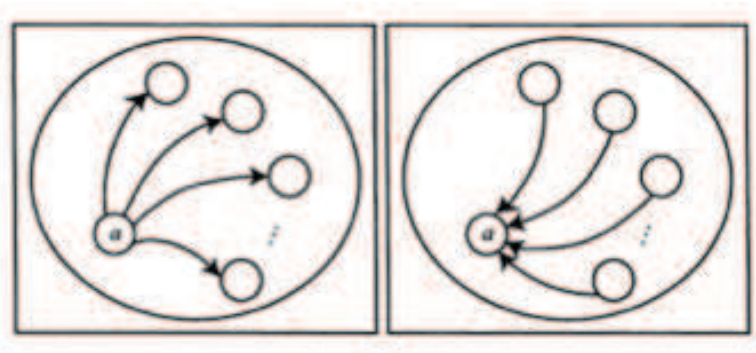

Figure 11. Flux positif (gauche) et flux négatif (droite)

En d'autres termes, les flux positifs et négatifs représentent les intensités de préférences qu'une alternative a_i possède vis-à-vis de l'ensemble des autres alternatives. Ils sont utilisés différemment par les méthodes PROMETHEE I et PROMETHEE II afin de définir des relations de sur-classement.

PROMETHEE I : elle permet un classement partiel des alternatives dans un ensemble d'alternatives. Elle consiste à les classer par flux positifs croissants et flux négatifs décroissants.

$$\begin{aligned} a_i \, P \, a_j &\Leftrightarrow \begin{bmatrix} \phi^{+}(a_i) > \phi^{+}(a_j) \;\; et \;\; \phi^{-}(a_i) < \phi^{-}(a_j) \\ \phi^{+}(a_i) > \phi^{+}(a_j) \;\; et \;\; \phi^{-}(a_i) = \phi^{-}(a_j) \\ \phi^{+}(a_i) = \phi^{+}(a_j) \;\; et \;\; \phi^{-}(a_i) < \phi^{-}(a_j) \end{bmatrix} \\ a_i \, I \, a_j &\Leftrightarrow \phi^{+}(a_i) = \phi^{+}(a_j) \;\; et \;\; \phi^{-}(a_i) = \phi^{-}(a_j) \\ a_i \, R \, a_j &\Leftrightarrow \begin{bmatrix} \phi^{+}(a_i) > \phi^{+}(a_j) \;\; et \;\; \phi^{-}(a_i) > \phi^{-}(a_j) \\ \phi^{+}(a_i) < \phi^{+}(a_j) \;\; et \;\; \phi^{-}(a_i) < \phi^{-}(a_j) \end{bmatrix} \end{aligned} \tag{22}$$

P, I, R sont respectivement des relations de préférence, d'indifférence et d'incomparabilité.

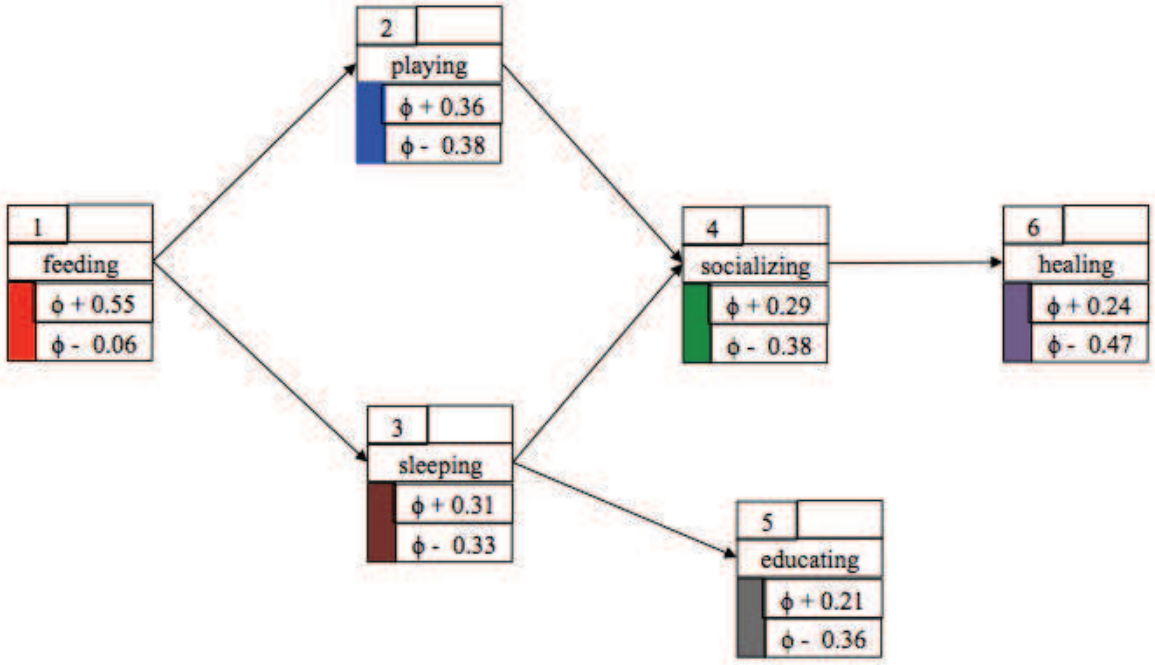

Figure 12. Classement partiel avec PROMETHEE I

PROMETHEE II : PROMETHEE II ne se concentre que sur les deux relations P et I. Elle est appliquée dans le cas où le décideur a besoin d'un classement total. La nouvelle grandeur, flux net, est calculée comme suit :

$$\phi(a_i) = \phi^+(a_i) - \phi^-(a_j) \tag{23}$$

Le classement total est défini par l'expression :

$$\begin{aligned} a_i \, P \, a_j \;&\Leftrightarrow \phi(a_i) > \phi(a_j) \\ a_i \, I \, a_j \;&\Leftrightarrow \phi(a_i) = \phi(a_j) \end{aligned} \tag{24}$$

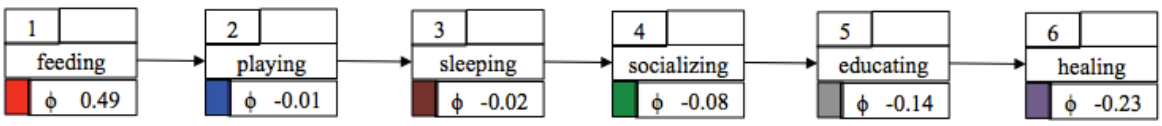

Figure 13. Classement total avec PROMETHEE II (cas du Tamagotchi, voir Chapitre 5)

Nous remarquons que la méthode PROMETHEE II se rapproche nettement des méthodes d'utilité, le flux net étant l'utilité associée à l'alternative a_i. Par contre, la méthode PROMETHEE I garde toujours le principe de sur-classement grâce aux flux positifs et négatifs similaires aux notions de concordance et discordance des méthodes ELECTRE.

2.2.2.4 Autres approches

Cette section va nous permettre de récapituler les autres approches existantes.

2.2.2.4.1 Méthode des métriques pondérées

La méthode des métriques pondérées (Method of Weighted Metrics) sélectionne un vecteur de critères qui minimise la distance à une alternative de référence $R = (r_1, r_2, \dots, r_m)$ qui est choisie par le décideur en fonction de ses préférences [47]. Nous allons calculer la distance Δ_d entre R et chaque alternative a_i parmi les n alternatives. Une alternative a_i est représentée par le vecteur de m critères $A(a_i)$. Cette distance est définie comme suit :

$$\Delta_d(a_i) = \left(\sum_{h=1}^{m} w_h \times \left(R_h - A_h(a_i) \right)^d \right)^{\frac{1}{d}} \tag{25}$$

Selon la valeur de *d*, nous disposons des cas suivants :

- d = 1 : la distance de Manhattan

$$\Delta_1(a_i) = \sum_{h=1}^{m} (w_h \times |R_h - A_h(a_i)|) \tag{26}$$

- d = 2 : la distance euclidienne

$$\Delta_2(a_i) = \sqrt{\left(\sum_{h=1}^{m} w_h \times \left(R_h - A_h(a_i) \right)^2 \right)} \tag{27}$$

- d = ∞ : la distance de Tchebychev

$$\Delta_\infty(a_i) = \max_{h \in [1,m]} (w_h \times |R_h - A_h(a_i)|) \tag{28}$$

La méthode des métriques utilise aussi le poids des critères. De plus, la méthode est très sensible au choix de l'alternative de référence *R*. La méthode des métriques dépend fortement à la fois du choix de l'alternative de référence et de la pondération des poids des critères utilisés dans le calcul de la distance.

2.2.2.4.2 Méthode de programmation par buts

La méthode de programmation par buts (Goal Programming) [48] est une extension de la méthode de programmation linéaire au cas multi-objectif. Elle vise à caractériser la fixation d'un but à atteindre pour chaque critère. En d'autres termes, cette méthode consiste à minimiser les déviations de chaque critère par rapport à son but. Mathématiquement, cette méthode est décrite comme suit :

Nous définissons m valeurs réelles $C_1, C_2, \dots, C_m$ qui représentent les buts à atteindre pour les fonctions de critères $f_1, f_2, \dots, f_m$. Nous créons ensuite $m \times 2$ variables $\delta_1^+, \delta_1^-, \dots, \delta_m^+, \delta_m^-$ qui sont les variables de déviation représentant le degré de déviation de l'alternative a_i par rapport aux fonctions des critères $f_1, f_2, \dots, f_m$. Nous obtenons :

$$\begin{cases} minimiser(\delta_1^+ \; ou \; \delta_1^-, \dots, \delta_m^+ \; ou \; \delta_m^-) \\ f_1(a_i) = C_1 + \delta_1^+ - \delta_1^- \\ \dots \\ f_m(a_i) = C_m + \delta_m^+ - \delta_m^- \\ \delta_h^+ \geq 0 \, , \delta_h^- \geq 0 \, , \delta_h^+ \times \delta_h^- = 0 \; avec \; h \in \{1, \dots, m\} \end{cases} \tag{29}$$

Cette approche est ramenée au problème de minimisation d'un vecteur de déviation. La difficulté de cette méthode réside dans la définition des fonctions des critères. De plus, elle demande des valeurs des buts à atteindre pour pouvoir être appliquée.

2.2.2.4.3 Méthode évolutionnaires

Les méthodes évolutionnaires sont basées sur les algorithmes génétiques (AG). Les AG ont été introduits par [49]. Ils sont inspirés de la théorie de l'évolution des espèces vivantes. Le but est d'obtenir une solution approchée à un problème d'optimisation lorsqu'il n'existe pas de méthode exacte (ou que la solution est inconnue). Le principe général de cette approche est très simple : les êtres vivants sont soumis au mécanisme de compétition qui sélectionne les individus les plus adaptés à leur environnement. Les individus les plus robustes auront plus d'occasion de se reproduire et de transmettre leurs gènes favorables à leurs enfants. Par conséquent, au fil des générations, ce sont les individus portant des gènes adéquats qui sont les plus populaire dans la population. Par ailleurs, lors des générations, il pourra apparaître de nouveaux individus qui portent des nouveaux gènes, cela permet d'enrichir la population. Cette méthode utilise les concepts biologiques suivants pour résoudre un problème d'optimisation.

- Individu : représente une alternative potentielle. Il est composé de plusieurs gènes dont les valeurs peuvent être un entier, un bit (0 ou 1) ;
- Population : l'ensemble des individus traités ;
- Fonction d'évaluation : une fonction qui associe à chaque individu une valeur numérique représentant sa qualité ;
- Génération : une itération de l'algorithme génétique en effectuant des opérateurs génétiques dans une population ;
- Sélection : associe à chaque individu une probabilité de reproduction à chaque génération, cette valeur est corrélée avec la qualité d'individu ;
- Remplacement : sélectionne l'ensemble des individus qui seront conservés à la nouvelle génération ;

- Mutation : un opérateur qui permet de modifier aléatoirement les gènes d'un individu ;
- Croisement : un opérateur qui permet de combiner un ou plusieurs individus parents pour former un ou plusieurs individus enfants.

La Figure 14 décrit le fonctionnement général d'un algorithme génétique. En l'appliquant à notre problème de décision, nous pouvons générer aléatoirement un ensemble d'alternatives. Les alternatives doivent être évaluées par les fonctions d'adaptation. Puis l'algorithme génétique entre dans une boucle générationnelle (chaque itération étant une génération) dans laquelle les différentes phases sont exécutées consécutivement. L'algorithme génétique s'arrête lorsqu'il arrive à un nombre d'itérations (générations) défini. À la fin de l'algorithme, les meilleurs individus ou les meilleures alternatives sont retenus comme solution suggérée au décideur.

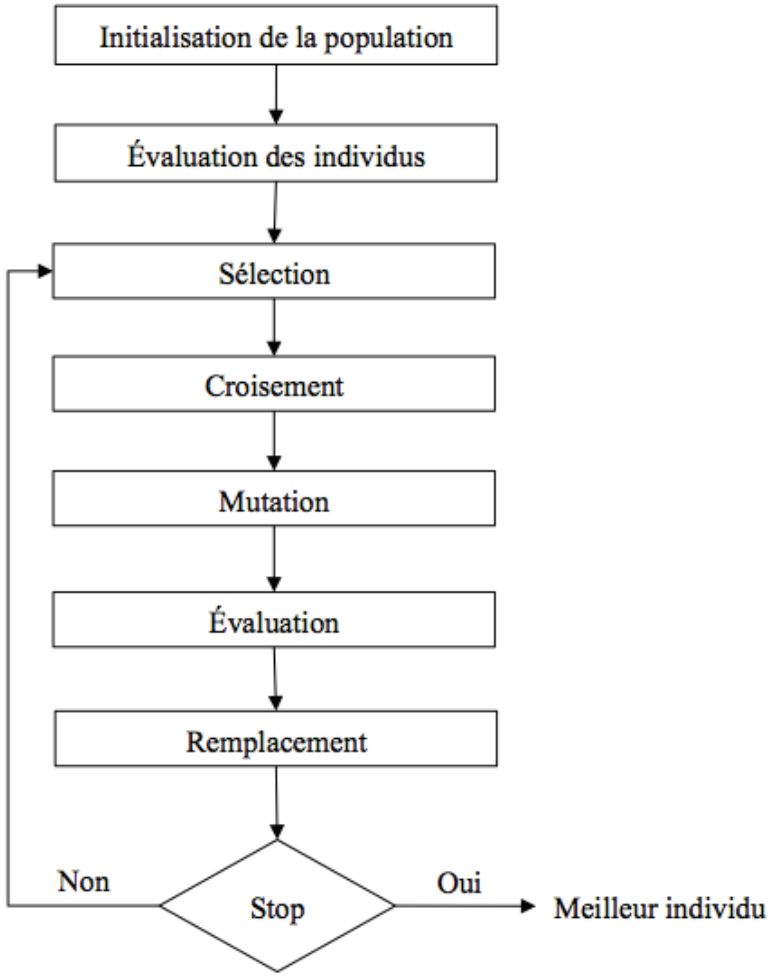

Figure 14. Fonctionnement d'un algorithme génétique [50]

L'algorithme génétique ne convient qu'aux cas suivants : le temps de calcul de la fonction d'évaluation doit être court car cette fonction est utilisée de nombreuses fois. Le nombre initial d'individus est aussi un des facteurs qui influence le résultat de cet algorithme.

2.3 ANALYSE DES METHODES EXISTANTES

Nous avons présenté brièvement les deux approches : la recommandation dans la section 2.2.1 et la prise de décision multicritère dans la section 2.2.2. Nous allons analyser ces deux approches pour choisir celle qui est la plus adaptée à nos travaux.

Parmi les deux approches d'adaptation des sections 2.2.1 et 2.2.2, nous allons tenter de choisir l'approche qui convient le mieux à nos besoins d'identification de la suite de situations optimales. Si nous appliquons la recommandation pour mettre en œuvre l'adaptation dans une exécution interactive à base de situations, nous allons filtrer les situations pertinentes pour un utilisateur. La prise de décision multicritère joue un rôle similaire à celui joué par la recommandation ; elle doit suggérer à l'utilisateur une situation appropriée afin d'adapter l'exécution de l'application.

La recommandation donne les situations pertinentes à un instant donné. Elle permet de réutiliser les informations recueillies dans les exécutions précédentes des utilisateurs ainsi que les caractéristiques des situations. C'est un avantage de la recommandation. Bien que nous soyons dans un contexte interactif où l'utilisateur est l'acteur principal, nous avons aussi à prendre en compte la logique du concepteur de l'application. Il connaît la structure de l'application et il sait comment adapter structurellement l'exécution à un moment donné. Dans ce cas, la recommandation ne nous permet pas de considérer les préférences au niveau structurel du concepteur. D'autre part, la prise de décision n'utilise pas les informations recueillies comme la recommandation mais elle permet de dialoguer directement avec l'utilisateur pour savoir ce qu'il veut faire. Par ailleurs, nous pouvons mettre en place le choix du concepteur ou encore le choix des différents acteurs dans le système de décision et appliquer des mécanismes particuliers pour agréger ces choix.

Nous avons résumé dans le Tableau 3, une synthèse de deux approches (recommandation et prise de décision) d'adaptation en tenant compte des besoins dans notre contexte.

Tableau 3. Synthèse des approches d'adaptation selon nos besoins

Approches d'adaptation	En temps réel	Prise en compte des informations passées	Agrégation des différents résultats
Recommandation	Oui	Oui	Non
Prise de décision	Oui	Non	Oui

D'après l'analyse du Tableau 3 et le résumé de ces deux approches de recommandation et de prise de décision, malgré le fait que la recommandation soit une approche efficace, elle ne répond pas à tous nos besoins d'adaptation. Elle peut effectuer une adaptation en temps réel et elle peut tenir compte des connaissances dans le passé. Pourtant, elle n'arrive pas à supporter une combinaison des différents résultats d'adaptation venant des différents acteurs lors de l'exécution. Dans ce cas, l'adaptation par la recommandation ne nous permet pas de résoudre ce besoin. En outre, la prise de décision est aussi une méthodologie utilisée pour effectuer l'adaptation. Elle peut fonctionner en temps réel, mais son calcul ne se base pas sur les informations dans le passé. Par ailleurs, la prise de décision supporte des approches pour récupérer instantanément les informations des acteurs et agréger des résultats de choix de plusieurs acteurs. Entre deux approches : la recommandation et la prise de décision, chacune comporte des points forts et faibles selon les besoins d'adaptation dans notre contexte. Bien que la prise de décision ne prenne pas en compte les informations situées dans le passé, elles pourront être intégrées pour surmonter ce besoin.

En analysant les avantages ainsi que les inconvénients de ces deux approches dans le contexte de cette thèse, nous avons donc décidé de ne considérer que l'approche de prise de décision pour mettre en œuvre notre objectif principal : le choix d'une situation pour adapter l'exécution d'une application interactive à base de situations.

Le Tableau 4 présente une synthèse des avantages ainsi que les inconvénients pour chaque méthode de décision multicritère mentionnée dans la section 2.2.2.

Tableau 4. Analyse des méthodes de décision multicritère

Méthode	**Avantages**	**Inconvénients**
WSM	Simple à implémenter	Elle ne peut s'appliquer que dans le cas monodimensionnel (une seule unité de mesure pour tous les critères).
WPM	Simple à implémenter	Elle peut être appliquée dans les 2 cas : mono et multidimensionnel
MAUT	Simple à appliquer	Difficulté lors de détermination des fonctions d'utilité pour chaque critère
AHP	Elle permet de calculer les poids des critères ainsi que classer les alternatives en priorité Possibilité d'ajouter d'autres critères	Dépend strictement de l'intuition de l'utilisateur
ELECTRE	Plusieurs versions de ELECTRE qui conviennent aux différentes applications.	Dépend du poids des critères qui ont été obtenus par la détermination de l'utilisateur
PROMETHEE	Des versions pour différents contextes	Dépend du poids des critères et le choix des fonctions de préférences
Des métriques pondérées	Elle se base uniquement sur le calcul de distance pour mettre en œuvre le problème de choix. Le calcul de distance est facile à appliquer	Elle a besoin des poids des critères en trouvant une solution référence
Programmation par buts	Approche mathématique visant à résoudre le problème de minimisation	Bien définir les fonctions des critères
Évolutionnaires	Répéter plusieurs d'itérations permettant de trouver la solution	Prendre beaucoup de temps, elle ne convient pas aux applications ayant besoin de trouver une solution en temps réel

Bien que chaque méthode comporte des avantages et des inconvénients particuliers, elles ne sont pas toutes applicables dans le contexte de la thèse. En nous basant sur le Tableau 4, nous allons positionner notre besoin de décision multicritère à la section 2.4.

2.4 POSITIONNEMENT

Notre thèse s'intéresse à la décision multicritère dans une application interactive à base de situations. Pendant son déroulement, la décision multicritère vise à enchainer les situations en

vue d'atteindre un objectif défini. Cet objectif est évalué par plusieurs critères. Chaque critère est pondéré différemment. Cette pondération est utilisée comme une information lors de la décision. Par ailleurs, quand une situation est terminée, nous avons besoin de choisir la meilleure situation suivante dans l'ensemble des situations candidates (alternatives potentielles). En regardant les alternatives, nous pouvons expliquer pourquoi le système va choisir cette situation en nous basant sur une mesure ou une évaluation. En outre, nous sommes dans un contexte interactif où l'utilisateur joue un rôle très important. De plus, le concepteur s'occupe aussi la participation à la prise de décision car il connaît bien l'application. En effet, c'est le concepteur qui a conçu l'application et les situations, il a une base de connaissance autour de l'aspect de structure. Il peut donc donner son choix sur le point de vue du concepteur. Dans le cas d'une décision automatique, nous pouvons également imaginer qu'un assistant intelligent observe l'état du système en cours d'exécution et applique une des techniques multicritères pour aider à la prise de décision.

En combinant la décision multicritère et nos besoins, nous définissons alors les propriétés indispensables d'un processus de décision multicritère :

- Optimisé : il nous donne une alternative qui répond à notre besoin et au contexte de l'application. L'alternative choisie est optimisée selon un modèle ;
- Interactif : un mécanisme de décision est interactif s'il considère aussi les avis de l'utilisateur ;
- Collectif : cette troisième propriété vise à tenir compte des préférences venant de différents décideurs.

En fonction de ces trois propriétés ci-dessus et en analysant le processus de décision de la Figure 7, nous avons identifié quatre caractéristiques indispensables d'un système d'aide à la décision multicritère:

- **(C1)** pondération des critères ;
- **(C2)** ensemble des alternatives possibles ;
- **(C3)** choix final satisfaisant au mieux tous les critères ;
- **(C4)** choix final représentant un choix collectif, combinaison des différents choix des acteurs de l'application : concepteur, système, utilisateur.

Le Tableau 5 compare les méthodes abordées dans ce chapitre selon les caractéristiques ci-dessus.

Tableau 5. Analyse des méthodes de décision multicritère existantes selon quatre caractéristiques

Méthodes	**C1**	**C2**	**C3**	**C4**
WSM	non	non	oui	non
WPM	non	non	oui	non
MAUT	non	non	oui	non
AHP	existe mais très intuitif : pondération par le décideur	non	oui	non
ELECTRE	non	oui	oui	non
PROMETHEE	non	non	oui	non
Méthodes des métriques pondérées	non	non	oui	non

Méthodes de programmation par buts	pas besoin	non	oui	non
Méthodes évolutionnaires	non	oui mais très aléatoire et dépend du nombre de générations	oui	non

L'analyse du Tableau 5 montre que ces méthodes ne se concentrent que sur la caractéristique (C3) et ne considèrent pas vraiment (C1) et (C2). Par ailleurs, aucune méthode ne prend en compte la caractéristique (C4), alors que nous avons besoin d'un choix final satisfaisant non seulement pour l'utilisateur mais aussi pour le concepteur et le système. L'analyse de l'état de l'art montre donc qu'il n'existe pas de système d'aide à la décision multicritère susceptible de prendra en compte les caractéristiques manquantes des méthodes existantes comme le montre le Tableau 5. En d'autres termes, un système d'aide à la décision multicritère intégrant dans une application interactive à base situations doit disposer de :

- La liste des poids des critères ;
- La liste des alternatives possibles ;
- L'algorithme de choix ;
- Le choix final collectif.

Comme nous l'avons indiqué, nous sommes amenés à lever les trois verrous suivants :

1. Comment pondérer automatiquement des critères ?
2. Comment déterminer automatiquement des alternatives ?
3. Comment obtenir un choix collectif ?

2.5 CONCLUSION

Dans ce chapitre, nous avons rappelé les principes fondamentaux de la décision multicritère. Nous avons présenté les différentes méthodes qui permettent de trouver les solutions qui s'accordent au mieux avec les préférences du décideur. Dans le contexte de cette thèse, nous avons besoin d'un système d'aide à la décision multicritère dans une application interactive à base de situations. En étudiant les techniques existantes, nous avons identifié leurs éléments manquants

Le chapitre suivant introduit un mécanisme de gestion des informations historiques des anciennes exécutions qui a pour but de fournir les données nécessaires pour répondre aux trois questions de la fin de la section 2.4.

CHAPITRE 3 MECANISMES DE GESTION DE TRACES

SOMMAIRE

Lors de l'interaction entre un utilisateur et un environnement numérique, de nombreuses données sont générées. Ces données, appelées traces, peuvent être très abondantes, et sont porteuses d'information qui pourra être utile pour l'analyse et l'adaptation de la logique d'exécution d'un programme pris dans un contexte particulier. Le système à base de traces définit ainsi la façon d'observer, de collecter, d'analyser… les traces pertinentes qui permettront d'adapter l'exécution de l'application en conformité avec le contexte spécifique.

Dans ce chapitre, nous commencerons par présenter brièvement des notions employées dans cette thèse, notamment sur les notions de trace et de leurs significations. La notion de trace en général apparait dans tous les domaines de recherche liés au paradigme documentaire, à l'étude des usages, à l'ingénierie des connaissances. Nous nous intéressons à la définition du terme *trace* en général tout en l'étendant pour définir le terme trace dans le domaine informatique. Nous introduirons, ensuite, une vue générale d'un système à base de traces. Enfin, nous positionnerons nos besoins par rapport aux systèmes à base de traces existant dans la littérature.

3.1 INTRODUCTION

L'idée de conserver des enregistrements des opérations effectuées par un ordinateur au cours de son exécution n'est pas nouvelle. Par exemple, les fichiers de log (logfiles) contiennent toutes les opérations effectuées lors de l'exécution d'un programme. Ceci permet de disposer d'un outil qui facilite la compréhension des opérations effectuées par l'ordinateur. De plus, il s'agit non seulement de comprendre les opérations effectuées par l'ordinateur mais aussi, les interactions entre les utilisateurs et l'ordinateur. Les travaux menés en IHM ont permis de formaliser ces aspects. L'avènement des environnements numériques de travail et des réseaux sociaux a fait émerger des besoins d'adaptation de l'exécution de programmes en fonction du comportement de l'utilisateur et/ou d'un contexte déterminé.

Comme l'indiqué à la section 1.1.1.2, l'adaptation permet au système interactif de maintenir la continuité de l'exécution d'une application interactive en améliorant la pertinence et la performance du système. La structuration au moyen des situations nécessite de disposer de mécanismes d'adaptation particuliers permettant de contrôler les comportements du système ainsi que ceux de l'utilisateur durant le fonctionnement de l'application. Par « adaptation », nous désignons la capacité d'un système à s'adapter par ses services et ses ressources aux besoins de l'utilisateur et aux exigences du concepteur. Il existe de nombreuses approches pour l'adaptation telles que les approches de décision multicritère décrites dans le Chapitre 2. Elles nécessitent la prise en compte des besoins de l'utilisateur grâce à des méthodes directes demandant directement l'avis à l'utilisateur. Ce type d'approches a l'avantage d'être explicite pour l'utilisateur, mais leur grand inconvénient est qu'elles exigent de l'utilisateur qu'il ait un haut niveau de connaissance sur les choix. Une autre approche, consiste à observer les comportements de l'utilisateur ainsi que leur contexte d'exécution pendant le déroulement de l'application. L'observation a pour objectif d'exploiter ces informations pour pouvoir adapter l'environnement. L'observation de l'exécution d'une application exige la collecte d'informations afin de prendre en compte le « quand », le « comment », le « quoi », le « pourquoi » et pour un « qui » particulier dans une activité réalisée [51]. Ces informations ou traces vont faire l'objet d'une analyse directe ou/et de traitements en vue de permettre une exécution adaptative. Les traces informatiques sont à la base de ce type d'approche mais n'ont pas fait l'objet d'une étude approfondie en tant qu'objets possédant des propriétés propres et des méthodes propres [52].

Nous mettons en œuvre des traitements exploitant les traces dans une application interactive à base de situations. Pour cela, nous allons répondre aux questions suivantes : comment récupérer des données dans les traces, faire les transformations et visualiser les traces ? À cet effet, une architecture des Systèmes à Base de Traces sera présentée dans ce chapitre.

Nous présentons un état de l'art des applications informatiques utilisant ou fournissant des informations récupérées lors de leurs exécutions. Cependant, il est difficile de fournir un état de l'art exhaustif sur le problème de traces au vu du grand nombre d'applications concernées et des domaines de recherches impliqués. Nous nous sommes fixés comme objectif de couvrir le maximum de types d'application disposant de systèmes traçants ou à base de traces.

3.2 DEFINITIONS DES TRACES NUMERIQUES

Dans sa définition la plus générale, selon le dictionnaire Larousse, une trace est « une suite d'empreintes ou de marques que laisse le passage d'un être ou d'un objet ; marque laissée par une action quelconque ; ce à quoi on reconnait que quelque chose a existé ; ce qui subsiste d'une chose passée ». Le terme trace fait ainsi référence à plusieurs sens [53] : en littérature, trace est ce qui subsiste de quelque chose du passé sous la forme de débris, des vestiges, etc.; en psychologie, la trace désigne ce qui subsiste dans la mémoire d'un événement passé[3]. Nous disons que le mot trace regroupe beaucoup de sens dans plusieurs domaines différents. Pourtant, la notion générale de trace la plus souvent utilisée est « une trace est la marque laissée par une activité ». Parmi les différentes définitions, nous nous focalisons sur la notion de **trace numérique** dans les systèmes d'information et qui sera la marque laissée lors d'une activité médiée par un dispositif informatique [53]. Enfin, la **trace numérique** est intimement liée aux notions d'historique et de traçabilité, ce qui nous permet de dire: « la **trace numérique** est trace de l'activité d'un utilisateur qui utilise un outil informatique pour mener à bien cette activité, s'inscrivant sur un support numérique » [54]. Par soucis de simplification, une trace désignera une trace numérique dans la suite du manuscrit.

3.3 SYSTEMES A BASE DE TRACES MODELISEES

En retenant le principe de trace, nous nous donnons la possibilité de gérer ces informations en tant que telles au sein d'un environnement informatique, nommé Système à Base de Traces (SBT). Le SBT est une solution générique au problème de modélisation et de manipulation des traces. Afin de définir de tels systèmes, nous définissons et formalisons tout d'abord les concepts de traces et de modèle de traces puis nous présentons le principe du système à base de traces modélisées en en précisant les étapes.

3.3.1 MODELISATION DE TRACE

Dans notre contexte, une trace est issue de l'observation d'une activité réalisée par un utilisateur pendant l'exécution de l'application. Elle est composée des objets situés les uns par rapport aux autres dans un intervalle de temps déterminé, appelé extension temporelle. Lors de

[3] http://www.larousse.fr/encyclopedie/nom-commun-nom/trace/98060

l'exécution d'une application, l'utilisateur effectue une ou plusieurs activités, chacune est associée à un observé comme le montre la Figure 15.

Extension temporelle : une extension temporelle associée à une trace peut être **:**

- Un intervalle temporel déterminé par deux dates : date de début et date de fin de l'observation, appelé extension temporelle à base d'intervalles ;
- Une séquence d'éléments à un instant donné, appelé extension temporelle à base d'instants.

Observé : Les observés sont des données relatives à une observation faite d'une activité [55].

En considérant les observés définis pour chaque activité effectuée par l'utilisateur, nous donnons une définition plus générale des traces dans un système informatique comme suit :

Trace : une trace est une information ou une séquence d'informations générées par toute action relative à un objet ou élément. Dans les systèmes informatiques, la trace est considérée comme la suite des observés [52].

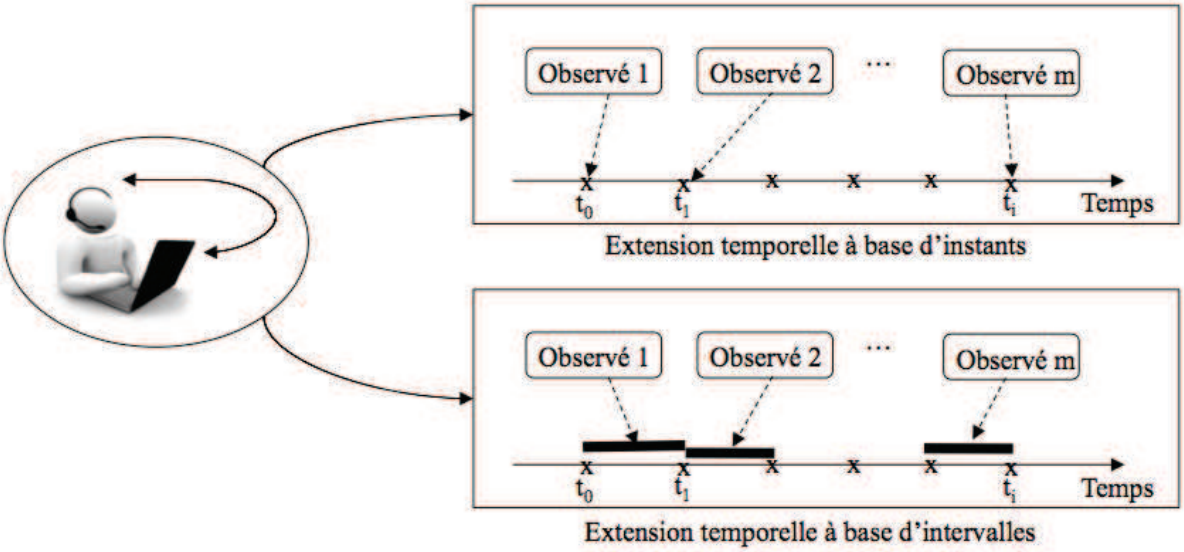

Figure 15. Trace, observé et extension temporelle

3.3.2 TRACE MODELISEE

Une trace numérique dans un système informatique respecte un modèle de trace, qui décrit les objets qui en font partie. Ce modèle pourra représenter de manière formelle et abstraite les éléments constitutifs d'une trace. Chaque trace peut être associée à un modèle de trace dont la définition est la suivante :

Modèle de trace : Un modèle de trace est un modèle représentant la trace d'une manière formelle. Il contient notamment les propriétés et attributs de la trace : date et heure, identifiant d'utilisateur, action réalisée, etc. Ce modèle doit être explicite et permet d'expliciter la trace. Un modèle de trace peut être implicite, c'est-à-dire uniquement présent implicitement dans le code de l'outil l'utilisant. L'objectif d'un modèle de trace est d'expliciter et de décrire abstraitement les objets qui font partie de la trace [56].

Par exemple dans le cas des fichiers de journalisation du serveur Apache, le modèle de trace est Common LogFile Format explicitant la sémantique de la trace.

Trace modélisée : Une trace modélisée ou m-trace est une traces qui est associée avec modèle de traces [52]. Une m-trace est toujours associée à un modèle de trace définissant les éléments qui la composent : observés et relations.

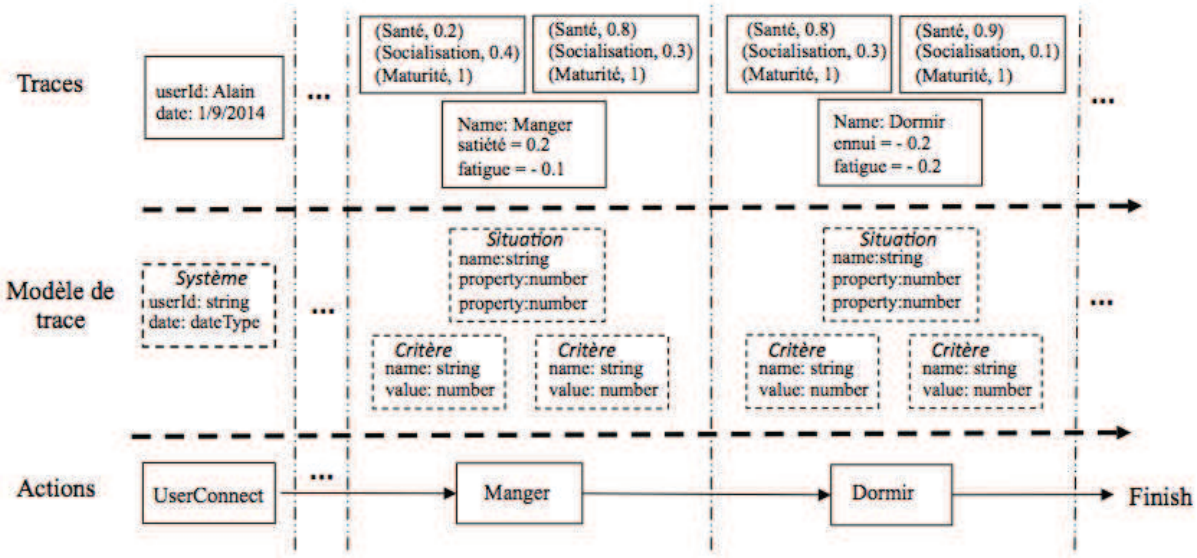

Figure 16. Trace modélisée (m-trace) dans le cas d'étude Tamagotchi

3.3.3 PRINCIPE D'UN SYSTEME A BASE DE TRACES MODELISEES

Un système à base de traces est un système informatique permettant et facilitant l'exploitation des traces [57]. Un système à base de traces modélisées est tout système dont le fonctionnement implique à des degrés divers la gestion, la transformation et la visualisation de traces modélisées explicitement [58].

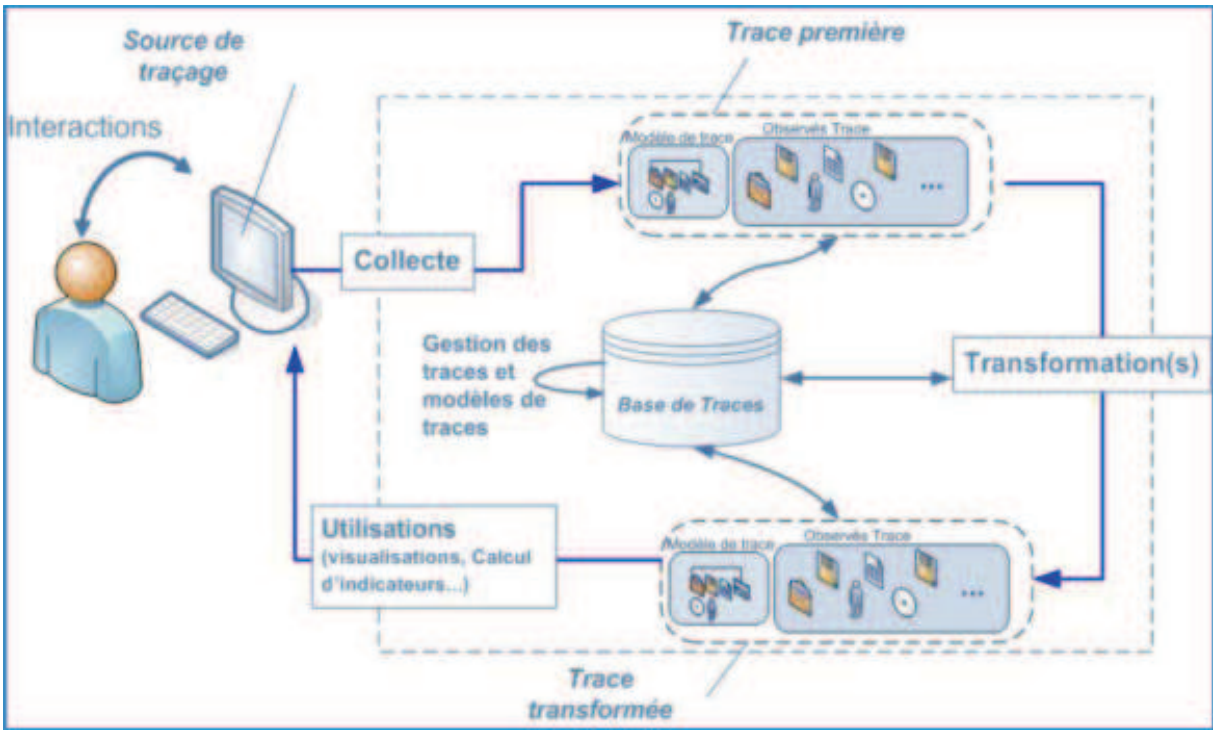

Figure 17. Principe général d'un système à base de traces modélisées [59]

La Figure 17 décrit le principe général d'un système à base de traces modélisées. Il existe plusieurs types des systèmes à base de traces. Chaque approche correspond à un système de traces différent. Malgré un grand nombre de SBT, chacun doit respecter un modèle de fonctionnement tel que montré dans la Figure 18. Ce modèle représente les composants qui constituent au fonctionnement d'un système à base de traces modélisées. Dans cette figure, une source de traçage est un objet dans l'application ayant des formats explicites quelconques (vidéo, fichiers, etc.). Une trace dans le SBT est la séquence temporelle d'observés associée à un modèle de trace. Nous allons donc décrire les différents composants d'un système à base de traces modélisées.

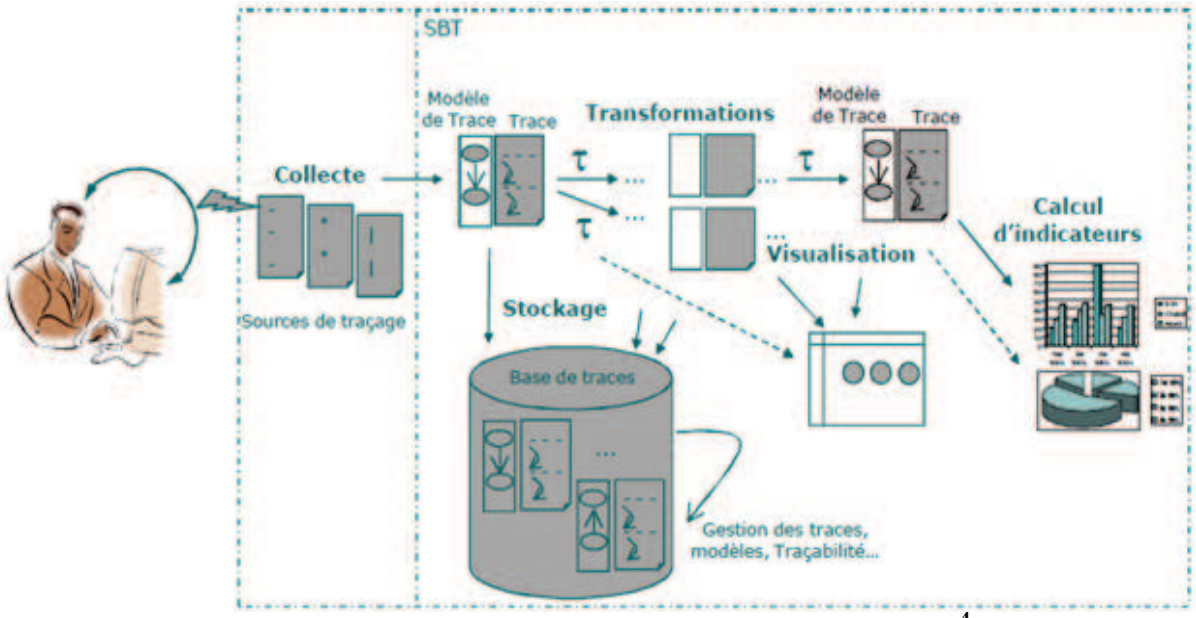

Figure 18. Fonctionnement du système à base de traces[4]

Un système à base de traces modélisées permet de :

- Définir les sources des traces ;
- Récupérer les traces ;
- Gérer des modèles (modèles de traces, de transformation, etc.) ;
- Mettre à jour la base de traces, conservation des traces ;
- Analyser les traces.

Nous retrouvons ces différentes missions en trois phases du système à base de traces. Elles sont :

- Collecte de traces :
  - La définition des sources de traçages dans une application interactive ;
  - La définition du modèle de traces ;
  - La récupération des traces.
- Transformation de traces :
  - La définition des modèles de transformation ;
  - La transformation des traces obtenues de la phase de collecte.
- Analyse de traces :
  - L'analyse des traces selon les buts d'utilisation.

Nous allons donc décrire ces trois phases en précisant les missions incluses.

3.3.3.1 Collecte de traces

La collecte des traces permet l'observation de l'utilisation d'un système à partir d'une ou de plusieurs sources de données, appelées *sources de traçage*. Elle sert à convertir ou à transformer des informations générées par l'interaction utilisateur/système pendant l'activité en une trace initiale, dite *trace première*.

[4] http://cluster-isle-eiah.liris.cnrs.fr/

Source de traçage : Une source de traçage est un fichier ou un flux de données dans un format explicite. Nous dirons que ce sont des ressources disponibles à tout moment dans le système à base de traces. Plusieurs sources de traçage peuvent être utilisées pour récupérer des traces [57].

Trace première : Une trace première est une trace collectée au fil de l'activité (trace brute) [60].

Un processus de collecte de traces modélisées consiste à exploiter de façon manuelle ou automatique un ensemble de sources de traçage pour construire une trace modélisée. Pour cela, nous avons besoin de construire un ensemble d'observés associés à un modèle de trace défini. Une trace modélisée issue du processus de collecte est appelée trace modélisée première ou m-trace première du système à base de traces modélisées.

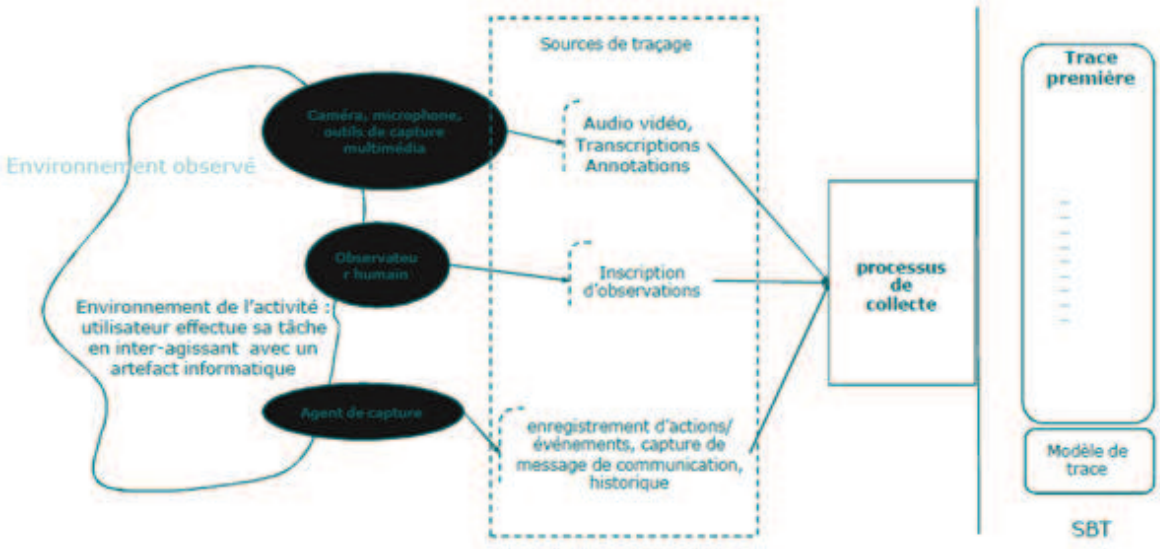

Figure 19. Collecte de traces[5]

Il y a deux techniques de traçage :

- Approche ad-hoc qui consiste à tracer tous les observables (l'ensemble des traces enregistrables) ce qui conduit généralement à une grande quantité de traces non directement interprétables par un utilisateur humain (par exemple, les traces collectées par un serveur web ou une application de type Keylogger) ;
- Approche par instrumentation de l'environnement de travail, qui consiste à collecter les traces par rapport à un objectif d'observation et qui permet de tracer un ensemble restreint d'observables jugé pertinent pendant la session d'observation.

3.3.3.2 Transformation de traces modélisées

Les traces premières sont collectées depuis les sources de traçage. Elles sont stockées dans une base qui s'appelle base de traces modélisées.

Base de traces modélisées : l'ensemble des traces modélisées ou les m-traces qui sont manipulées par le système à base de m-traces.

[5] http://cluster-isle-eiah.liris.cnrs.fr/

La trace obtenue à l'issue de la phase de collecte n'est pas toujours exploitable directement, il faut passer par une phase de transformation permettant d'effectuer une ou plusieurs transformations sur une trace en entrée et dont le résultat de transformation est une nouvelle trace appelée trace transformée.

Trace transformée : Une trace transformée est une trace obtenue après la transformation selon des intentions d'analyse particulière [60].

Elle sert à assurer la capacité d'interprétation des traces recueillies. Cela permet à l'utilisateur de comprendre les éléments obtenus et permet de faciliter l'analyse de ces éléments selon les besoins différents. Les m-traces premières sont les seules m-traces non transformées d'un SBT. Il s'agit de deux catégories de transformations qui sont prises en charges par le SBT : la transformation manuelle et la transformation automatique utilisant des modèles de transformation.

Une transformation manuelle désigne toute création directe par l'utilisateur d'une trace à partir d'une trace existante. Ce type de transformation est réalisé par un utilisateur interagissant avec sa trace pour la mettre à jour ainsi que son modèle. Le SBT permet de garder la cohérence entre les traces et leurs modèles lors d'une transformation manuelle.

La transformation automatique est employée dans un SBT en définissant les modèles de transformation. Un modèle de transformation est l'ensemble des règles permettant de sélectionner ou de réécrire des motifs (un motif représente la séquence ou un sous ensemble d'observés de la trace). Nous pouvons distinguer trois types de transformations automatiques comme la montre la Figure 20 :

- Type de sélection : c'est la création d'une nouvelle trace contenant tous les observés selon un filtre donné. Ce type de transformation permet de séparer les observés pertinents de la trace et ceux qui sont considérés comme du bruit. Les filtres sont des contraintes sur les objets de la trace, le domaine temporel et les relations structurelles ;
- Type de réécriture de motifs : permet de remplacer un motif d'observés en un autre observé. Ces motifs peuvent être constitués d'observés qui ne se suivent pas directement. La transformation a pour but de réécrire un motif représenté par une succession d'observés ;
- Type de fusion : selon les domaines temporels, elle consiste à fusionner les observés de plusieurs traces en une seule trace en respectant la temporalité. Le modèle de la trace fusionné assemble les modèles des traces fusionnés. Ce type de transformation est possible à appliquer si les traces sont des extensions temporelles homogènes (permettant des translations et des conversions de temps).

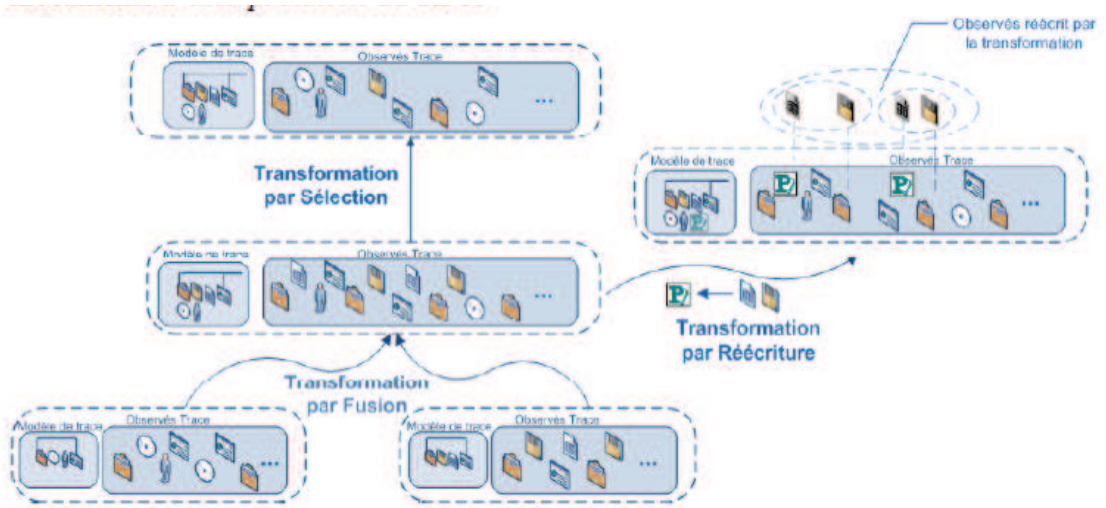

Figure 20. Transformation de traces [52]

3.3.3.3 Exploitation des traces

Les traces obtenues peuvent être visualisées telles quelles ou bien servir à l'élaboration d'indicateurs plus complexes.

Visualisation : Le système de visualisation est l'ensemble des techniques de présentation des traces. La visualisation nécessite quelques transformations :

- extraction et sélection des données à partir des sources de traçage ;
- transformations pour les mettre dans le format demandé ;
- transformations sous forme de primitives graphiques qui seront les briques pour la visualisation des données aux utilisateurs ;
- génération de la visualisation graphique à partir des primitives.

Calcul d'indicateurs : Un indicateur est une variable au sens mathématique à laquelle est attribuée une série de caractéristiques. Les valeurs de l'indicateur peuvent prendre des formes numériques, alphanumériques ou même graphiques. La valeur possède un statut, c'est-à-dire qu'elle peut être brute (sans unité définie), calibrée ou interprétée [61]. Les indicateurs facilitent les adaptations à réaliser dans l'environnement numérique. L'obtention des indicateurs n'est possible que grâce à une meilleure analyse des traces.

Les techniques de datamining représentent des approches qui peuvent être utilisées pour l'analyse des traces numériques collectées. Elles sont appropriées pour l'analyse des grandes quantités de traces numériques. Dans ce cas, l'analyse des traces présente trois parties principales : le prétraitement des traces, la découverte des patterns (ou motifs) d'utilisation intéressants et l'analyse de ces patterns. En effet, l'approche de Data Mining représente une alternative particulière et intéressante pour l'analyse des traces d'utilisation [56].

Le choix de technique utilisée pour mettre en œuvre cette phase dépend strictement des besoins et des applications particulières. La variété des systèmes de gestion de traces est basée sur les approches d'analyses. Nous allons présenter dans la section suivante, un système qui permet de gérer toutes les traces générées, appelé Système à base de traces.

La phase d'exploitation de traces nous permet de les utiliser comme une source d'informations abondantes. Nous allons décrire les utilisations déjà expérimentées des traces pour l'adaptation d'une application interactive. Ceci nous permettra de faire ressortir les besoins pour les Systèmes à Base de Traces dans une application interactive à base de situations.

3.3.4 UTILISATIONS DES TRACES

Les expériences constituent une source de connaissance qui pourrait aider à apprendre à partir des solutions qui ont fonctionné dans des contextes définis. En nous basant sur ces expériences, appelées expériences partagées, nous pouvons renforcer le transfert de compétences, améliorer les performances et faciliter l'extraction des connaissances à partir des cas vécus. Ces expériences sont les traces générées de plusieurs utilisateurs différents. Elles représentent les logs entre l'utilisateur et l'application dans laquelle les interactions sont effectuées. Le terme « utilisateur différent » désigne des utilisateurs de profils différents. Un profil d'utilisateur décrit l'ensemble des informations concernées à l'utilisateur. Elles pourront être les capacités, les compétences, les préférences, etc. L'objectif principal de cet utilisation des traces est de proposer des modèles et des outils permettant au système de transformer les traces partagées par un utilisateur source u1 pour qu'elles soient adaptées aux utilisateurs cibles u2 et u3 (ce sont trois utilisateurs différents) quels que soient leurs profils et besoins spécifiques [62].

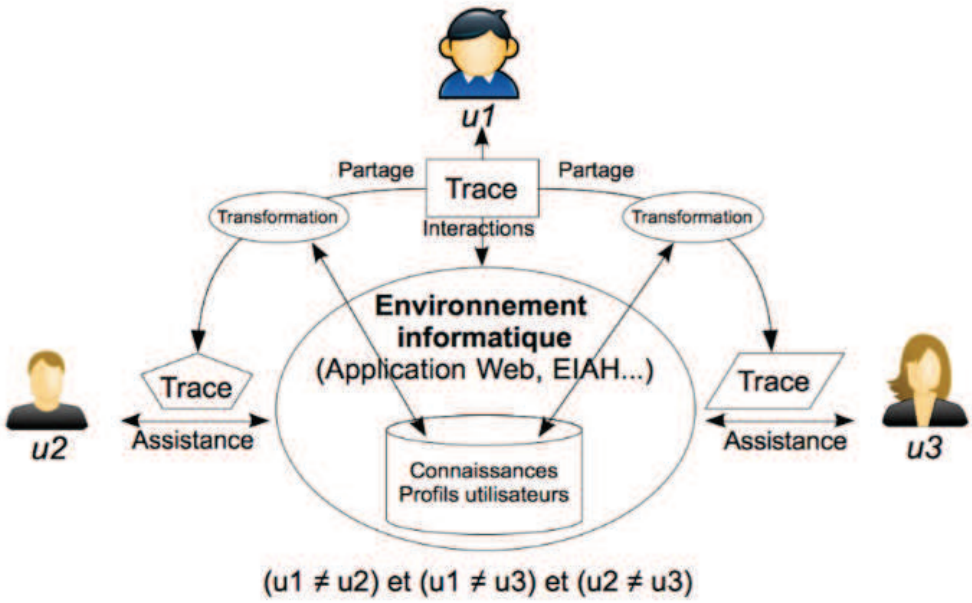

Figure 21. Partage d'expériences entre utilisateurs de profils différents [62]

Les domaines suivants sont ceux sur lesquels nos travaux peuvent porter: le contenu, la présentation et les modalités d'interaction [63] :

- L'adaptation du contenu : elle consiste à ajouter, supprimer ou modifier les éléments de la trace ;
- L'adaptation de présentation : elle concerne l'affichage des traces et ses propriétés. Elle choisit la méthode de visualisation la trace la plus convenable en fonction de ses propriétés [64] ou le profil de l'utilisateur ;

- L'adaptation des modalités d'interaction : elle concentre sur la façon de faire pour réaliser les tâches. La plupart des travaux se focalisent sur le troisième type d'adaptation des modalités d'interaction.

Plusieurs systèmes de partage d'expériences ont été développés. Le principe de ces systèmes est d'exploiter les expériences passées, les traces, afin d'identifier celles qui sont similaires au problème cible. Ces résultats de similarité sont utilisés pour apporter une assistance à un utilisateur dans la réalisation de sa tâche. Par exemple, un système de partage d'annotations [65] a été proposé pour des séances d'apprentissage collaboratif. [66] propose un système de recommandation dans le contexte d'environnement d'apprentissage. Ce système est basé sur le partage de traces qui est fait par le calcul du degré de confiance entre l'utilisateur source et cible. Le partage d'expérience est aussi utilisé comme un support permettant d'aider les personnes aveugles dans leurs navigations de l'Internet [67]. Le partage dans ce contexte est basé sur l'historique de navigation des personnes voyantes pour déduire des connaissances qui pourront aider des aveugles dans leurs tâches.

Il y a eu de nombreuses recherches sur le partage d'expériences mais il s'agit d'un inconvénient sur la modélisation des informations et elles ne prennent pas en compte les spécificités de l'utilisateur cible. Si nous ne connaissons pas l'utilisateur cible, le partage d'expérience est effectué pourront donner les résultats qui ne sont pas appropriés. Afin de surmonter cette lacune, [62] propose de transformer les expériences partagées sous forme de traces modélisées, afin de les adapter à l'utilisateur cible. Dans son travail, l'auteur tient en compte un modèle de représentation de profils d'utilisateurs, de traces, de connaissances ainsi que les modèles de génération et d'extraction de connaissances.

Par ailleurs, nous pouvons extraire de connaissances à partir des traces représentant les activités des utilisateurs très différents. Chacun d'utilisateur a donné des feedback après avoir exécuté l'application. Il s'agit d'une approche d'extraction interactive des connaissances d'adaptation à partir des feedbacks des utilisateurs.

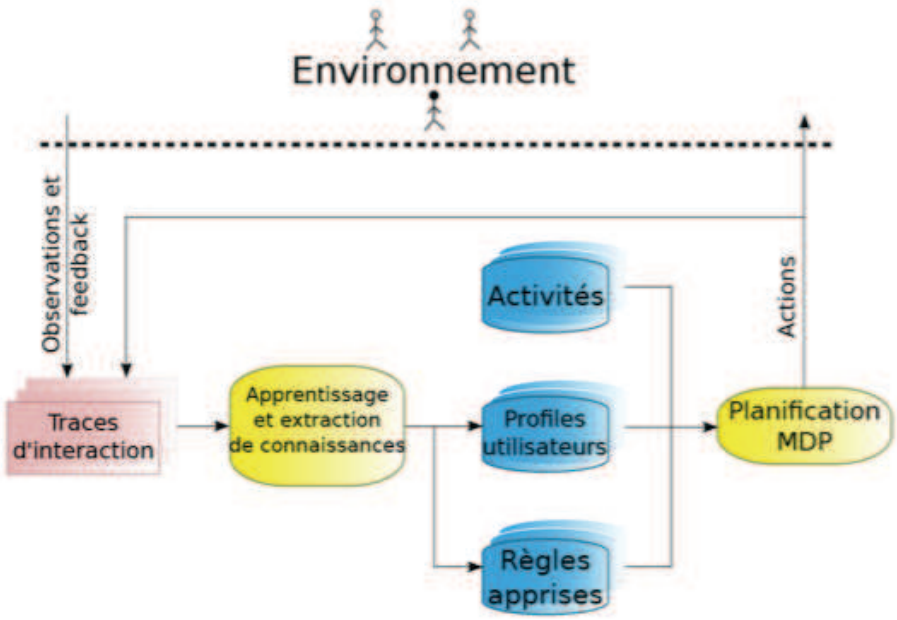

Figure 22. Architecture générale du système adaptatif: extraction de connaissances d'adaptation [68]

La Figure 22 présente l'architecture du système qui est composée de quatre modules (traces, activités, profils et règles d'adaptation) et deux moteurs de raisonnement permettant de planifier et extraire de connaissances. Les connaissances d'adaptation servent à prendre de décisions adaptées au contexte d'usage. L'objectif est de planifier le scénario qui maximise la satisfaction de l'utilisateur en considérant l'environnement spécifique [62]. Les connaissances d'adaptation expriment les généralités d'habitudes et de préférences des utilisateurs. Chaque connaissance est modélisée sous forme de règle indiquant la satisfaction des utilisateurs vis-à-vis d'une action du système dans un contexte particulier. Dans [62], l'auteur a proposé deux algorithmes d'extraction de règles à partir de traces. Le premier est direct mais il demande un nombre de traces pour effectuer le calcul. Le deuxième, plus rapide mais incertain car il consiste à généraliser les règles disponibles pour appliquer à des nouveaux contextes. Cette utilisation a montré des performances mais elle ne s'agit pas d'un algorithme permettant de sélectionner un ensemble d'activités adéquates. De plus, la modélisation de connaissances sous forme des règles est simple, elle ne permet pas de savoir le niveau de similarité entre deux contextes d'usages. Puisque chaque contexte à un moment de donné n'est pas tout à fait identique, ils pourront être un peu différent, différent, etc. Dans ce cas, c'est un peu difficile pour extraire les conclusions à partir des règles existantes.

Nous avons présenté l'utilisation des traces pour mettre en œuvre l'adaptation dans quelques contextes. Nous trouvons que les traces nous apportent une grande source de connaissances qui sont très richesses et potentielles à fouiller. Par conséquent, pour gérer un grand nombre de traces comme ça, il nous faut un système à base de traces qui permet de les gérer automatiquement. Nous aimerons d'aborder dans la section suivante les systèmes à base de traces existants ou des méthodes/ des outils qui traitent plus ou moins la gestion de traces.

3.4 SYSTEMES A BASE DE TRACES EXISTANTS

Un système à base de traces est un tel système permettant de gérer les traces depuis la collecte jusqu'à la réutilisation des traces dans différentes applications. La popularité de l'utilisation des traces est de temps en temps augmente. Nous allons voir quels sont les outils, les approches existants qui fonctionnent en tant qu'un système à base de traces ou spécifiquement les SBT existants pour les positionner dans notre contexte de la thèse : application interactive à base de situations.

3.4.1 COIAT (ANALYSE)

ColAT [69] est un outil d'analyse de traces d'apprentissage indépendant de tout système d'apprentissage. La nouvelle version de ColAT, ActivityLens, est un environnement développé par Human - Computer Interaction Research Group (HCI Group) of Electrical & Computer Engineering Department, de l'université de Patras pour soutenir la recherche ethnographique et pour faciliter l'analyse des données produites à partir des études d'évaluation d'utilisabilité.

ActivityLens est un outil d'analyse basé sur l'activité, spécialement conçu pour les chercheurs et les analystes qui sont impliqués dans les études ethnographiques. Son but est de les aider dans leur analyse et permet d'étudier les données provenant de sources multiples recueillies au cours de ces études. L'utilisation intensive du multimédia permet aux chercheurs de travailler avec des fichiers textes, graphiques, sons, vidéos, mais aussi des fichiers de log qui sont produits à partir de différentes études. Il offre une vaste gamme de fonctionnalités pour

l'observation et l'analyse des données recueillies. Ainsi, ActivityLens permet une manipulation facile et une navigation dans les sources multimédias, l'annotation des données, l'analyse statistique, le filtrage des données, la visualisation de données annotées, la génération de rapports, etc.

3.4.2 APLUSIX

APLUSIX [70] est un environnement d'aide à l'apprentissage de l'algèbre qui utilise la trace d'apprentissage pour aider les élèves à résoudre des exercices de calcul numérique[6]. C'est un logiciel innovant, développé au Laboratoire d'Informatique de Grenoble, pour aider les élèves à apprendre l'arithmétique et l'algèbre. Il permet de renforcer les compétences des élèves, en diminuant leurs erreurs de calcul, et montre comment résoudre les exercices aux élèves qui en ont besoin.

L'Aplusix permet d'enregistrer de productions d'élèves, comme celles obtenues dans l'environnement classique papier, mais comportant de plus d'autres informations, comme le temps, les hésitations, les corrections. Ce travail produit une trace brute qui représente une modélisation comportementale des élèves [71]. Ce logiciel procède à une restriction, un découpage et une interprétation des traces brutes obtenues lors des anciennes manipulations des élèves et produit une trace enrichie contenant des règles algébriques expliquant les transformations des élèves et identifiant le contexte où elles apparaissent. Par ailleurs, Aplusix présente une modélisation des connaissances des élèves qui s'est appuyée sur la confrontation des analyses manuelles et automatiques d'expérimentions ayant eu lieu dans des établissements scolaires de différents pays [71].

3.4.3 MUSETTE

MUSETTE (Modeling USEs and Tasks for Tracing Experience) [72] est une approche générale de modélisation, et peut être implémentée grâce à divers langages ou formalismes de représentation. C'est un outil permettant et facilitant la manipulation des traces.

Dans le cadre des recherches sur le traçage des interactions, le modèle MUSETTE a été développé dans le but de modéliser l'utilisation d'un système par un (ou des) utilisateur(s) pour pouvoir tracer, dans un objectif de réutilisation des traces. Elle est une approche permettant de modéliser l'expérience de l'utilisateur dans l'optique de réutiliser cette expérience comme une source de connaissances. Cette approche est basée sur le traçage des interactions, conformément à un modèle d'utilisation du système qui décrit les objets et relations manipulés par l'utilisateur. Les traces peuvent être visualisées, intégrées à une base de connaissances non-orientée tâche, et peuvent être ensuite analysées grâce à des signatures de tâches. MUSETTE vise donc à représenter l'expérience concrète, en lien avec son contexte d'utilisation.

La mise en œuvre de l'approche MUSETTE est développée et diversifiée dans les travaux de l'équipe CEXAS du LIRIS. Certains travaux s'attachent par exemple à étudier les possibilités de MUSETTE appliquée à de multiples utilisateurs et au partage et à la réutilisation d'ontologies (approche multi-agents de MUSETTE, modèle MAZETTE) [73]. D'autres approches, s'appuyant également sur l'approche MUSETTE, visent l'étude des processus

[6] http://www.aplusix.com/

métacognitifs dans le cadre de l'apprentissage (projet Clever@ [74]), ou l'assistance à la réutilisation de l'expérience dans un contexte coopératif de conception. Enfin, une approche dénommée « MUSETTE-Analyse » vise non plus à assister directement l'utilisateur, mais à fournir des outils à des analystes d'activités tels que la conduite automobile (projet INRETS), ou bien la manipulation d'outils informatiques pour les personnes âgées (projet MNESIS).

3.4.4 TATIANA

TATIANA (Trace Analysis Tool for Interaction ANAlysis) [75] est un système qui sert aux chercheurs pour analyser des traces d'interactions. C'est un logiciel orienté vers l'analyse de séances de travail collaboratif. C'est un environnement logiciel basé sur ce modèle et conçu pour aider les chercheurs à gérer, synchroniser, visualiser et analyser leurs données au travers de la création itérative d'artefacts qui leur permettent d'améliorer ou de concrétiser la compréhension qu'ils ont de leurs données.

La Figure 23 présente le schéma fonctionnel de TATIANA.

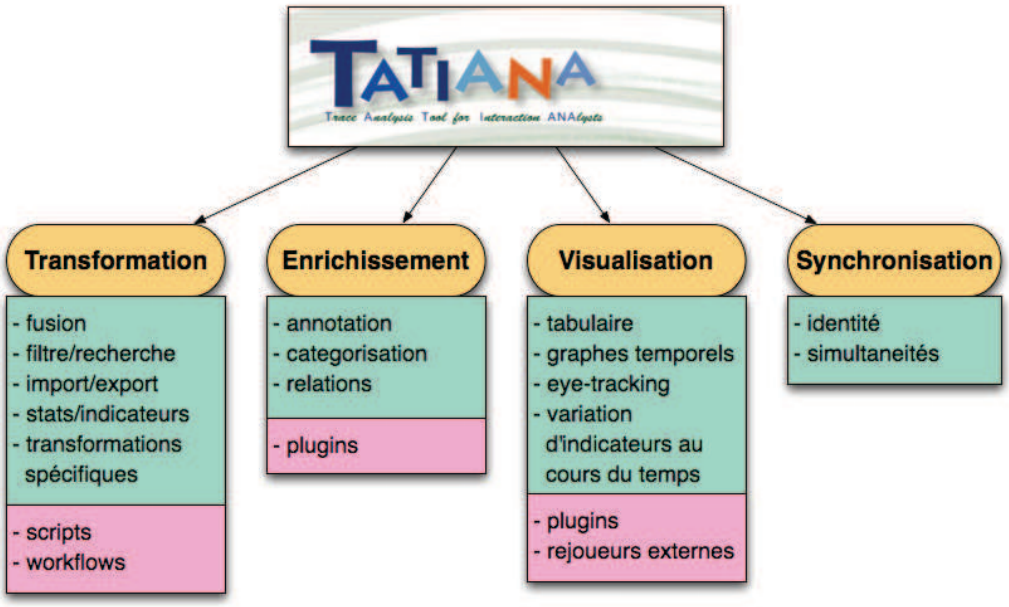

Figure 23. Schéma fonctionnel TATIANA[75]

3.4.5 VISU

VISU [64] est une plateforme de visioconférence à base de traces pour le tutorat synchrone. VISU intègre directement dans ses interfaces les éléments de visualisation et de manipulation de traces. Il s'appuie fortement sur les traces modélisées pour représenter et enrichir les interactions homme-machine. VISU est un logiciel issu du projet ANR Ithaca[7]. Il permet aux tuteurs de préparer une séance, de la mener tout en posant des marqueurs. Il permet aux tuteurs et aux apprenants de revenir sur celle-ci a posteriori et d'en construire des bilans.

[7] https://liris.cnrs.fr/~ithaca/

VISU a été conçu, dans un premier temps, pour répondre à des besoins spécifiques de l'enseignement-apprentissage des langues en ligne. Par rapport à l'existant, VISU permet aux enseignants de disposer de trois fonctionnalités principales pendant la séance qui sont :

- un assistant de préparation qui permet de rentrer en amont de la séance des consignes, des mots-clés, des documents iconographiques ou vidéo, afin d'organiser l'activité pédagogique et rendre l'interaction plus facile pour les enseignants qui n'ont plus qu'à cliquer sur ces différents items pour les rendre visibles à leurs étudiants distants ;
- une zone de communication centrale qui concentre toutes les interactions, écrites, visuelles et orales et facilite le travail d'apprentissage en ligne ;
- une zone de suivi qui permet aux enseignants (et aux étudiants) de poser des marqueurs pour manger le feedback apporté aux étudiants (précisions, corrections, etc.) sans gêner le cours de l'interaction orale qui reste la priorité principale.

3.4.6 SBT_IM

SBT-IM [76] est un système à base de traces pour le calcul d'indicateur dans Moodle. Il a été développé par le laboratoire LIRIS. SBT-IM est composé d'un système de collecte de traces, un système de transformation des traces, un système de visualisation et un système de calcul d'indicateurs à base des traces.

C'est un système à base de traces qui a été construit pour utiliser dans Moodle. Il respecte bien le principe d'un SBT présenté dans la section 3.3.3.

3.4.7 SYNTHESE DES SYSTEMES A BASE DE TRACES

Nous allons décrire maintenant la synthèse des six outils/approches/logiciels à base de traces présentés ci-dessus en vérifiant avec nos besoins d'un système à base de traces dans une application interactive à base de situations.

Dans le contexte de la thèse, nous avons besoin d'un système à base de traces qui doit :

- Définir les sources de traçage ainsi que la collecte de traces ;
- Transformer les traces obtenues selon les besoins particuliers ;
- Analyser les traces transformées dans un objectif défini ;
- Fonctionner dans une application interactive à base de situations.

Le Tableau 6 présente une synthèse des SBT existants.

Tableau 6. Synthèse et Analyse des SBT existants

	Collecte	Transformation	Analyse	Adapté dans le contexte de situation	Domaine appliqué
COIAT	Collecte multiples sources	oui	oui	non	Recherche ethnographique
APLUSIX	oui	oui	oui	non	Education (aide à

					l'apprentissage de l'algèbre)
MUSETTE	oui	oui	oui	non	Approche générale, pouvoir développer et diversifier
TATIANA	oui	oui	oui	non	Chercheurs dans une séance de travail collaboratif
VISU	oui	oui	oui	non	Education (répondre aux besoins de l'enseignement-apprentissage des langues en ligne)
SBT-IM	oui	oui	oui	non	Moodle

Nous voyons que les SBT existants présentent bien le principe du système à base de traces, mais nous ne pouvons pas les réutiliser dans le cas d'une application interactive à base de situations car il ne s'agit pas des observés de situations dans la phase de collecte. Évidemment, c'est difficile de trouver un SBT qui corresponde bien à nos besoins mais la plupart des SBT existants supportent la même architecture générale. Par conséquent, nous allons réutiliser les principes existants d'un SBT pour l'adapter dans notre contexte, qui est le système à base de traces dans une application interactive à base de situations.

3.5 CONTRIBUTION 1 : SYSTEME A BASE DE TRACES DANS UNE APPLICATION A BASE DE SITUATIONS

Nous présentons dans cette section notre première contribution concernant le système à base de traces dans une application interactive à base de situations. La contribution ne concernant pas l'élaboration de nouveaux concepts autour des SBT. Elle réside dans l'adaptation des concepts généraux des SBT aux systèmes à base de situations. Cette section est la mise en place des principes fondamentaux des systèmes à base de traces pour construire un SBT en conformité avec notre contexte. Notre travail sur les traces n'a pas donné un apport particulier dans le domaine de gestion de traces mais il nous permet de collecter et de stocker dans une base de traces formalisées des données liées à l'exécution à base de situations. Cette base de traces est une condition préalable pour pouvoir entrer dans les autres contributions de la thèse concernant le processus d'aide à la décision à base de traces.

Nous réutilisons le principe général du SBT présenté dans la section 3.3.3 en adaptant dans le contexte de la thèse.

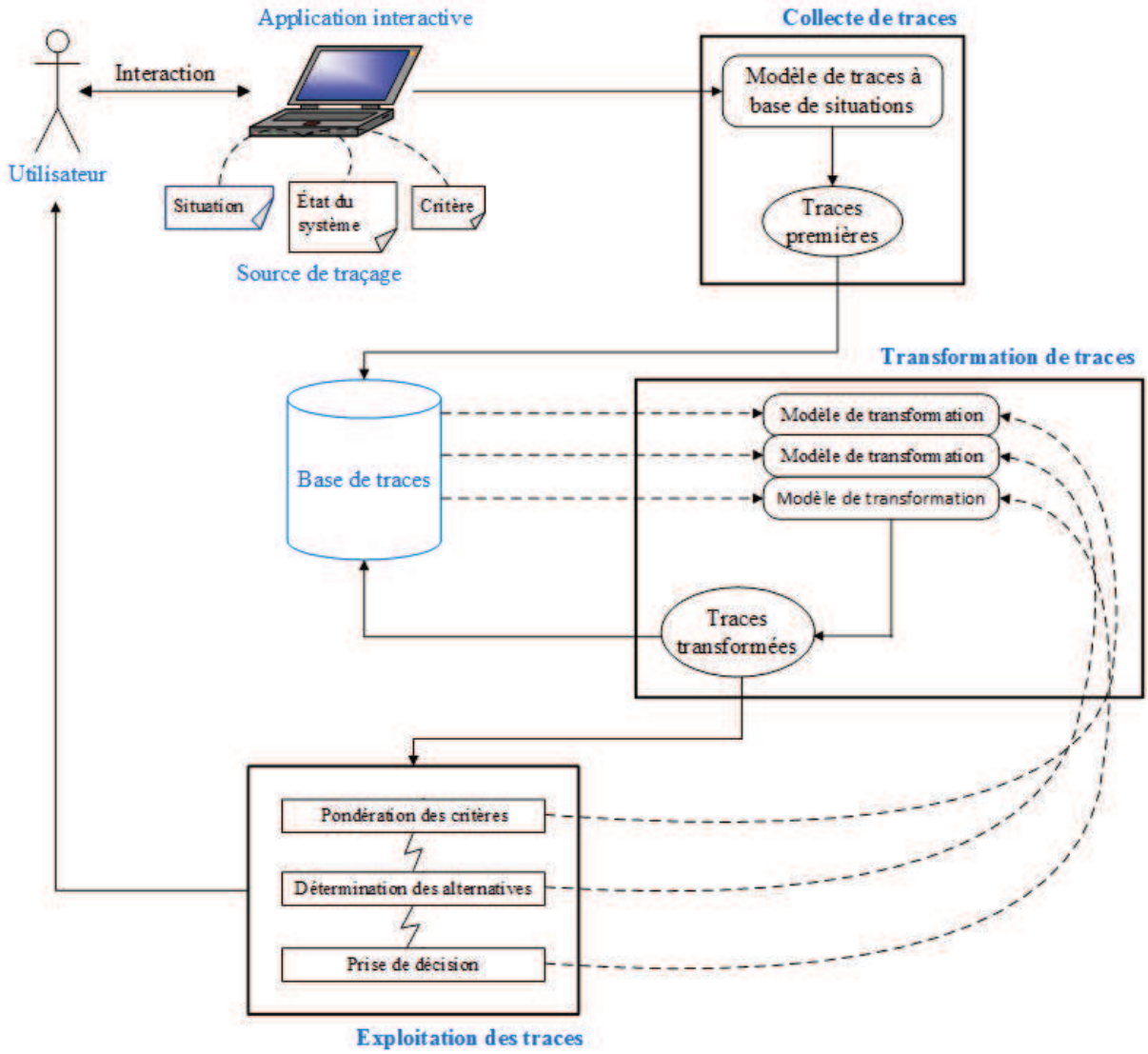

Figure 24. Architecture du système à base de traces dans une application interactive

La Figure 24 présente l'architecture de notre SBT dans une application interactive à base situations. Nous avons un utilisateur qui interagit avec l'application interactive par la réalisation des interactions. Lors de l'exécution, le système à base de traces fonctionne en respectant les trois sous-systèmes suivants :

- Collecte de traces ;
- Transformation de traces ;
- Exploitation de traces.

Nous allons décrire en détail chaque sous-système ci-dessus dans la suite de cette section.

3.5.1 COLLECTE DE TRACES

La collecte de traces dans notre SBT prend en charge la récupération des traces et les conserve dans la base de traces. Elle doit comprendre les fonctionnalités suivantes :

Définition des sources de traçage :

L'utilisateur connecte à l'application pour la faire fonctionner. L'exécution de cette application est un enchaînement des situations, chaque situation est exécutée à un moment de l'exécution. Pour savoir ce qui s'est passé au cours du déroulement, nous avons besoin d'observer les situations exécutées en regardant ses interactions effectuées. C'est pour cela que nous mettons en œuvre « situation » comme une des sources de traçage.

A un moment lors de l'exécution, nous disposons une autre source de traçage qui s'appelle « vecteur d'état » qui permet d'observer l'ensemble des états du système. Ce vecteur d'état contient tous les états contribuant au fonctionnement de l'application tels que : l'état de l'utilisateur, l'état de ressources, etc.

Quand l'utilisateur utilise l'application, il veut atteindre un objectif défini par le concepteur. Cet objectif est évalué par plusieurs critères. Chaque critère est défini dans la section 2.2.2.1. Nous ajoutons aussi à l'ensemble des sources de traçage une nouvelle source qui est « critère ». Elle sert à observer l'état ainsi que les valeurs associées à chaque critère pendant l'exécution de l'application.

Définition du modèle de traces à base de situations :

En nous basant sur les trois sources de traçage ci-dessus, nous définissons trois types d'observés associés pour décrire les traces :

- l'observé du système ;
- l'observé de situation ;
- l'observé de critère.

La Figure 25 décrit formellement les trois observés identifiés dans notre SBT.

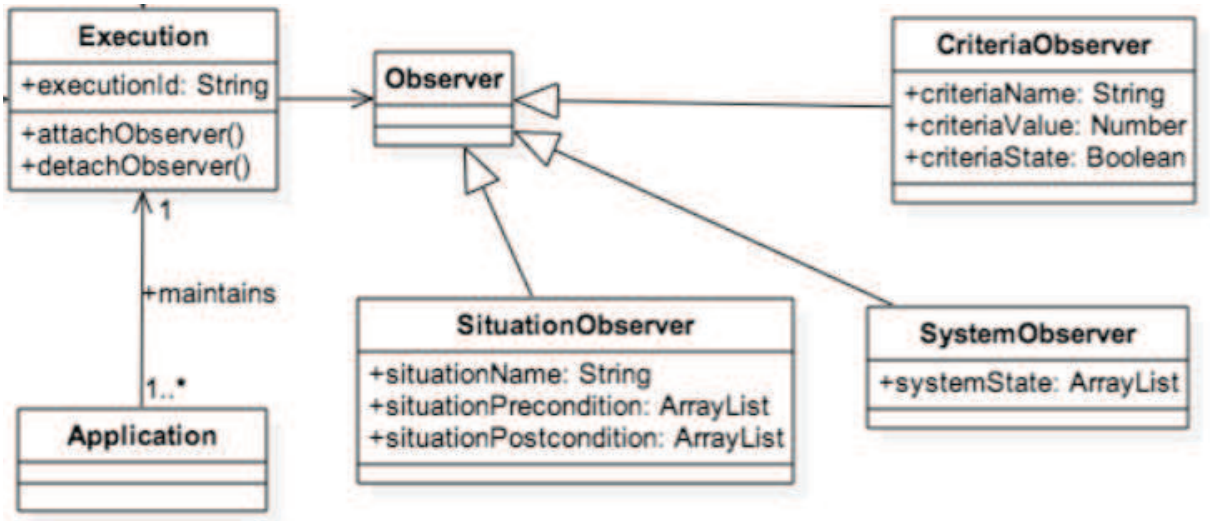

Figure 25. Trois observés dans notre SBT

Notre SBT collecte les données issues des interactions et les représentent dans des traces premières. Chaque trace première est associée par un modèle de traces permettant de recueillir les informations selon les trois observés définis. Ce modèle est noté par T_P, qui se compose par 3 éléments :

- U décrit le profil de l'utilisateur ;
- $\{V, S, \Delta\}_t$ décrit l'exécution à un moment t dans laquelle :
    - V : le vecteur d'état du système, il rassemble tous les états contribuant à l'exécution de l'application ;
    - S : la situation exécutée ;
- Δ : l'ensemble des valeurs des critères.
- Δ_{fin} est l'ensemble des états des critères.

Lors de l'exécution de l'application, nous récupérons les informations selon ce modèle T_P, nous obtenons les traces premières. Ces traces sont conservées dans la base de traces pour être utilisée dans les différents sous-systèmes.

3.5.2 TRANSFORMATION DE TRACES

Les traces récupérées dans la base obtenue sont les informations brutes. Nous avons besoin de les transformer pour extraire les données pertinentes avec nos besoins particuliers. Pour chaque besoin, nous définissons un modèle de transformation qui sélectionne depuis T_P, les éléments qui seront utilisés dans le sous-système d'exploitation de traces. Le résultat de la transformation est la trace transformée, notée T_T.

3.5.3 EXPLOITATION DE TRACES

Dans notre SBT, la prise de décision sera abordée dans le sous-système d'exploitation de traces. Selon nos besoins identifiés dans la section 2.4, notre processus de prise de décision multicritère contient de 3 étapes : la pondération des critères, la détermination des alternatives et l'algorithme de décision multicritère.

Pour chaque étape dans ce processus, nous définissons un modèle de transformation pour extraire les données qui permet de mettre en œuvre chaque étape. La description des informations extraites et le détail des calculs seront présentés dans le Chapitre 4.

3.6 SYNTHESE

Abondantes, les traces sont porteuses d'informations utiles pour l'analyse et l'adaptation de la logique d'exécution selon un contexte particulier. C'est pourquoi, nous proposons d'affiner le choix par l'analyse des traces relevées sur le système en fonctionnement afin d'en extraire les expériences des anciens utilisateurs ou de connaître la tendance des exécutions antérieures. Les traces vont ainsi permettre au système de faire un choix plus optimisé, plus affiné. Ce système s'appelle système d'aide à la décision multicritère à base de traces.

CHAPITRE 4 AIDE A LA DECISION MULTICRITERE A BASE DE TRACES

SOMMAIRE

Dans ce chapitre, nous décrivons étape par étape notre processus d'aide à la décision multicritère à base de traces. La question à laquelle nous tenterons de répondre est : comment prendre une décision multicritère en nous basant sur les traces générées pendant l'exécution d'une application interactive ?

La pondération des critères joue un rôle très important dans le mécanisme de décision multicritère. Nous montrons alors comment associer un poids à un critère parmi un ensemble de critères définis. Ensuite, nous nous intéressons à la phase d'identification des alternatives en proposant une méthode qui permet de déterminer quelles sont les alternatives possibles afin de prendre la décision. Finalement, dans la section 2.2.2.3.3, nous adaptons PROMETHEE II en intégrant les traces en vue de mieux prendre en compte le contexte d'exécution de notre application. Notre méthode de décision ne tient pas seulement compte de l'algorithme de décision mais aussi des différents choix possibles tels que : celui de l'utilisateur, du concepteur et du système. Ainsi, nous combinons ces différents points de vue par l'application d'une méthode d'agrégation pour proposer le meilleur choix à l'utilisateur.

4.1 PROCESSUS D'AIDE A LA DECISION MULTICRITERE A BASE DE TRACES

Avant de décrire en détail la décision multicritère en se basant sur les traces, nous commençons par présenter le processus par lequel la décision multicritère est faite. Notre processus de décision multicritère dépend strictement des traces récupérées auprès du système à base de traces (SBT) présenté dans le Chapitre 3.

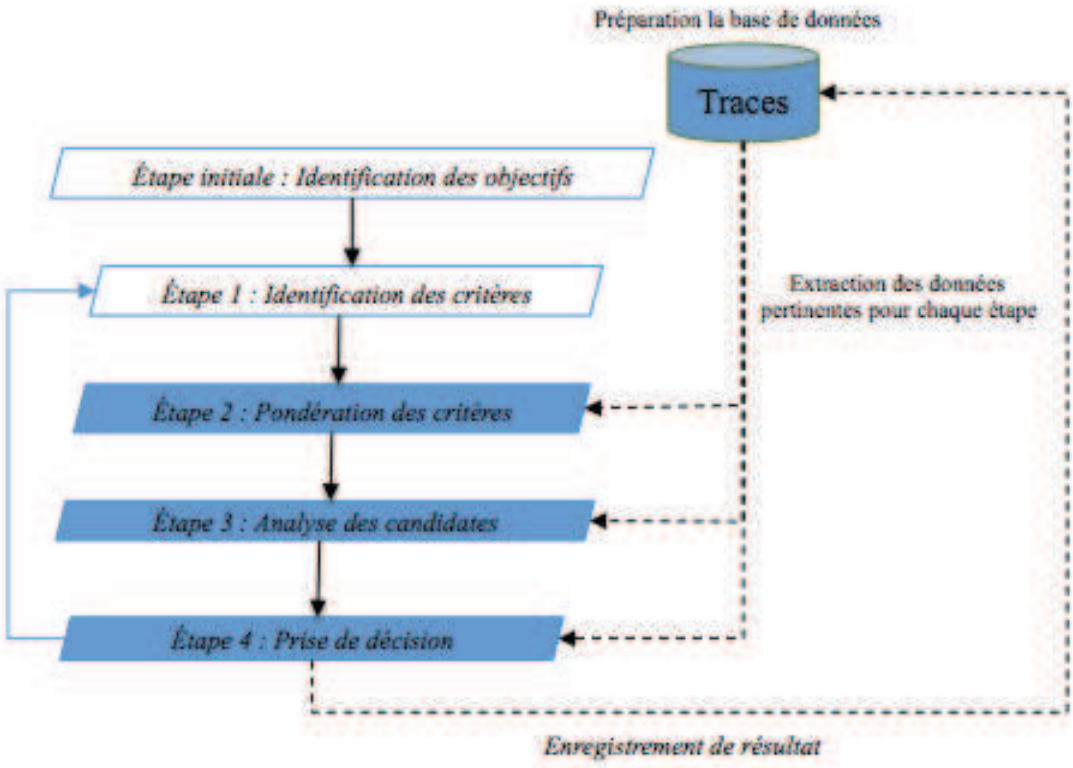

Figure 26. Processus d'aide à la décision multicritère à base de traces

Comme le montre la Figure 26, le processus d'aide à la décision multicritère se compose de cinq étapes et trois actions :

- Étape initiale : l'identification des objectif(s) : il est important de bien définir l'objectif de l'application car l'utilisateur doit savoir ce qu'il doit atteindre ;
- Étape 1 : l'identification des critères : pour évaluer un ou plusieurs objectifs, il nous faut disposer des critères qui nous guident dans l'accomplissement de l'objectif ;
- Action 1 : la préparation de la base de données : la base de données mentionnée dans cette partie est la base de traces ;
- Action 2 : l'extraction des données pertinentes ;
- Étape 2 : la pondération des critères : avant de pouvoir appliquer les méthodes de prise de décision, il faut fournir des pondération afin de quantifier chaque critère. En effet, les différents critères dans une application ne sont pas classifiés au même niveau et n'ont pas le même poids. Chaque critère a un niveau d'importance différent. Nous proposons une approche pour résoudre ce problème dans la section 4.2;
- Étape 3 : l'analyse des situations candidates : c'est une étape nécessaire dans la prise de décision car toutes les méthodes de décision ont besoin d'un ensemble de candidats disponibles pour calculer le choix final ;
- Étape 4 : la prise de décision : cette phase est le cœur du processus, nous nous intéressons aux différentes techniques permettant de proposer des solutions aux utilisateurs ;
- Action 3 : la mise à jour du résultat : cette partie apparaît à la fin du processus de décision multicritère à base de traces. Elle sert à observer le résultat de la décision qui a été prise par l'utilisateur afin de mettre à jour dans la base de traces. Cette information sera utilisée pour les prochaines décisions de l'application.

Pour prendre une décision, il faut suivre les étapes et les actions décrites ci-dessus. Seule l'étape initiale d'identification des objectifs est effectuée une seule fois, lors de la conception de l'application. L'utilisateur n'a pas besoin d'en tenir compte au cours de l'exécution de l'application. Par ailleurs, l'étape 1, l'identification des critères, est faite après la phase initiale lors de la conception de l'application mais elle doit être rebouclée afin de réexaminer l'ensemble des critères parcourus à chaque phase de décision.

4.1.1 IDENTIFICATION DE L'OBJECTIF

Selon [77], un objectif doit comporter plusieurs caractéristiques dont six peuvent être regroupées sous l'acronyme SMARTE (**S**pécifique, **M**esurable, **A**pplicable, **R**éaliste, **T**emporel, **E**cologique). Dans nos travaux, nous retiendrons les 5 premières caractéristiques (SMART). Celles-ci permettront à l'utilisateur de mieux comprendre et de faciliter son interprétation du résultat attendu.

- Spécifique : l'énoncé d'un objectif doit être formulé de manière spécifique et sans ambiguïté. Il est aussi important d'indiquer la formulation d'un objectif sans négation. Un bon objectif s'exprimera ainsi « il doit… » ou « il devrait… » ;
- Mesurable : un objectif est mesurable parce que cela permet au concepteur ou au système de calculer ou d'estimer le taux d'accomplissement de l'objectif ;
- Applicable : l'objectif d'une application doit être rapporté au contexte de l'application. Il est une partie relative à l'application et ne peut donc pas dépasser l'étendue de l'application ;

- Réaliste : un objectif doit être atteignable. Un objectif atteignable permet de définir les moyens nécessaires à sa réalisation ;
- Temporel : le meilleur objectif est celui qui satisfait les contraintes temporelles.

4.1.2 IDENTIFICATION DE CRITERES

Selon le dictionnaire Larousse, un critère est « un principe, élément de référence qui permet de juger, d'estimer, de définir quelque chose ». Dans notre cas, les critères sont des références qui servent à évaluer l'accomplissement des objectifs de l'application. Pour évaluer un objectif, il faut se baser sur un ou plusieurs critères. Un critère est déterminé par un type de valeur en lien avec l'objectif final.

Un critère peut être continu ou discret. Le mode d'évaluation d'un critère peut se formaliser de la façon suivante : nous avons l'ensemble $C = \{C_1, C_2, \dots, C_m\}$ de m critères. Pour chaque critère C_h, une fonction f_h est définie pour évaluer l'accomplissement de C_h pour l'objectif O de l'application.

$$f_h : C_h \longmapsto O \tag{30}$$

4.1.3 PREPARATION DE DONNEES

Les données de la base dans le processus de décision multicritère contiennent deux types d'informations. L'une est le profil de l'utilisateur, l'autre est les traces d'activités enregistrées lors des exécutions antérieures.

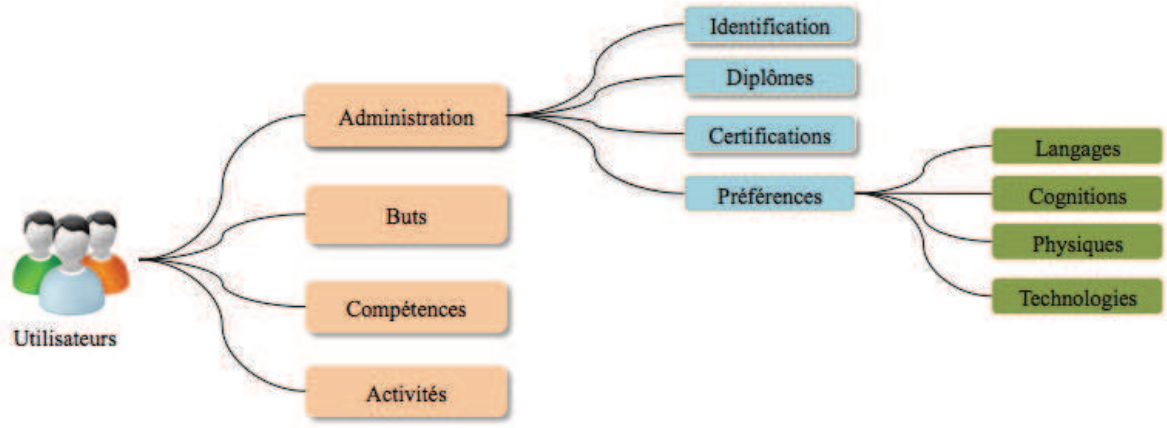

Figure 27. Modélisation d'un utilisateur par différentes catégories

Nous avons besoin de conserver toutes les informations concernant les utilisateurs ainsi que les traces qu'ils ont générées. Nous détaillons dans cette partie comment structurer les profils d'utilisateurs et leurs traces d'activités. La Figure 27 rassemble le profil de l'utilisateur et les traces associées à l'utilisateur.

Concernant la structuration du profil de l'utilisateur, les informations sont divisées en 3 catégories : Administratif, Buts et Compétences [78]. Ce sont les trois catégories basiques d'un

profil utilisateur. Il ne contient pas les historiques de l'utilisateur. Cette information nous permet d'extraire des connaissances utilisables. C'est pour cela que nous rajoutons une nouvelle catégorie appelée Activités pour conserver les traces de l'utilisateur en appliquant le principe dans le Chapitre 3. La construction de profil d'utilisateur est basée sur le principe de la thèse de Sawadogo [78].

- Administration :
    - Identification : des informations personnelles de l'utilisateur (nom et prénom, date de naissance, sexe) ;
    - Diplômes ou certifications : tous les diplômes ou certifications de l'utilisateur ; chacun se contient d'intitulé, de mention, du classement et des notes moyennes (si elles existent) ;
    - Préférences et non préférences : ce type d'information indique les domaines préférés et non-préférés de l'utilisateur :
        - Langages (anglais, français, espagnol,…) ;
        - Cognitions : l'utilisateur a une bonne cognition sur quel type d'application. Ou l'habitude par lequel l'utilisateur intéresse ;
        - Physiques : l'utilisateur aime des applications ayant besoin de mouvements ou les applications interactives ayant besoin de gestes,…
        - Technologies : de quelle type de technologie préférée par l'utilisateur ?
- Buts : les souhaits que l'utilisateur veut atteindre dans l'avenir.
- Compétences : ce sont des capacités reconnues dans un domaine de l'utilisateur. Elles peuvent être des savoirs personnels ou des compétences expertisées.

En ce qui concerne les traces d'interaction de l'utilisateur, nous appliquons le principe présenté dans la section 3.5 pour récupérer, stocker et transformer les actions réalisées lors de l'interaction entre l'utilisateur et le système. Dans la Figure 27, la partie « Traces » est représentée sous la catégorie « Activités ».

4.1.4 EXTRACTION DE DONNEES PERTINENTES

La phase d'extraction de données doit permettre d'identifier les informations les plus pertinentes. L'idée est alors de constituer une base de traces générées par chaque occurrence d'utilisation de l'application par l'utilisateur. Cette action est effectuée dans la phase de transformation intégrant dans le système à base de traces présenté dans la section 3.5.

Concernant les étapes restantes dans le processus de décision multicritère, nous les décrirons en détail dans les sections suivantes car elles font parties des contributions principales de la thèse :

- Contribution 2 : Pondération des critères (section 4.2)
- Contribution 3 : Sélection des situations candidates (section 4.3)
- Contribution 4 : Prise de décision (section 4.4)

4.2 CONTRIBUTION 2 : METHODE DE PONDERATION DES CRITERES

Dans cette section, nous présenterons une approche dans laquelle le système pondère un ensemble de critères identifiés. Pour que les algorithmes de décision puissent fonctionner

correctement, il faut d'abord prendre en compte l'initialisation des algorithmes de décision. Dans cette phase d'initialisation, nous devons prendre en compte la pondération des critères qui permettra de classer les critères par priorité. Si les critères ne sont pas hiérarchisés, les algorithmes de décision vont considérer qu'ils ont le même niveau d'importance. Cela peut influencer les valeurs des critères et l'objectif final. La pondération des critères va donc être une phase très importante dans la décision multicritère.

Afin de mettre en œuvre la pondération des critères, nous avons besoin des données pour effectuer le calcul. La pondération des critères sera effectuée lors de la première connexion de l'utilisateur. Le système devra récupérer les informations de l'utilisateur pour savoir quels sont les critères à mettre en priorité. Ces informations permettent au système de connaître les compétences de l'utilisateur, grâce auxquelles le système peut le guider à satisfaire les critères les plus efficaces. Le calcul sera réalisé en se basant sur une base de données contenant des profils et les traces d'interactions des utilisateurs lors des exécutions antérieures. La section 4.2.1 va présenter la base de données utilisée pour la pondération des critères.

4.2.1 BASE DE DONNEES UTILISEE POUR LA PONDERATION DES CRITERES

Depuis la base de profils des utilisateurs de la section 4.1.3, nous extrayons certaines informations des deux catégories « Administration » et « Compétences » et une partie des informations de catégorie « Activités » pour constituer une base de données qui sera utilisée pour la pondération des critères. Les informations pertinentes extraites depuis « Administration » et « Compétences » dépendent du type de l'application. Il n'y a pas de principe général pour traiter ce problème. Lors de la construction de l'application, le concepteur doit identifier quelles sont les informations du profil qui influenceront la réalisation des critères ou des objectifs. Nous nous basons sur cela pour extraire les données pertinentes et les conserver comme le Tableau 7. Supposons que nous extrayons *n* informations d'un utilisateur depuis deux catégories Administration et Compétences et *m* critères à satisfaire. Un enregistrement dans notre base de données utilisée est formalisé par :

$$(user, info_1, \dots info_n, état_{critère\ 1}, \dots, état_{critère\ m})$$

Tableau 7. Base de données pour la phase de pondération des critères

Utilisateur	**info*1***	**…**	**info*n***	**Critère *1***	**…**	**Critère *m***
A	val_1	...	val_i	Succès	...	Échec
B	val_2	...	val_j	Échec	...	Succès
...	...	...	...	...	...	...

Le Tableau 7 représente le format de la base de données que nous allons utiliser pour résoudre le problème de pondération des critères. Par exemple, le profil de l'utilisateur A est défini par le vecteur *(val_1, val_2,..., val_i)*. Une fois l'exécution terminée, l'observation des critères permet de quantifier le taux de satisfaction des critères. Nous enregistrons plusieurs exécutions afin de constituer notre base de données.

Quand un nouvel utilisateur arrive, nous prenons son profil pour prédire l'état d'accomplissement des critères selon son profil. Si un utilisateur a de meilleures connaissances ou compétences que les autres, il est susceptible d'avoir la capacité de mieux satisfaire les critères que les autres.

À partir des données collectées, nous construisons un modèle de prédiction en utilisant les algorithmes de classification. Le but est d'associer des poids correspondant aux critères définis. Le poids considéré dans notre contexte est une valeur variant dans l'intervalle [0, 1]. Un critère est associé à un poids *w* qui permet d'interpréter l'importance d'un critère par rapport aux autres critères. La somme des poids des critères est égale à 1. Formellement, nous avons un ensemble $C = (C_1, C_2, \dots, C_m)$ de m critères, w_h représente le poids du critère h, et l'ensemble des poids doit être vérifié :

$$\sum_{h=1}^{m} w_h = 1 \tag{31}$$

Tableau 8. Exemple des poids des critères (cas du Tamagotchi)

Critère	Santé	Socialisation	Maturité
Poids	0,45	0,25	0,3

Un exemple du résultat de pondération des critères est présenté dans le Tableau 8. Il s'agit trois critères à satisfaire qui sont : la santé, la socialisation et la maturité. Nous supposons qu'après le calcul de pondération, nous obtenons des poids indiquant l'importance de chaque critère par rapport aux autres. Le résultat de cette étape est une liste des valeurs associées aux critères. Il constitue l'un des moyens qui contribuent à la méthode de décision.

4.2.2 NATURE DE NOTRE APPROCHE

Nous allons détailler dans la suite notre méthodologie de pondération des critères pour un nouvel utilisateur. Le but est d'utiliser la base de profils des utilisateurs et les traces associées, comme le montre le Tableau 7, pour prédire l'état des critères. Les utilisateurs veulent atteindre l'objectif défini, et cela est évalué par m critères.

Les informations dans le profil représentent les compétences, les connaissances de l'utilisateur, celles qui influenceront à la satisfaction des critères lors de l'exécution d'une application. Par exemple, si nous avons une application à donner à l'utilisateur à réaliser qui demande une compétence avancée en informatique, si un utilisateur de haut niveau informatique connectera à l'application, il sera évidemment facile de la manipuler en satisfaisant les critères définis associés par rapport aux utilisateurs de bas niveau informatique. Ce que nous voulons montrer dans notre méthodologie est la dépendance entre le profil d'utilisateur et la satisfaction des critères. Un bon profil pourra augmenter la capacité de satisfaire les critères. D'après cette idée, nous pensons extraire les informations implicites de la base de données pour pouvoir prédire le résultat des critères et cela sera utilisé pour pondérer les critères lorsqu'il y a un nouvel utilisateur qui arrive et manipule l'application.

Sur la base des profils et des traces des utilisateurs, nous estimons l'état de satisfaction des critères par le calcul ses probabilités associées. Ces probabilités représentent la capacité de satisfaction de critères. Nous nous rendons compte que ce problème est un problème de classification binaire. Le Tableau 9 présente une analyse comparative des méthodes les plus utilisées. Nous avons évalué ces méthodes selon 4 dimensions : la mise en œuvre, le temps de construction du modèle de prédiction (nous appelons le temps *t* qui est le temps nécessaire pour construire le modèle de prédiction), les données traitées, les faiblesses. Les quatre

méthodes suivantes, issues de la famille d'approches de fouille de données, sont susceptibles de convenir pour construire un modèle de prédiction : Naïve Bayes [79] [80], k voisins les plus proches (k-NN) [81], Neural Network [81], Support Vector Machine (SVM) [82].

Tableau 9. Récapitulatif des méthodes de prédiction

Méthode	**Mise en œuvre**	**Temps de construction du modèle de prédiction**	**Données traitées**	**Inconvénient**
Naïve Bayes	Simple, basée sur la probabilité Bayes	*t*	continues ou discrètes	Diminution de la capacité s'il existe des attributs superflus
kNN	Simple, basée sur le calcul de distance	Très long en raison du re-calcul la distance de toutes les données	continues ou discrètes	Besoin d'une grande base de données
Réseau neurone	Complexe, basée sur le fonctionnement des neurones biologiques en appliquant les méthodes probabilistes	10*t*	continues ou discrètes	Dépendance du nombre de neurones fixées
SVM	Complexe, basée sur la recherche d'un hyperplan qui minimise la distance entre deux marges	8*t*	continues	Très complexe, des paramètres de SVM sont difficiles à déterminer

Nous décidons finalement d'utiliser l'algorithme Naïve Bayes parce qu'il est le plus adapté à notre problème. Les données que nous avons vues et utilisées dans notre cas sont de différents types (continue ou discrète). Ceci est la raison pour laquelle nous n'utilisons pas SVM bien que sa performance de prédiction soit très élevée. En outre, le procédé de Bayes est simple et facile à mettre en œuvre. En raison de la petite taille de notre base de données, la méthode k-NN ne nous permet pas d'obtenir une performance efficace. Entre les deux méthodes restantes, Naïve Bayes et Neural Network, nous nous rendons compte que Neural Network doit choisir un nombre de neurones et le temps de construction du modèle de prédiction est très élevé. Cela ne correspond pas à une application en temps réel. Pour conclure, nous considérons que la méthode Naïve Bayes est bien le plus appropriée pour notre cas.

4.2.3 PRINCIPE GENERAL DE LA METHODE NAÏVE BAYES

La méthode Naïve Bayes est une méthode de classification Bayésienne probabiliste simple basée sur le théorème de Bayes avec une forte indépendance (dite naïve) des hypothèses. Elle met en œuvre un classifieur bayésien naïf, appartenant à la famille des classifieurs linéaires. En termes simples, un classifieur bayésien naïf suppose que l'existence d'une caractéristique pour une classe, est indépendante de l'existence d'autres caractéristiques. Malgré leur conception

naïve et leurs hypothèses de base extrêmement simples, les classifieurs bayésiens naïfs ont fait preuve d'une efficacité plus que suffisante dans beaucoup d'applications réelles.

Le modèle probabiliste pour un classifieur est un modèle conditionnel $P(A/p_1, p_2, \dots, p_t)$ où A est une variable de classe dépendante dont les instances sont conditionnées par plusieurs caractéristiques $p_1, p_2, \dots, p_t$. Lorsque le nombre de caractéristiques t augmente fortement, se baser ce modèle sur des tableaux de probabilités devient impossible. À l'aide du théorème de Bayes, nous pouvons énoncer des probabilités conditionnelles : étant donné un événement A et un ensemble de caractéristiques $p_1, p_2, \dots, p_t$; le théorème de Bayes permet de calculer la probabilité de A sachant ses caractéristiques, si l'on connaît les probabilités de A, de $p_1, p_2, \dots, p_t$ et de $p_1, p_2, \dots, p_t$ sachant A.

$$P(A \,/\, p_1, \dots, p_t) = \frac{P(p_1, \dots, p_t/A) * P(A)}{P(p_1, \dots, p_t)} \tag{32}$$

Avec

$P(A)$ est la probabilité a priori de A,

$P(A \,/\, p_1, \dots, p_t)$ est appelée la probabilité a posteriori de A sachant $p_1, \dots, p_t$

$P(p_1, \dots, p_t/A)$ est appelé la vraisemblance de $p_1, \dots, p_t$ sachant A.

Nous allons maintenant appliquer la méthode Naïve Bayes pour pondérer les critères.

4.2.4 APPLICATION DE NAÏVE BAYES A LA PONDERATION DES CRITERES

Considérons une application interactive où un ou plusieurs utilisateurs interagissent avec le système pour satisfaire un ou plusieurs critères. Supposant que nous avons m critères, notre objectif maintenant est d'associer une pondération à chaque critère. Cette valeur représente le niveau d'importance d'un critère par rapport aux autres. Plus la pondération est élevée, plus le critère a de l'importance.

Comme indiquée dans la section 4.2.2, lorsqu'un utilisateur est représenté par un vecteur I de n dimensions :

$$I = \{info_1, info_2, \dots, info_n\} \tag{33}$$

Basé sur le vecteur I de l'utilisateur, nous prédisons sa capacité d'évaluation des critères par le calcul de la probabilité à l'aide de la méthode Naïve Bayes. Un critère pourra être **Succès** (l'utilisateur a satisfait ce critère) ou **Échec** (ne pas satisfaire ce critère). Pour savoir l'état d'un critère, nous estimons la probabilité d'être Succès ou Échec de l'utilisateur basé sur le vecteur I.

Tableau 10. Probabilité de prédiction pour m critères

	Critère 1	**Critère 2**	**…**	**Critère m**
Succès	$Predict_1(Succès)$	$Predict_2(Succès)$		$Predict_m(Succès)$

Échec	$Predict_1(\text{Échec})$	$Predict_2(\text{Échec})$		$Predict_m(\text{Échec})$

Avant d'obtenir le résultat montré dans le Tableau ci-dessus, il nous faut construire un modèle de prédiction en appliquant Naïve Bayes. Afin d'avoir ce modèle, nous avons besoin d'effectuer une phase d' « apprentissage ». Cette phase permet d'accéder à la base de données et de parcourir tous les enregistrements de la base pour calculer les valeurs statistiques correspondantes. Les valeurs obtenues pour le modèle dépendent du type des données analysées donc du vecteur I.

La valeur $P_h(Succès/I)$ indique la probabilité d'être à l'état Succès sachant le vecteur I du critère h. Elle est calculée par application du théorème de Bayes par la probabilité a posteriori. Nous pouvons écrire :

$$P_h(Succès/I) = \frac{P_h(Succès) \times P_h(I\ /Succès)}{P_h(I)} \tag{34}$$

$$P_h(\text{Échec}/I) = \frac{P_h(\text{Échec}) \times P_h(I\ /\text{Échec})}{P_h(I)} \tag{35}$$

L'état d'un critère est décidé après voir comparé ces deux probabilités. Celle qui est supérieure à l'autre représente l'état correspondant du critère. Les deux formules deviennent :

$$\begin{aligned} P_h(Succès/I) &= P_h(Succès) \times P_h(I\ /Succès) \ = P_h(Succès) \times P_h(info_1, \dots, info_n/Succès) \\ &= P_h(Succès) \times \prod_{i=1}^{n} P_h(info_i/Succès) \end{aligned} \tag{36}$$

$$\begin{aligned} P_h(Echec/I) &= P_h(Echec) \times P_h(I\ /Echec) = P_h(Echec) \times P_h(info_1, \dots, info_n/Echec) \\ &= P_h(Echec) \times \prod_{i=1}^{n} P_h(info_i/Echec) \end{aligned} \tag{37}$$

Nous pourrons alors compléter le Tableau 10 après une phase de normalisation :

$$Predict_h(Succès) = \frac{P_h(Succès/I)}{P_h(Succès/I) + P_h(Echec/I)} \tag{38}$$

$$Predict_h(Echec) = \frac{P_h(Echec/I)}{P_h(Succès/I) + P_h(Echec/I)} \tag{39}$$

Après avoir calculé la valeur de prédiction pour chaque critère, nous avons constaté un principe sur lequel nous nous basons pour pondérer les critères. Le principe est énoncé comme suit :

« Si un utilisateur a tendance à atteindre un critère dans un ensemble de critères, la prise en compte de la priorité sur les autres peut être justifiable. Nous devons, au contraire, le mettre en priorité des critères dont les valeurs de prédiction d'évaluation sont faibles afin de permettre à l'utilisateur de remplir tous les critères ».

En appliquant ce principe, nous nous concentrons sur les valeurs de prédiction « **Échec** » des critères. Le poids d'un critère *h* dans un ensemble *m* critères est calculé par :

$$w_h = \frac{Predict_h(Echec)}{\sum_{j=1}^{m} Predict_j(Echec)} \quad (40)$$

L'algorithme de pondération des critères est décrit comme suit :

Algorithme 1 : Processus de pondération des critères

Entrée : le vecteur d'état *U* et l'ensemble de *m* critères

1 : **let** $W = \emptyset$ (ensemble des poids des critères)
2 : **for all** *h* **do** (*h* from 1 to *m*)
3 : **compute** $P_h(Succès/U)$, $P_h(Echec/U)$
4 : **compute** $Predict_h(Succès)$, $Predict_h(Echec)$
5 : **compute** w_h
6 : $W = W \cup \{w_h\}$
7 : **end for**

Sortie : l'ensemble des poids des critères *W*

Nous obtenons alors la liste des poids des critères. Notre mécanisme de décision multicritère se poursuit par une approche d'identification des situations candidates pour la décision.

4.2.5 PUBLICATIONS SCIENTIFIQUES

La contribution sur la pondération de critère a donné lieu à 1 article dans une conférence internationale [83] qui a été retenu dans une revue internationale [84].

- Article 1 :
  - Titre : Trace-Based Weighting Approach for Multiple Criteria Decision Making ;
  - Revue : *Journal of Software*
- Article 2 :
  - Titre : A process for Trace-Based Criteria Weighting in Multiple Criteria Decision Making ;
  - Conférence : *6th International Conference on Computer Research and Development*, 2014.

Ces deux articles résument le processus de pondération des critères à bases de traces dans les applications interactives. Au cours de l'utilisation de l'application, l'utilisateur est caractérisé par un profil contenant ses informations. Lorsqu'il veut prendre de décision, il faut tenir en compte un multiple critère. Chaque critère comporte deux états : Succès ou Échec. Pour chaque utilisateur, nous élaborons un profil en ajoutant l'état des critères. Nous appliquons la

méthode présentée, Naïve Bayes, pour prédire la capacité d'atteinte des critères d'un nouvel utilisateur. Ces valeurs sont utilisées pour pondérer les critères.

4.3 CONTRIBUTION 3 : METHODE DE SELECTION DES SITUATIONS CANDIDATES

La détermination des situations candidates lors de la décision est une étape importante à faire avant de lancer l'algorithme de décision multicritère. Pourtant, il n'existe pas d'approche intégrant cela dans les mécanismes de décision. Dans notre contexte à base de situations, à chaque sortie d'une situation, nous disposons d'un vecteur d'état global *V* représente les attributs (*att*) du système qui ont été observées à la fin de l'exécution de la situation actuelle. Nous vérifions d'abord quelles sont les situations satisfaisant leurs pré-conditions avec le vecteur d'état V. Ces situations représentent l'ensemble des **situations activées**. Nous ne pouvons pas appliquer l'algorithme de décision pour prendre une décision sur toutes les situations activées de l'application en raison de l'augmentation du temps de calcul qui dépend du nombre de situations à traiter et de la complexité des algorithmes de décision multicritère.

4.3.1 BASE DE TRACES UTILISEE POUR LA SELECTION DES SITUATIONS CANDIDATES

Depuis la base de données de la section 4.1.3, nous n'extrayons que les informations dans la catégorie « Activités », les traces d'activités de l'utilisateur lors des exécutions antérieures de l'application. Selon le principe défini dans le Chapitre 3, nous obtenons une base de traces premières. Dans cette section, nous effectuons la phase de transformation dans notre système à base de traces pour extraire les informations pertinentes qui seront utilisées pour la sélection des situations candidates.

Nous avons besoin d'une base de traces dans laquelle chaque enregistrement se rapporte au contexte suivant : Quel est l'état du système ? Quelle est la situation choisie pour âtre exécutée ? Quel est le changement de l'évaluation des critères ? Depuis la base de traces premières, nous extrayons des traces transformées dont chaque d'enregistrement contient des composants suivants :

- $\{V, S\}$: est l'ensemble du vecteur d'état *V* associé avec la situation exécutée *S*.

Tableau 11. Format du composant Ω dans la base de données

Attribut *1*	Attribut *2*	….	Attribut *p*	Situation exécutée
att_1	att_2	….	att_p	sit_i

- Δ : l'ensemble des critères et leurs valeurs d'évaluation avant et après avoir exécuté une situation. Formellement, un enregistrement j dans Δ est représenté par : $\langle\{\gamma^j_{avant}(h)\} \times sit_i \times \{\gamma^j_{après}(h)\}\rangle$ avec *h* indiquant le critère *h*, *h* variant de 1 à *m* et correspondant aux critères définis.

Tableau 12. Format du composant C dans la base de données

avant	Situation exécutée	après

$\gamma^1_{avant}(1)$	....	$\gamma^1_{avant}(m)$	sit_i	$\gamma^1_{après}(1)$	....	$\gamma^1_{après}(m)$

Les deux composants indiqués dans les Tableau 11 et Tableau 12 seront utilisés séparément pour notre algorithme de sélection des situations candidates dans la section suivante.

4.3.2 ALGORITHME DE SELECTION DES SITUATIONS CANDIDATES

Cette section présente notre stratégie visant à identifier l'espace des solutions (situations candidates) pour la prise de décision. Pour ce faire, nous utilisons une base de traces transformée obtenue dans la section 4.3.1. Notre méthode comporte deux étapes :

- la prédiction des situations potentielles ;
- l'estimation de l'utilité de ces situations potentielles.

La première étape consiste à évaluer la capacité à être potentiellement exécutable à la prochaine exécution ceci selon l'état actuel du système de toutes les situations activées. Pour chaque situation activée, nous calculons la probabilité d'être potentiellement exécutable. Nous obtenons un ensemble de probabilités liées à toutes les situations activées. Une **situation potentielle** est celle qui a une probabilité calculée supérieure à un seuil s défini et fixé par le concepteur ou l'utilisateur. Si une situation est considérée comme potentielle, nous l'ajouterons à l'ensemble des situations potentielles.

La deuxième étape calcule pour les situations potentielles leur pertinence selon les critères définis. Cette mesure, appelée utilité (voir 2.2.2.2.2), permet aux utilisateurs de comparer la pertinence des situations obtenues dans la première étape. La combinaison de ces deux étapes nous aide à déterminer quelles sont les situations candidates pour le processus décisionnel et de quantifier leur impact potentiel sur l'évaluation des critères de l'application.

4.3.2.1 Prédiction des situations potentielles

La Figure 28 décrit notre stratégie de prédiction à base de traces des situations potentielles. Nous avons utilisé la base de traces extraite dans la section 4.3.1 pour construire le modèle de prédiction. Les traces obtenues sont analysées pour prédire quelles sont les situations potentielles en fonction de l'état actuel du système et d'un seuil d'acceptation défini par l'utilisateur.

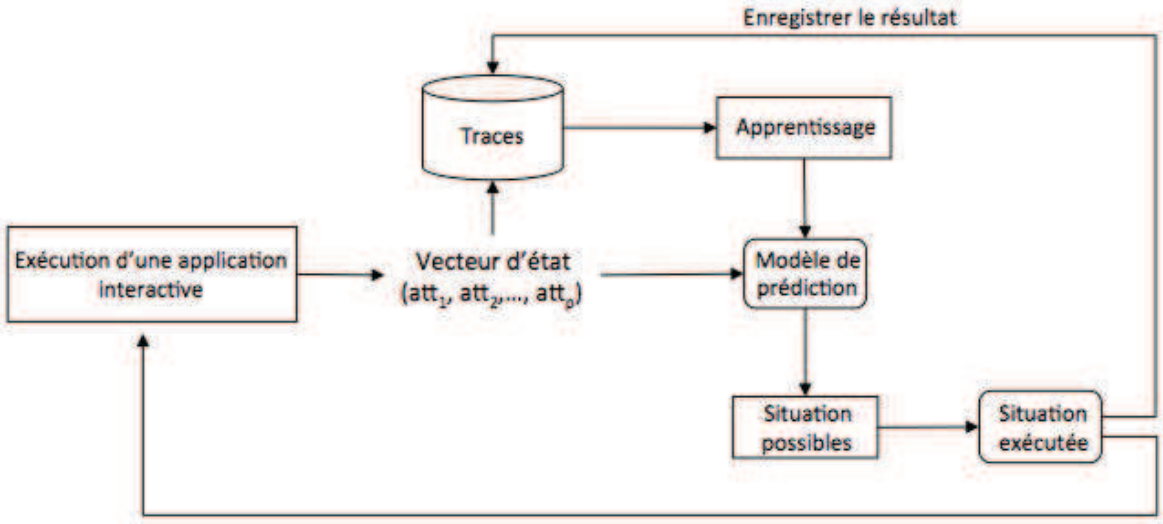

Figure 28. Processus de prédiction à base de traces des situations possibles

Nous construisons maintenant le modèle de prédiction en appliquant la méthode Naïve Bayes.

Le processus détaillé est présenté dans l'algorithme suivant.

Algorithme 2 : Processus de prédiction des situations potentielles

Entrée : le vecteur d'état *V* et l'ensemble *SA* de *n* situations activées

1 : **let** $Pot = \emptyset$ (l'ensemble des situations potentielles)
2 : **for all** sit_i **do** (i from 1 to n)
3 : **compute** $Predict(sit_i)$
4 : **if** $Predict(sit_i) \geq s$ **then**
5 : $Pot = Pot \cup \{sit_i\}$
6 : **end if**
7 : **end for**

Sortie : l'ensemble des situations potentielles *Pot*

La probabilité d'être potentiellement exécutable d'une situation i à partir d'un état $V = \{att_1, att_2, \dots, att_p\}$, noté $Predict(sit_i)$ est calculée par :

$$Predict(sit_i) = \frac{P(sit_i/V)}{\sum_{i=1}^{n} P(sit_i/V)} \quad (41)$$

Avec

$$P(sit_i/V) = P(sit_i) \times \prod_{j=1}^{m} P(att_j/sit_i) \quad (42)$$

Dans la formule ci-dessus, il nous faut calculer $P(att_j/sit_i)$. Le calcul de cette valeur dépend du type de donnée de att_j. Si elle est numérique, nous supposons que ses valeurs respectent la

distribution de Gauss et la probabilité de att_j sachant sit_i est calculée par l'expression suivante :

$$P(att_j/sit_i) = \frac{1}{\sqrt{2\pi}\times\sigma_j^i}\times e^{\frac{-(att_j-\mu_j^i)^2}{2\times(\sigma_j^i)^2}} \quad (43)$$

avec μ_j^i et σ_j^i étant respectivement la moyenne et l'écart type de l'attribut *j* pour la situation *i*. Si la valeur de att_j n'est pas numérique, nous devrons calculer la probabilité de att_j sachant la situation *i* dans la base de données pour obtenir la probabilité $P(att_j/sit_i)$.

Après avoir calculé pour chaque situation activée, la probabilité d'être potentiellement exécutable, nous obtenons une liste des résultats. Nous devons ensuite effectuer la vérification par la comparaison avec le seuil *s*. La valeur du seuil dépend du choix de l'utilisateur ou de concepteur. Le choix du seuil sera présenté dans notre cas d'étude dans la section 6.2.3. Une situation est considérée comme situation potentielle si sa probabilité est supérieure ou égale à *s*.

4.3.2.2 Estimation de l'utilité des situations potentielles

Cette étape vise à estimer, pour chaque situation potentielle, son utilité, ce qui représente l'impact de la progression de la valeur du critère au cours des exécutions précédentes. Pour évaluer l'utilité de chaque situation potentielle, nous avons besoin d'analyser les traces laissées dans les exécutions antérieures. Depuis la base de traces transformées, nous extrayons l'ensemble *C* qui contient des enregistrements dans lesquels sont décrits : la situation exécutée et le changement des valeurs des critères avant et après avoir exécuté cette situation. Nous avons commencé par le calcul de la déviation *d* de chaque critère. Cette valeur est la différence entre l'évaluation du critère *h* avant et après avoir exécuté une situation. Supposant que nous considérons l'enregistrement *j* dans la base de traces, la déviation est obtenue par :

$$d_h^j = \gamma_{après}^j(h) - \gamma_{avant}^j(h) \quad (44)$$

Si nous avons une base contenant *q* enregistrements, la déviation globale du critère *h* est calculée par :

$$d_h = \frac{\sum_{j=1}^{q}\left(\gamma_{après}^j(h) - \gamma_{avant}^j(h)\right)}{q} = \frac{\sum_{j=1}^{q} d_h^j}{q} \quad (45)$$

Nous utilisons la déviation obtenue pour définir notre fonction d'utilité $u(d_h)$ où p_h représente le seuil d'acceptation d'une déviation d'un critère *h*. Ce seuil est défini expérimentalement par le concepteur ou par l'utilisateur. Chaque critère a un seuil différent. La valeur d'utilité d'un critère *h* est calculée en fonction de la déviation obtenue dans (45). Nous voulons avoir une fonction qui permet de déterminer si la valeur des critères peut nous apporter des utilités positives ou négatives. Cette fonction se base sur la déviation entre les valeurs pour identifier l'utilité des critères. C'est pour cela que nous fixons un seuil p_h pour bien distinguer le niveau d'utilité comme le montre.

$$u(d_h) = \begin{cases} -1 & if\ d_h < 0 \\ 0 & if\ d_h = 0 \\ \dfrac{d_h}{p_h} & if\ 0 < d_h < p_h \\ 1 & if\ d_h \geq p_h \end{cases} \quad (46)$$

Comme décrit dans la Figure 29, si la déviation est négative (la situation exécutée a diminué la valeur d'évaluation du critère), la valeur d'utilité correspondante sera négative. Si la déviation est supérieure à un seuil p_h, la valeur d'évaluation du critère a augmenté après l'exécution de la situation, l'utilité sera égale à 1. L'utilité sera égale à 0 si et seulement si la déviation est égale à 0. Si non, l'utilité associée est $\frac{d_h}{p_h}$.

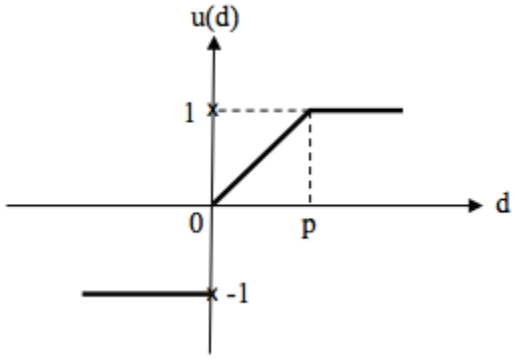

Figure 29. Fonction d'utilité

Nous appliquons cette approche pour estimer l'utilité des m critères de l'application pour chaque situation possible obtenue ci-dessus. Une situation possible est considérée comme candidate s'il existe au moins *K* critères sur *m* dont l'utilité est supérieure à 0. Dans le cas contraire, nous la considérons non-candidate. Toutes les situations obtenues au bout de cette phase constituent un ensemble de candidates pour la prise de décision multicritère présentée dans la section suivante. L'algorithme suivant résume notre méthodologie :

Algorithme 3 : Processus d'estimation de l'utilité des situations potentielles

Entrée : l'ensemble de situations potentielles *Pot,* l'ensemble de *m* critères

1 : **let** $Cand = \emptyset$ (ensemble des situations candidates)
2 : **for all** sit_i **do** (i from 1 to n)
3 : **let** $U = \emptyset$
4 : **for all** h **do** (h from 1 to m)
5 : **compute** $u(d_h)$
6 : **if** $u(d_h) \geq 0$ **then**
7 : $U = U \cup \{h\}$
8 : **end if**
9 : **end for**
10 : **if** $size(U) \geq K$ **then**
11 : $Cand = Cand \cup \{sit_i\}$
12 : **end if**
13 : **end for**

Sortie : l'ensemble des situations candidates *Cand*

Nous prenons un petit exemple tiré de notre cas d'étude présenté le Chapitre 5 : le calcul d'utilité d'une situation « Manger ». Depuis la base de traces, nous extrayons un enregistrement dans lequel chaque composant est décrit par :

$\gamma_{avant}(Santé) = 0{,}2$; $\gamma_{avant}(Socialisation) = 0$; $\gamma_{avant}(Maturité) = 1$, $sit_i = Manger$

$\gamma_{après}(Santé) = 0{,}8$; $\gamma_{après}(Socialisation) = 0$; $\gamma_{après}(Maturité) = 1{,}5$

Pour calculer l'utilité de la situation Manger, nous calculons la déviation de chaque critère ci-dessus. Nous avons :

$$d_{Santé} = 0{,}8 - 0{,}2 = 0{,}6$$
$$d_{Socialisation} = 0 - 0 = 0$$
$$d_{Maturité} = 1{,}5 - 1 = 0{,}5$$

Nous allons ensuite appliquer la fonction d'utilité avec les 3 seuils : $p_{Santé} = 0{,}3$; $p_{Socialisation} = 0{,}7$; $p_{Maturité} = 0{,}55$; nous obtenons :

$$u_{Santé} = u(d_{Santé}) = 1 \quad parce\ que\ \ d_{santé} = 0{,}6 > 0{,}3$$
$$u_{Socialisation} = u(d_{Socialisation}) = 0 \quad parce\ que\ \ d_{Socialisation} = 0$$
$$u_{Maturité} = u(d_{Maturité}) = \frac{0{,}5}{0{,}55} = 0{,}9 \quad \text{parce que } 0 < d_{maturité} = 0{,}5 < 0{,}55$$

En nous basant sur cette valeur, nous pourrons filtrer alternatives à partir des situations potentielles. L'ensemble de ces solutions constitue l'ensemble des situations candidates qui pourra utiliser dans les algorithmes de prise de décision multicritère.

4.3.3 PUBLICATIONS SCIENTIFIQUES

Les travaux dans cette contribution ont donné lieu à 1 article dans une conférence internationale [85].

Titre : Trace-Based Decision Making in Interactive Application: Case of Tamagotchi systems.

Conférence : *IEEE International Conference on Control, Decision and Information Technologies*, 2014.

L'idée principale de cet article est l'appui sur l'exploitation des traces pour déterminer les situations candidates pour la prise de décision dans la phase suivante. Nous avons alors considéré les traces collectées lors de l'exécution de l'application. Ainsi, en nous basant sur les états du système et la base de traces ainsi construite, nous calculons l'option optimale, ce qui fait que pour chaque alternative, nous obtenons une probabilité indiquant quelle alternative pourra être exécutée en tenant compte de la tendance des utilisateurs antérieurs.

4.4 CONTRIBUTION 4 : METHODE DE DECISION MULTICRITERE

Cette phase est la plus importante de notre processus de décision multicritère. Elle vise à trouver une situation dans l'ensemble des situations candidates identifiées ci-dessus. Nous

allons donc proposer dans la Figure 30 une approche de décision multicritère respectant les deux aspects suivants :

- Classer les situations candidates en une liste de priorité selon différents points de vue, notamment celui de l'utilisateur, celui du système et celui du concepteur ;
- Combiner les différents points de vue pour avoir une liste de choix qui seront suggérés à l'utilisateur pour l'exécution suivante.

Chaque point de vue ci-dessus donne un choix, dit **choix individuel**. Dans notre approche, nous proposons de reformuler ce choix individuel et de combiner ces différents choix de la façon suivante :

- Point de vue du système : donne un choix individuel du système ;
- Point de vue de l'utilisateur : donne un choix individuel de l'utilisateur.
- Point de vue du concepteur.

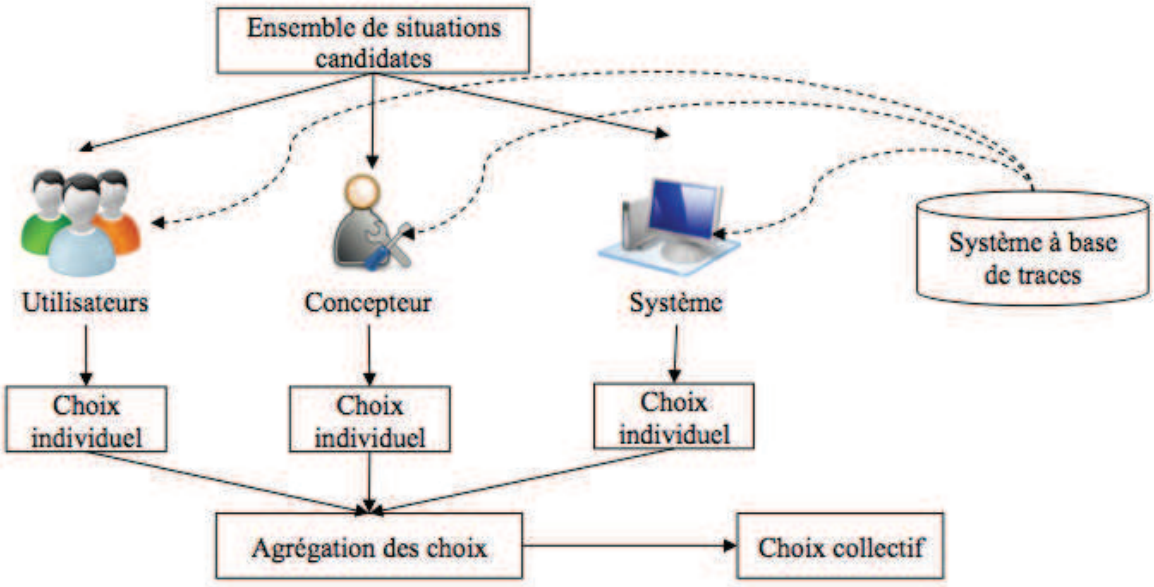

Figure 30. Architecture de notre approche de prise de décision multicritère

4.4.1 CHOIX DU SYSTEME

Le choix du système va utiliser les états du système pour classer automatiquement les situations candidates. Pour mettre en œuvre ce choix, nous avons décidé d'utiliser les techniques de décision multicritère existantes dans la communauté. Nous nous intéressons plus particulièrement à la méthode PROMETHEE II (voir section 2.2.2.3.3) parce qu'elle a démontré ses performances dans plusieurs cas d'applications [86]. Elle vise à classer des situations candidates sous forme d'une liste de priorités. En nous basant sur cette liste, nous pouvons déduire quelle est la solution optimale recherchée et la proposer à l'utilisateur. La Figure 31 détaille les 5 étapes de cette méthode pour résoudre un problème ayant un ensemble *C* de *m* critères et un ensemble A de *n* candidates.

Cependant, en analysant ce processus, nous constatons rapidement une faiblesse dans la première étape. PROMETHEE II ne nous fournit pas une méthode pour déterminer

l'évaluation des alternatives pour chaque critère. C'est l'utilisateur qui prend en charge cette mission. Les évaluations sont déterminées intuitivement selon la perception de l'utilisateur.

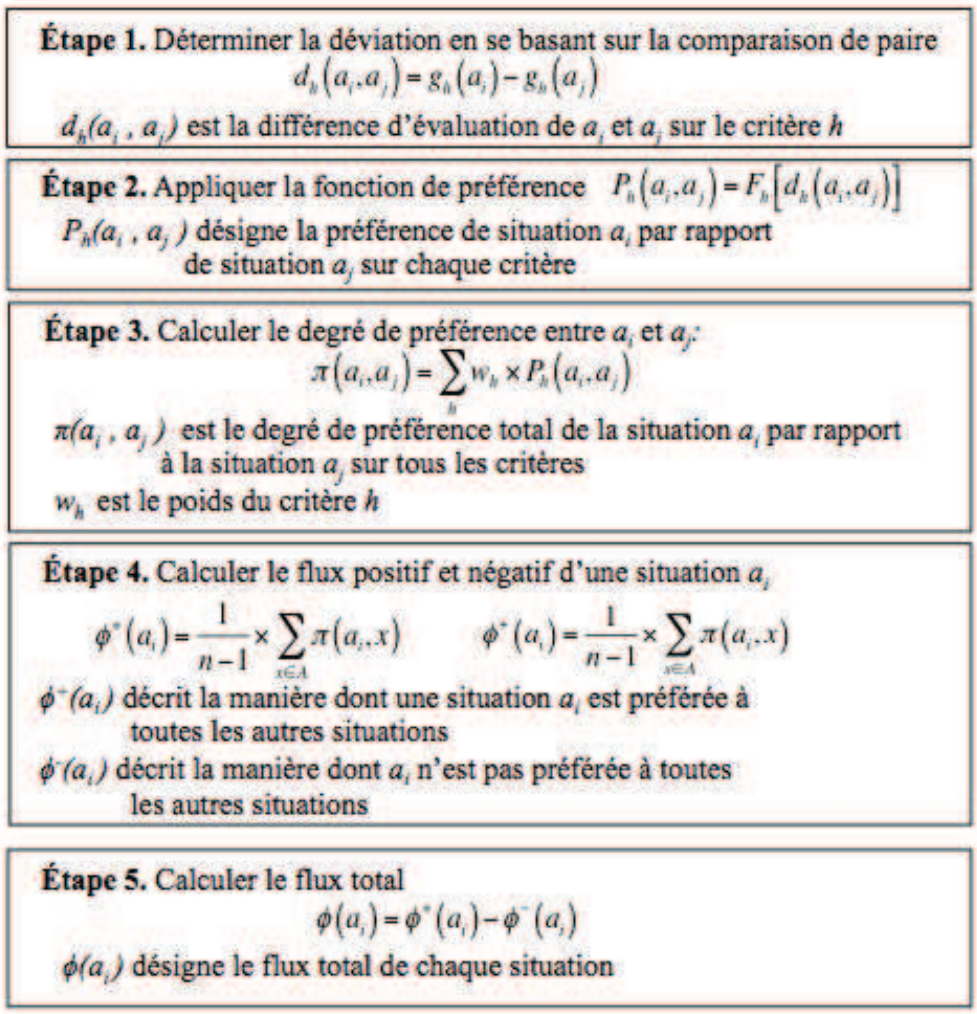

Figure 31. Processus de PROMETHEE II classique

Dans notre contexte, à chaque instant de l'exécution, nous disposons de l'état du système représenté par un vecteur contenant les états des composants du système contribuant à l'application, vecteur *V*. Dans PROMETHEE II, ces valeurs ne sont pas prises en compte pour évaluer les alternatives selon les critères définis, l'évaluation étant plutôt paramétrée intuitivement par les préférences du décideur. Nous avons cherché à surmonter cet inconvénient et étendre l'approche classique en améliorant le processus de décision par l'exploitation des traces générées lors des exécutions passées d'une application destinée aux environnements d'apprentissage. Cette approche nous permet d'obtenir la valeur de la fonction *g* dédiée à l'évaluation des critères.

4.4.1.1 Base de traces utilisée pour l'évaluation des situations candidates

Depuis la base de données de la section 4.1.3, nous n'extrayons qu'une partie des informations dans les traces d'activités de l'utilisateur obtenues lors des exécutions antérieures de l'application.

Nous avons besoin d'une base de traces dans laquelle chaque enregistrement décrit le contexte suivant : avec quel état du système la situation exécutée a-t-elle été choisie ? Depuis la base de traces premières, nous extrayons des traces transformées dont chaque d'enregistrement a le format suivant : $V \times S$ (l'ensemble du vecteur d'état associé à la situation exécutée). Cette base de données est similaire à celle utilisée dans la section 4.3.1.

De plus, nous précisons dans le vecteur d'état V quel attribut du vecteur d'état contribue à l'évaluation des critères. Un attribut pourra contribuer à un ou plusieurs critères d'évaluation. Par exemple, dans le cas d'étude Tamagotchi (voir le Chapitre 5), le vecteur d'état $V = \{att_1, att_2, att_3, att_4, att_5, att_6\}$ contient 6 attributs avec les 3 premiers attributs contribuant au critère Santé, l'attribut 4 contribuant au critère Socialisation et les 2 derniers attributs contribuant au critère Maturité. Par ailleurs, il est possible qu'un attribut contribue à plusieurs critères en même temps. Avec l'attribut 1, 2 et 3 contribuent au critère Santé, l'attribut 1 et 4 contribuent au critère Socialisation, l'attribut 1, 4 et 5 contribuent au critère Maturité. L'attribut 1 contribue en même temps dans 3 critères (Santé, Socialisation et Maturité) et l'attribut 4 contribue au 2 critères (Socialisation et Maturité).

C'est pour cela que nous créons m ensembles correspondant à *m* critères, dont chacun contient des enregistrements ayant un format comme suit :

$$eng = \left(att_1^h, att_2^h, \dots, att_r^h, sit_i\right)$$

Chaque enregistrement a r attributs dont chacun représente la valeur d'un attribut contribuant au critère h de l'application. À la fin de chaque enregistrement, nous gardons sit_i qui est la situation exécutée dans les traces transformées extraites depuis $\Omega \subseteq V \times S$.

Dans la section suivante, nous présentons la mise en œuvre des traces dans le processus de PROMETHEE II.

4.4.1.2 Approche à base de traces dans PROMETHEE II

À chaque instant de l'exécution, nous disposons d'un vecteur d'état global V contenant p attributs qui contribuent à l'exécution de l'application. Le système de décision multicritère PROMETHEE II se base sur le vecteur V selon *m* critères sur l'ensemble *Cand* de *n* candidates obtenues dans 4.3 pour choisir la meilleure solution en satisfaisant ces *m* critères. La Figure 32 décrit le processus d'évaluation des alternatives à base de traces pour chaque critère. Nous distinguons 3 étapes :

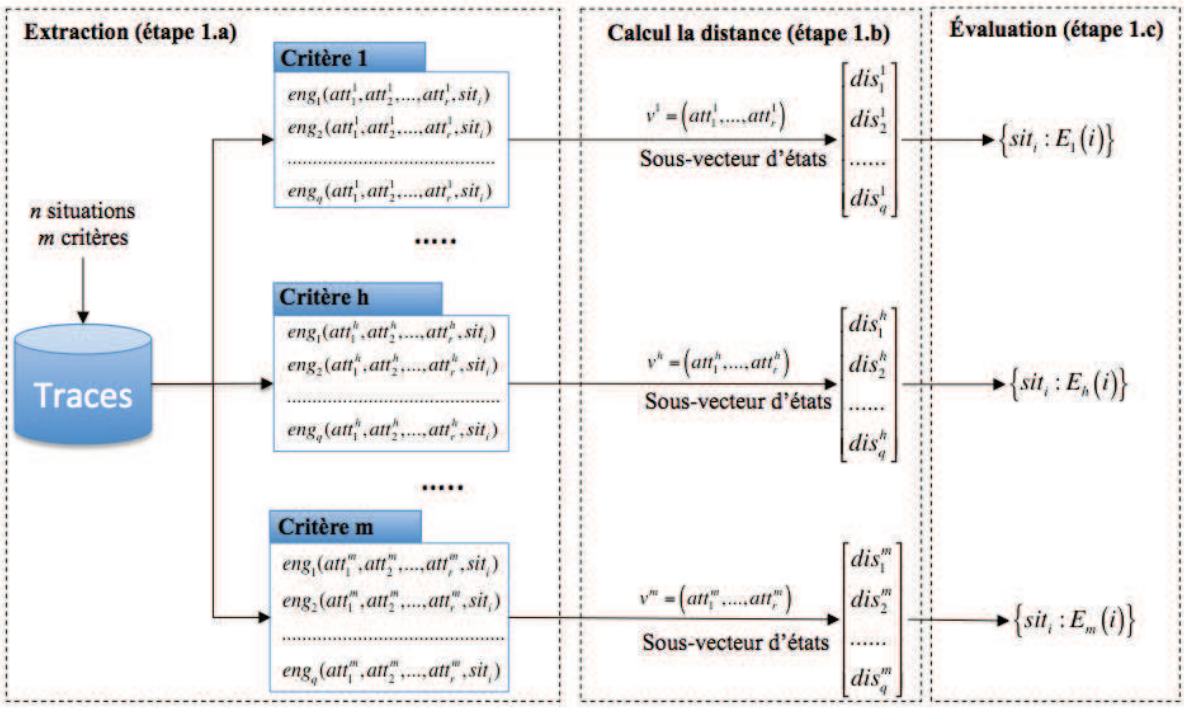

Figure 32. Approche à base de traces pour déterminer l'évaluation des situations candidates selon chaque critère

Étape 1.a : Extraction des données

Pour chaque critère, nous avons une base de traces transformées obtenue dans 4.4.1.1. Chaque enregistrement a le format : $(att_1^h, att_2^h, \dots, att_r^h, sit_i)$. Nous extrayons depuis la base q enregistrements comportant une des n situations candidates : on choisit *eng* tel que $sit_i \in$ A.

Par exemple, si nous avons les 3 situations candidates : sit_1, sit_3 et sit_8, il faut extraire les enregistrements dans lesquels sit_i est : sit_1 ou sit_3 ou sit_8. Puis, avec la base extraite, nous créons m ensembles en fonction des m critères. Un critère est évalué par un ou plusieurs attributs du vecteur d'états et par conséquent les enregistrements de chaque ensemble auront un format différent. Par exemple si $eng = (att_h^1, att_h^2, att_h^4, sit_3)$ est un enregistrement dans l'ensemble du critère h, ce critère h est évalué par 3 attributs. Plus généralement, chaque enregistrement dans l'ensemble du critère h est évalué par r attributs et est représenté par :

$$eng = (att_1^h, att_2^h, \dots, att_r^h, sit_i)$$

Enfin, nous décomposons le vecteur d'états V en m sous vecteurs selon les m critères. Le sous vecteur du critère h est décrit par

$$v^h = (att_1^h, att_2^h, \dots, att_r^h)$$

Étape 1.b : Calcul de la distance

Pour chaque ensemble obtenu dans la phase précédente, nous calculons la distance entre les sous vecteurs et les enregistrements. Il existe différentes méthodes pour calculer la distance entre deux données. La distance euclidienne [87] est souvent utilisée lorsque les vecteurs contiennent des valeurs continues. $dis_h^q(candidate\ i)$ représente la distance de la candidate i à

v^h et eng_q. Nous obtenons donc pour chaque critère une matrice de taille $q \times 1$ contenant toutes les distances entre le sous-vecteur d'état actuel et les q enregistrements.

Étape 1.c : Évaluation des alternatives selon chaque critère

À partir des m matrices précédentes, nous choisissons k plus proches voisins (méthode k-Nearest Neighbors [87]) pour effectuer la dernière phase qui consiste à calculer l'évaluation de la candidate i sur le critère h, notée $E_h(candidate\ i)$. Nous aimerons d'appliquer une méthode de calcul de distance sur un grand volume de données pour estimer la différence entre les valeurs. Pourtant, nous ne pouvons pas prendre compte toutes les données mais il nous faut limiter par un seuil. Cela nous donnons l'idée de l'utilisation la méthode k plus proches voisins.

$$E_h(candidate\ i) = \frac{e_h(candidate\ i)}{\sum_{i=1}^{n} e_h(candidate\ i)} \tag{47}$$

avec

$$e_h(candidate\ i) = \sum_{j=1}^{k} dis_h^j(candidate\ i) \tag{48}$$

Le Tableau 13 illustre le résultat de cette phase.

Tableau 13. Résultat d'évaluation des situations candidates selon l'ensemble des critères

	Candidate 1	**Candidate 2**	**….**	**Candidate n**
Critère 1	$E_1(candidate\ 1)$	$E_1(candidate\ 1)$	….	$E_1(candidate\ n)$
….	….	….	….	
Critère h	$E_h(candidate\ 1)$	$E_h(candidate\ 2)$	…..	$E_h(candidate\ n)$
….	….	….	….	
Critère m	$E_m(candidate\ 1)$	$E_m(candidate\ 2)$	….	$E_m(candidate\ n)$

Étape 1.d : Calcul de la déviation entre deux situations candidates

Ainsi, en reprenant l'étape 1 de Figure 28, nous remplaçons la formule de PROMETHEE II classique par l'expression suivante :

$$d_h(candidate\ i, candidate\ j) = E_h(candidate\ i) - E_h(candidate\ j) \tag{49}$$

Une fois ces calculs effectués pour chaque critère, le processus PROMETHEE II enchaîne avec les étapes restantes pour construire une liste prioritaire des alternatives, dont la première situation de cette liste sera la situation choisie à exécuter. Cette approche est combinée avec PROMETHEE II, nous l'appelons **PROMETHEE II à base de traces** (Annexe A).

4.4.2 CHOIX DU CONCEPTEUR

Le choix du concepteur est celui qui est calculé par les règles prédéfinies du concepteur. Le résultat attendu de ce choix est une liste de priorité des situations candidates qui a été déduit selon le point de vue du concepteur.

Avant d'exécuter une application interactive, le concepteur a défini lui-même ses règles d'exécution en se basant sur la structure des situations. La structure des situations concerne les pré-conditions des situations. C'est-à-dire si l'utilisateur veut exécuter une situation, sa pré-condition correspondante doit être vérifiée. Nous avons déjà mentionné ce problème dans la section 4.3, sélection des situations candidates. Ce sont des situations activées. Par ailleurs, le concepteur a défini ses règles afin d'assurer sa stratégie globale. Ce travail est fait par le concepteur lui-même en donnant les interdictions ou les instructions sur le choix de situation pour exécuter. Nous voulons mettre en disposition ce type de choix car il représente le point de vue de celui qui a conçu l'application. En d'autres termes, nous disons que le choix du concepteur implique une vue conceptuelle.

Ce type de choix est strictement fait par le concepteur lors de la conception de l'application, ce n'est pas la peine de proposer des approches pour le modéliser. Puisque nous voulons garder une partie stable qui ne change pas en temps réel pour assurer l'exécution de l'application qui compromit avec la logique du concepteur.

Concernant le choix de l'utilisateur, nous le mettons en place dans la section 4.4.3.

4.4.3 CHOIX DE L'UTILISATEUR

En ce qui concerne l'utilisateur, nous pensons que ce choix est très simple à exprimer. Il s'agit juste de classer les situations candidates selon l'idée, l'intérêt ou le besoin de l'utilisateur. Il n'existe aucun modèle ou approche qui permet d'expliquer pourquoi l'utilisateur choisit telle situation. Lors de la prise de décision, les utilisateurs donnent leurs idées pour chaque situation, par exemple, « *Nous pensons que la situation Éduquer sera bonne pour l'exécution en comparaison avec les autres*». C'est un exemple de l'idée d'un utilisateur sur la situation éduquer. Les utilisateurs donnent successivement leurs idées pour toutes les situations. La synthèse de ces idées est considérée comme les préférences de l'utilisateur.

Pour modéliser les préférences de l'utilisateur, il existe trois types de méthodes implicites: approche collaborative, approche à base du contenu et approche hybride. Les approches à base du contenu modélisent les préférences de l'utilisateur en se basant sur les caractéristiques des items obtenues par les techniques de fouilles de données [88]. L'objectif de cette approche est de comprendre les préférences qui ont conduit à l'utilisateur de juger la pertinence d'un item. En outre, l'approche collaborative vise à construire le modèle de préférences d'un utilisateur en se basant sur les éléments qui sont déjà utilisés et évalués par les anciens utilisateurs [16]. L'avantage de cette approche est qu'elle peut surmonter le problème de la non-disponibilité du contenu de ces items. Ces items ne sont pas toujours disponibles ou difficiles à caractériser leur contenu. L'approche hybride combine les principes des deux méthodes ci-dessus [16]. Cependant, un inconvénient de ces méthodes est qu'elles ignorent le problème de l'incertitude de l'utilisateur lors de la modélisation de préférences. Habituellement, les utilisateurs ne sont pas certains ou précis où ils ont mis préférences. Pour améliorer cet inconvénient, dans [88] les auteurs proposent un cadre qui applique la logique floues. Mais cette approche ne convient pas

à notre contexte à base situations, car elle requiert des informations supplémentaires qu'une situation ne permet pas d'avoir. Dans notre cas, les utilisateurs ne connaissent que le nom de la situation et ce qu'il est censé faire. Avec seulement cette information, ils doivent choisir une meilleure situation pour la prochaine exécution.

En analysant ce problème, nous constatons que la logique subjective [89], [90] est une méthode souvent utilisée dans le domaine de probabilité de raisonnement. Elle définit une approche pour exprimer des idées (appelées opinions) de l'utilisateur et les moyens pour les classer par ordre de priorité. La partie suivante du document va présenter la terminologie et les principes de la logique subjective.

4.4.3.1 Principe fondamental de la logique subjective

La logique subjective est un type de logique probabiliste qui prend en compte l'incertitude et la confiance. Elle fournit un ensemble standard d'opérateurs de logique, destinés à être utilisés dans les domaines contenant l'incertitude ou plus spécifiquement des domaines dans lesquels des avis concernant la véracité ou la fausseté d'un ou plusieurs éléments différents. En général, la logique subjective est adaptée pour la modélisation et l'analyse dans les cas comportant des incertitudes et des connaissances incomplètes.

Afin de comprendre la terminologie utilisée dans cette section, nous présentons certains concepts de la logique subjective.

Supposons que nous avons un ensemble Δ qui délimite un ensemble d'états d'un événement. L'union des états possibles dans Δ est exprimée par 2^{Δ}.

Affectation de masse de confiance

Étant donné un ensemble d'état Δ, nous pouvons affecter une masse de confiance $m_{\Delta}(x)$ à chaque sous-état $x \in 2^{\Delta}$ tels que :

$$\begin{aligned} m_{\Delta}(x) &\geq 0 \\ m_{\Delta}(\emptyset) &= 0 \\ \sum_{x \in 2^{\Delta}} m_{\Delta}(x) &= 1 \end{aligned} \tag{50}$$

Pour chaque sous-état $x \in 2^{\Delta}$, la valeur $m_{\Delta}(x)$ est appelée la masse de confiance de x affectée sur Δ. Quand nous constatons que la masse de confiance est dans un certain d'état, nous nous référons non seulement à la masse de confiance dans l'état, mais aussi à celles des sous-états. La définition de fonction de confiance ci-dessous va expliquer plus claire.

Fonction de confiance

Une fonction de confiance correspondant à m_{Δ} est une fonction $b: 2^{\Delta} \mapsto [0, 1]$ définie par :

$$b(x) = \sum_{y \sqsubseteq x} m_{\Delta}(y), \qquad avec\ x, y\ \in 2^{\Delta} \tag{51}$$

Fonction de méfiance

La définition de la méfiance est similaire à celle de la confiance, elle doit être interprétée comme un total des confiances des états qui sont fausses.

Une fonction de méfiance correspondant à m_{Δ} est une fonction d: $2^{\Delta} \mapsto [0, 1]$ définie par :

$$d(x) = \sum_{y \cap x = \emptyset} m_{\Delta}(y), \qquad avec\ x, y\ \in 2^{\Delta} \tag{52}$$

Fonction d'incertitude

Une fonction d'incertitude correspondant à m_{Δ} est une fonction u: $2^{\Delta} \mapsto [0, 1]$ définie par :

$$u(x) = \sum_{\substack{y \cap x = \emptyset \\ y \notin x}} m_{\Delta}(y), \qquad avec\ x, y\ \in 2^{\Delta} \tag{53}$$

Une formule relie les trois valeurs de confiance, méfiance et incertitude :

$$b(x) + d(x) + u(x) = 1 \tag{54}$$

Par ailleurs, nous obtenons plusieurs cas particuliers :

- $b = 1$: est équivalent à la logique binaire VRAI
- $d = 1$: est équivalent à la logique binaire FAUX
- $b + d = 1$: est équivalent à la probabilité traditionnelle
- $b + d = 0$: est équivalent au cas d'incertitude totale
- $b + d < 1$: exprime le niveau d'incertitude (c'est le cas qui nous concerne)

Atomicité relative

L'atomicité relative de *x* sur *y* est une fonction a: $2^{\Delta} \mapsto [0, 1]$ qui est définie par :

$$a(x/y) = \frac{|x \cap y|}{|y|} \qquad avec\ x, y\ \in 2^{\Delta} \tag{55}$$

L'atomicité relative d'un état *x* sur Δ est notée par $a(x/\Delta)$ ou simplement $a(x)$.

Probabilité d'espérance

Une fonction de la probabilité d'espérance correspondant à m_Δ est une fonction $E: 2^\Delta \mapsto [0,1]$ définie par :

$$E(x) = \sum_y m_\Delta(y) \times a(x/y) \qquad avec\ x, y\ \in\ 2^\Delta \tag{56}$$

Opinion subjective

Étant donné un ensemble d'états Δ contenant x et leur complément $\bar{x}$, supposant une affectation de masse de confiance m_Δ avec les fonctions de la confiance, la méfiance, l'incertitude et l'atomicité relative sur x dans Δ telles que $b(x)$, $d(x)$, $u(x)$ et $a(x)$. Une opinion sur x, notée ω_x :

$$\omega_x = \langle b(x), d(x), u(x), a(x) \rangle \tag{57}$$

La probabilité d'une opinion subjective est calculée par :

$$E_x = b(x) + a(x) \times u(x) \tag{58}$$

Loi de classement d'opinions

Étant donné 2 opinions ω_x et ω_y. Elles peuvent être classées par les critères suivants en respectant l'ordre de priorité :

- Plus la probabilité d'espérance est haute, plus l'opinion est préférée ;
- Plus l'incertitude est faible, plus cette opinion est meilleure ;
- La plus petite valeur de l'atomicité nous donne une meilleure opinion.

Nous avons présenté les concepts employés par la logique subjective. Nous allons maintenant aborder les méthodes permettant de modéliser une opinion subjective à partir des profils idée des utilisateurs.

4.4.3.2 Modélisation d'une opinion subjective

D'après la logique subjective, l'élément principal est une opinion. Nous avons besoin de formaliser une opinion selon la définition donnée ci-dessus. La difficulté majeure de l'application de la logique subjective est de trouver une méthode pour modéliser systématiquement les opinions qui seront utilisées comme paramètres d'entrée. Il existe deux méthodes :

Méthode statistique

Une opinion est modélisée en se basant sur la statistique d'épreuve. Nous supposons qu'un processus produit des conséquences, soit positives, soit négatives. Dans les exécutions antérieures du processus, elles ont produit *r* et *s* qui sont respectivement les conséquences

positives et négatives. Une opinion disant une conséquence positive est modélisée sous forme de $\omega = \langle b, d, u, a \rangle$ avec :

$$b = \frac{r}{r + s + 2} \tag{59}$$

$$d = \frac{s}{r + s + 2} \tag{60}$$

$$u = \frac{2}{r + s + 2} \tag{61}$$

$$a = atomicité \tag{62}$$

Enquête de l'utilisateur :

Cette section décrit un questionnaire [91] qui permet de guider l'utilisateur ou le concepteur à la modélisation de ses propres idées au moyen d'un format d'opinion subjective. Il faut respecter l'ordre des questions suivantes :

1. Est-ce que la proposition est exprimée clairement ?
    a. Oui : aller à 2
    b. Non : refaire 1
2. Avez-vous une opinion pour ou contre la proposition ?
    a. Oui : aller à 3
    b. Non : vous êtes dans le cas d'incertitude totale $b = 0$, $d = 0$, $u = 1$: aller à 10
3. Quel est le niveau de certitude de votre idée ?
   → donner une valeur $0 \leq x \leq 1$: aller à 4
4. Quel est le niveau d'opposition de votre idée ?
   → donner une valeur $0 \leq y \leq 1$: aller à 5
5. Quel est le niveau d'appui votre idée ?
   → donner une valeur $0 \leq z \leq 1$: aller à 6
6. Normalisation des résultats :
   $b = \frac{z}{z+y+(1-x)}$ $d = \frac{y}{z+y+(1-x)}$ $u = \frac{1-x}{z+y+(1-x)}$ →aller à 7
7. Quel est le nombre total des états (n) dans l'ensemble d'états ? → aller à 8
8. Combien d'états (m) sont apparus dans la proposition ? → aller à 9
9. $a = \frac{m}{n}$: → aller à 10
10. $\omega = \langle b, d, u, a \rangle$

Comparaison des 2 méthodes

La méthode statistique est basée sur les informations issues des exécutions précédentes. Au niveau statistique, elle est bonne ; mais il existe un inconvénient qui est la perception de l'utilisateur. En effet, l'idée est de modéliser l'opinion de l'utilisateur. Selon la méthode 1, elle ne considère pas le rôle de l'utilisateur. Tandis que la seconde méthode tient compte totalement de l'idée de l'utilisateur et ne considère pas les modèles ou les données extérieures

qui permettent à l'utilisateur d'estimer une partie de son idée. En nous inspirant de ces 2 méthodes, nous proposons une nouvelle approche qui réutilise leurs avantages.

Notre nouvelle méthode a besoin d'une base de données telle qu'une base de traces pour pouvoir calculer les valeurs statistiques. Nous allons donc présenter dans la section suivante comment appliquer la logique subjective à base de traces à la modélisation des opinions. Ces opinions sont explicitement des préférences de l'utilisateur dont nous avons besoin pour calculer le choix individuel de l'utilisateur.

4.4.3.3 Application de la logique subjective à base de traces à la modélisation des préférences de l'utilisateur

Cette section vise à décrire l'application de la logique subjective à la modélisation des opinions de l'utilisateur dans une application interactive à base de situations. À la fin d'une situation, l'utilisateur doit choisir une situation parmi un ensemble *Cand* de n situations candidates pour la prochaine exécution. À ce moment, l'utilisateur donne son avis ou sa propre idée sur chacune des situations candidates identifiées. Chaque avis représente une opinion subjective de l'utilisateur sur une situation candidate. Nous allons donc obtenir un ensemble de *n* opinions subjectives. Le classement de ces opinions nous donne la liste de priorités des situations candidates qui représente formellement la préférence de l'utilisateur.

Par exemple, si nous avons l'ensemble *Cand* = {Manger, Dormir, Jouer} de 3 situations candidates. L'utilisateur donne alors son avis sur chaque situation séparément et nous obtenons un ensemble de 3 opinions subjectives $0Sub = \{\omega_{Manger}, \omega_{Dormir}, \omega_{Jouer}\}$. Nous effectuons le classement de ces 3 opinions pour obtenir la liste de priorités suivante : $\omega_{Manger} \succ \omega_{Jouer} \succ \omega_{Dormir}$. Nous en déduisons une liste de priorité des situations candidates : Manger ≻ Jouer ≻ Dormir. Cette liste est un exemple des préférences de l'utilisateur.

Pour appliquer la logique subjective à la modélisation des préférences de l'utilisateur, il nous faut arriver à modéliser les avis subjectifs venant de l'utilisateur ou la modélisation des opinions subjectives. Comme nous l'avons indiqué dans la section 4.4.3.2, il s'agit de deux méthodes de modélisation. Notre proposition sera nommée « Logique Subjective à base de Traces ». Nous allons présenter la base de traces que nous utiliserons dans la logique subjective.

Base de traces utilisée pour la modélisation des préférences de l'utilisateur

Depuis la base de données de la section 4.1.3, nous n'extrayons les informations que depuis la catégorie d'Activités, ou autrement dit les traces d'activités de l'utilisateur lors des exécutions antérieures de l'application. L'idée de cette extraction est d'avoir des informations pour calculer les conséquences positives et négatives à partir des traces d'activités. En effet, nous allons regarder les traces contenant les informations suivantes : Après être sorti d'une situation, quelles sont les valeurs de l'accomplissement des critères ? Est-ce que ces valeurs ont changé ? Quelle situation l'utilisateur a choisi d'exécuter ensuite ?

Par exemple, une suite des choix d'un utilisateur pourrait être la suivante :

$\langle\{Manger\}, \{Santé = 0{,}8; Socialisation = 0{,}3; Maturité = 1\}\rangle \rightarrow \langle\{Dormir\},\ \{Santé = 0{,}9; Socialisation = 0{,}1; Maturité = 1\}\rangle\ \rightarrow$

$\langle\{Jouer\}, \{Santé = 0{,}5; Socialisation = 0{,}35; Maturité = 1{,}1\}\rangle \rightarrow \langle\{Socialiser\}, \{Santé = 0{,}4; Socialisation = 0{,}6; Maturité = 1{,}25\}\rangle$

L'exemple ci-dessus décrit une partie d'une exécution qui a été fait par un utilisateur. À la fin de la situation Manger, les 3 critères atteignent 0,8 pour Santé, 0,3 pour Socialisation, 1 pour Maturité. Ensuite, il a exécuté la situation Dormir. Après avoir fini cette situation, les valeurs des 3 critères sont changées comme suit : Santé est égale à 0,9 ; Socialisation est égale à 0,1 et Maturité est égale à 1, etc. Ce sont une partie des traces premières telles que nous l'avons défini dans le Chapitre 3. Pourtant, pour bien les adapter à la logique subjective et pour faciliter le calcul des conséquences positives et négatives à partir des traces, nous avons effectué une phase de transformation de traces premières afin d'extraire ce que nous voulons pour appliquer la logique subjective à base de traces.

À partir d'une suite successive de choix comme ceux de l'exemple au-dessus, nous effectuons une transformation en sous-enregistrements respectant le format :

$\langle S_{prev}, C_{prev}, S_{next}, C_{next}\rangle$.

$C_{prev} = \{C_{prev}(m)\}$ représente l'ensemble des valeurs des m critères après avoir fini la situation S_{prev}. S_{next} représente la situation exécutée après la situation S_{prev}. $C_{next} = \{C_{next}(m)\}$ représente l'ensemble des valeurs des m critères obtenues après avoir fini la situation S_{next}.

Par exemple, nous pouvons extraire les 3 enregistrements suivants :

$$\langle Manger, \{Santé = 0{,}8; Socialisation = 0{,}3; Maturité = 1\}, Dormir, \{Santé = 0{,}9; Socialisation = 0{,}1; Maturité = 1\}\rangle$$

$$\langle Dormir, \{Santé = 0{,}9; Socialisation = 0{,}1; Maturité = 1\}, Jouer, \{Santé = 0{,}5; Socialisation = 0{,}35; Maturité = 1{,}1\}\rangle$$

$$\langle Jouer, \{Santé = 0{,}5; Socialisation = 0{,}35; Maturité = 1{,}1\}, Socialiser, \{Santé = 0{,}4; Socialisation = 0{,}6; Maturité = 1{,}25\}\rangle$$

Une fois que nous avons obtenu une base des traces transformées dont chaque enregistrement a le format ci-dessus, nous appliquons l'analyse des traces à la logique subjective.

Logique subjective à base de traces

Notre méthode hérite des avantages des deux méthodes de modélisation mentionnées en 4.4.3.2 :

- Prise en compte de l'avis de l'utilisateur : puisque l'utilisateur est l'acteur principal qui interagit directement avec le système. Son idée sur le choix de la situation suivante a un rôle important dont nous devons tenir compte lors de la modélisation de son idée par une opinion subjective ;
- L'utilisation des informations des exécutions antérieures : les traces sont des sources d'informations pertinentes. Grâce à la méthode statistique, nous pourrons extraire à partir des traces des conséquences positives et négatives qui permettent d'améliorer la modélisation des opinions subjectives. Par conséquent, nous appliquons ce principe en intégrant le calcul à base de traces pour modéliser une opinion subjective.

Notre méthode de logique subjective à base de traces est synthétisée par la procédure suivante :

Entrée : La situation qui vient de finir *Sit*, l'ensemble de situations candidates *Cand*, l'idée de l'utilisateur $I = \{potential\_sit\}$

Procédure :

1. Est-ce que I est exprimée clairement ?
    a. Oui : aller à 2
    b. Non : refaire 1
2. Avez-vous un point de vue qui est pour ou contre la proposition I?
    a. Oui : aller à 3
    b. Non : vous êtes dans le cas d'incertitude totale $b = 0$, $d = 0$, $u = 1$
3. Extraire des transitions dans la base de traces où S_{prev} est égale à Sit

    Nous définissons *r* et *s* qui sont respectivement la conséquence positive et négative de la situation *Sit*. Pour chaque enregistrement dans la base de traces, nous calculons :

$$\epsilon = \sum_{h=1}^{m} w_h \times \left(C_{next}(h) - C_{prev}(h)\right)$$

    Si $\epsilon > 0$ alors $r = r + 1$
    Si non $s = s + 1$

4. Calculer x : le niveau d'évidence de la situation $potential\_sit$ selon les traces : $x = \frac{r}{r+s}$
5. Calculer y : demander à l'utilisateur combien il est d'accord pour choisir la situation $potential\_sit$, $0 \leq y \leq 1$
6. Calculer z : demander à l'utilisateur combien il est contre pour choisir la situation $potential\_sit$, $0 \leq z \leq 1$
7. Normalisation des résultats :
    $b = \frac{z}{z+y+(1-x)}$; $d = \frac{y}{z+y+(1-x)}$; $u = \frac{1-x}{z+y+(1-x)}$
8. Quel est le nombre total des états (n) dans l'ensemble d'états ?
9. Combien d'états (m) sont apparus dans la proposition ?
10. Calculer a → a = 1/le nombre des situations dans Cand

Sortie : L'opinion $\omega = \langle b, d, u, a \rangle$

Si l'ensemble de situations candidates *Cand* contient n situations, l'utilisateur doit donner n idées pour exprimer les n opinions subjectives. L'ensemble des opinions sera classé en une liste de priorité des situations candidates. Pour chaque opinion, nous calculons sa probabilité d'espérance E_ω selon la formule (55). Nous devrons ensuite appliquer la loi de classement des opinions subjectives présentée dans la section 4.4.3.1. Nous obtenons finalement une liste des situations candidates correspondant au classement des opinions subjectives.

Nous venons de présenter une approche de modélisation de la préférence de l'utilisateur en utilisant la logique subjective. Pourtant, il existe des inconvénients dans la détermination traditionnelle des opinions subjectives. Nous avons donc proposé une procédure qui combine les deux méthodes statistique et l'enquête de l'utilisateur en utilisant les traces pour résoudre le problème. Cette nouvelle procédure permet non seulement d'éviter les inconvénients analysés, mais aussi d'avoir une préférence de l'utilisateur. Le résultat sera ajouté comme choix de l'utilisateur dans notre mécanisme de décision.

4.4.4 AGREGATION DES DIFFERENTES CHOIX

Nous obtenons une liste de choix venant de différents décideurs. Comment faire le meilleur choix convenant à tous les décideurs est une question qui apparaît au bout de notre algorithme de décision multicritère. Nous allons aborder dans cette section notre solution d'agrégation pour combiner les choix obtenus en produisant un choix final.

4.4.4.1 Problème d'agrégation des préférences

Notre système de décision tient compte non seulement du choix du système, mais aussi du choix de l'utilisateur et le choix du concepteur. Chacun d'entre eux nous donne une liste de priorité des situations candidates qui permet de représenter la perception de choix individuel selon le point de vue correspondant. Idéalement, ces 3 listes de choix ont le même résultat. Dans ce cas, il est facile de trouver un consensus sur la meilleure situation : celle qui est située à la première place des listes. Cette situation obtient le même niveau d'importance de l'utilisateur, du concepteur et du système pour être exécutée à la prochaine étape de l'application. Cependant, si les listes des priorités sont différentes totalement ou partiellement, il nous faut faire un choix qui agrège les 3 préférences. Cela repose sur un problème d'agrégation des préférences.

Dans la littérature, il existe des méthodes permettant de résoudre ce type de problème. Par exemple, la méthode Delphi [92] est une méthode interactive et systématique qui vise à intégrer les informations des experts indépendants. L'avantage de cette méthode est de vérifier plusieurs fois pour ajuster les idées des experts jusqu'à ce qu'une idée de compromis soit trouvée. Cette méthode supporte un point de vue interactif avec les experts. C'est-à-dire, d'abord, les experts donnent les idées sur les alternatives puis Delphi va évaluer ces idées pour trouver une solution correspondante. Si elle ne trouve pas, elle va reboucler pour demander aux experts de modifier leurs idées. Dans ce cas, les experts peuvent donner de nouvelles évaluations pour les alternatives. Si nous insistons sur le côté interactif, Delphi émergera comme une technique correspondante. Par contre, dans notre cas il n'y a que l'utilisateur qui peut changer ses idées. En ce qui concerne le choix du système et du concepteur, nous ne pouvons pas changer car c'est des résultats fixes. Nous pouvons donc conclure que Delphi n'est pas applicable pour notre besoin d'agrégation des préférences.

Par ailleurs, la méthode AHP [93] est aussi utilisée pour résoudre le problème. C'est une méthode qui sert à la décision multicritère. Mais elle a besoin de pondérer l'importance relative des experts. C'est un peu difficile à faire parce que la plupart des évaluations sont très intuitives. Si nous arrivons à fixer ces poids, AHP pourra être une méthode d'agrégation efficace. Par contre, dans notre contexte, nous supposons au début que les différents experts ont la même importance.

Il existe aussi les deux autres techniques d'agrégation que sont : Borda [94] et Condorcet [95]. Elles demandent à ordonner les candidats (les alternatives) en avance. Si nous avons N décideurs sur *n* alternatives, chaque décideur doit effectuer séparément le classement sur n alternatives. Nous pouvons puis appliquer les techniques Borda ou Condorcet pour agréger. Cela correspond à notre cas. Nous avons déjà des listes de priorité des situations candidates disponibles. Pourtant, selon [96]: « *l'agrégation d'ordres totaux (les classements individuels) par cette méthode ne donne pas toujours un ordre total* ». Le résultat de Condorcet est une solution partielle, il ne représente pas une solution pour choisir un choix en général. Alors que

la méthode Borda nous donne un résultat qui est un ordre total. En effet, la méthode Borda présente des inconvénients dans le cas où dans chaque liste de priorité (appelée rangement), il y a plusieurs votants.

Par exemple, nous avons 3 rangements qui sont des listes de priorité des alternatives comme suit :

Rangement 1 : a ≻ b ≻ c ≻ d : 7 votants
Rangement 2 : a ≻ c ≻ d ≻ c : 3 votants
Rangement 3 : c ≻ d ≻ a ≻ b : 5 votants

S'il y a plusieurs votants pour chaque rangement comme l'exemple ci-dessus, le résultat de cette méthode n'est pas une solution optimale (le choix final peut ne pas celui qui est obtenu la plupart des votants). Dans notre cas, pour chaque rangement il n'y a qu'un seul votant, c'est le choix individuel de l'utilisateur, du concepteur et du système. Donc cette méthode nous semble applicable.

Selon nos analyses, nous trouvons que Borda est une méthode qui convient dans notre cas. Dans la suite du document, nous allons décrire la méthode de Borda pour agréger les différentes préférences des experts.

4.4.4.2 Utilisation de la méthode Borda à l'agrégation des préférences

Nous appelons que nous disposons de 3 types de choix individuels : choix de l'utilisateur, choix du système, choix du concepteur. Chaque choix est calculé parmi n situations candidates pour obtenir des préférences particulières. S'il y a n situations candidates, pour chaque liste de choix individuel, nous attribuons *n* points à la situation placé en tête, *n – 1* points à la situation suivante, jusqu'à la dernière à qui nous attribuons 1 point. Puis nous calculons la somme des points de chaque situation dans chaque liste de priorité, ce qui donne les scores de Borda des situations définies. Le score Borda d'une situation *i*, noté *Borda(i)*, dans un ensemble *Cand* de *n* situations candidates est calculé par :

$$Borda(i) = score_i(utilisateur) + score_i(système) + score_i(concepteur) \quad (63)$$

avec $score_i(utilisateur)$, $score_i(système)$ et $score_i(concepteur)$ respectivement le score de la situation i dans le choix de l'utilisateur, du système et du concepteur.

Le meilleur choix est alors la situation ayant le score de Borda le plus élevé. Par ailleurs, nous pourrons obtenir une liste de préférences collective en triant les situations selon les scores décroissants. Si le résultat obtenu est ex-acquo, c'est l'utilisateur qui doit choisir en fonction de son intuition. Mais ce cas est très rare dans les applications réelles.

Nous présentons un exemple pour illustrer le principe de Borda : nous avons 4 situations candidates (Manger, Dormir, Jouer, Éduquer). Chaque expert va donner sa préférence comme suit :

Choix de l'utilisateur : Dormir ≻ Éduquer ≻ Manger ≻ Jouer
Choix du système : Éduquer ≻ Jouer ≻ Dormir ≻ Manger
Choix du concepteur : Jouer ≻ Éduquer ≻ Manger ≻ Dormir

Les scores de Borda sont calculés :

$$Borda\ (Manger) = 2 + 1 + 2 = 5$$
$$Borda\ (Dormir) = 4 + 2 + 1 = 7$$
$$Borda\ (Jouer) = 1 + 3 + 4 = 8$$
$$Borda\ (Éduquer) = 3 + 4 + 3 = 10$$

Nous pouvons donc déterminer la liste du choix collectif obtenu : Éduquer ≻ Jouer ≻ Dormir ≻ Manger. La solution cherchée est la situation éduquer, qui est le résultat final de notre processus d'aide à la décision multicritère. Cette situation sera suggérée à l'utilisateur pour la prochaine exécution de l'application.

4.4.5 PUBLICATIONS SCIENTIFIQUES

Cette contribution a donné lieu à un article dans une conférence internationale [97] (choix de l'utilisateur) et deux articles dans les conférences nationales [98], [99] (choix du système).

Choix de l'utilisateur :

Titre de l'article : Application of Trace-Based Subjective Logic to User Preferences Modeling.

Conférence : *20th International Conference on Logic Programming, Artificial Intelligence and Reasoning*, 2015.

Cet article décrit comment modéliser la préférence de choix d'un utilisateur en appliquant notre méthode proposée, Logique Subjective à base de traces, qui utilise la logique subjective et les traces. Cette méthode permet à l'utilisateur d'exprimer ses idées dans le cas incertain, et d'utiliser les traces pour calculer son choix final.

Choix du système :

- Article 1 :
  - Titre : Aide à la décision multicritère à base de traces avec PROMETHEE II ;
  - Conférence : *16ème Conférence ROADEF*, 2015.
- Article 2 :
  - Titre : Personnalisation interactive de parcours pédagogiques par une méthode de décision multicritère à base de traces ;
  - Conférence : *7ème Conférence sur EIAH,* 2015.

Les articles visent à décrire la méthode PROMETHEE II à base de traces. Elle s'appuie sur la méthode PROMETHEE II que nous avons amélioré avec une analyse des traces. Nous pouvons ainsi calculer automatiquement les évaluations des alternatives selon des différents critères.

4.5 SYNTHESE

Dans ce chapitre, la décision multicritère et l'analyse de traces en tant que mots-clefs des travaux de recherche de notre thèse ont été définis de manière formelle et approfondie. Nous avons proposé un système d'aide à la décision multicritère à base de traces en plusieurs étapes.

Chaque étape est une contribution de notre thèse qui vise à résoudre les inconvénients des systèmes de décision multicritère traditionnels dans la littérature. Nous avons introduit des algorithmes à base de traces pour mettre en œuvre automatiquement les phases de pondération de critères, de détermination des situations candidates. Concernant la contribution de prise de décision, nous avons proposé de combiner les différents choix qui sont les résultats de décision de l'utilisateur, du système et du concepteur. Le résultat obtenu est celui qui satisfait le point de vue de ces trois acteurs dans l'application. En outre, chaque choix individuel est réalisé séparément avec les approches : PROMETHEE II à base de traces (traitant le choix du système), Logique Subjective à base de traces (modélisant le choix de l'utilisateur), des mesures de correspondance et d'expérience (représentant le choix du concepteur).

Toutes les contributions ont donné lieu à un article dans une revue internationale [84], trois articles dans des conférences internationales [83], [85], [97] et deux articles dans des conférences nationales [98], [99].

Dans le chapitre 5 suivant, nous présentons l'étude de cas Tamagotchi que nous avons utilisée pour tester et évaluer notre système d'aide à la décision multicritère à base de traces.

Partie 3 : Application et Evaluation

CHAPITRE 5 JEU DU TAMAGOTCHI

SOMMAIRE

Nous avons présenté tout au long dans ce manuscrit l'adaptation de l'exécution d'une application interactive. Ce chapitre présente le cas d'étude sur lequel nous allons valider toutes nos contributions. Par conséquent, la première section de ce chapitre vise à identifier les objectifs que nous voulons démontrer.

Ensuite, nous décrivons notre cas d'étude qui est le jeu du Tamagotchi. C'est un jeu interactif dans lequel le joueur doit soigner l'acteur principal, Tamagotchi, animal virtuel que le joueur doit soigner, nourrir, faire dormir, faire jouer… Chaque tranche de la vie du Tamagotchi est transcrite en situations. La description en détail de ces situations sera abordée dans la suite du chapitre.

5.1 OBJECTIF DE L'ETUDE DE CAS

Notre premier objectif est de vérifier si notre système d'aide à la décision dans une application interactive fonctionne. Autrement dit, s'il arrive à sélectionner une solution pertinente et optimale pour permettre la suite de l'exécution de l'application. De plus, cette étude de cas doit nous permettre de valoriser notre deuxième objectif de la thèse concernant la gestion de traces. Pour cela, nous allons récupérer les traces générées par l'utilisateur et le système pendant leurs interactions. Ces traces seront exploitées dans les mécanismes de décision que nous avons proposés pour répondre au premier objectif de la thèse. Afin de vérifier si cette étude de cas est pertinente au regard des problématiques que nous traitons, il nous faut identifier les problèmes soulevés dans nos travaux que nous voulons illustrer. Il existe deux grands thèmes :

- Aide à la décision : ce travail vise à montrer comment choisir une solution pertinente pour la prochaine exécution de l'application. Dans cette problématique, nous avons 3 sous-problèmes qui sont :
    - Pondération des critères (objectifs) ;
    - Détermination des situations candidates pour décider ;
    - Décision : choisir une situation optimale parmi les candidates qui ont été sélectionnées dans la phase précédente. Dans cette partie, non seulement nous nous concentrons sur les mécanismes de décision mais aussi il faut arriver à expliquer pourquoi nous choisissons cette solution.
- Gestion de traces :
    - Récupération des traces pendant l'exécution de l'application ;
    - Déterminer quelles sont les traces pertinentes pour l'aide à la décision.

Pour construire un cas d'étude qui réponde à tous les thèmes listés ci-dessus, il nous faut concevoir un exemple d'application interactive répondant aux contraintes suivantes :

- Une application pouvant être décomposée en situations avec un nombre réduit mais suffisant de situations ; un nombre trop important de situations nécessitera un trop grand temps de calcul et un nombre trop bas de situations ne serait pas significatif pour illustrer nos propositions ;
- Une application où il y aura assez d'interactions entre les différents acteurs afin d'avoir suffisamment de traces générées pendant l'exécution de l'application à exploiter dans les mécanismes de décision. Nous souhaitons avoir une application où nous pourrions récupérer toutes les traces des interactions ou des activités de l'utilisateur sans pour autant nous retrouver avec trop grande base de traces.

Nous avons choisi comme cas d'étude une variante du jeu de Tamagotchi qui répond à l'ensemble de nos contraintes exposées ci dessus. Ce jeu décrit simplement la vie d'un animal virtuel. L'utilisateur joue au Tamagotchi en effectuant les actions qui servent à maintenir le Tamagotchi en vie. Il y a des interactions entre l'utilisateur et son Tamagotchi ainsi que des interactions entre les différents Tamagotchis de l'environnement de l'application.

5.2 ÉTUDE DE CAS : JEU TAMAGOTCHI

Nous décrivons dans les paragraphes qui suivent les principes du jeu Tamagotchi que nous avons adoptés pour notre cas d'étude. L'objet de cette réalisation n'est pas de proposer un jeu complet et compliqué, mais d'avoir un outil qui nous permettra d'illustrer les contributions de nos travaux.

5.2.1 DESCRIPTION DU JEU

Nous considérons la vie du Tamagotchi à partir du début : Tamagotchi est un œuf au départ et l'utilisateur doit le faire couver afin qu'il éclose. À partir de ce moment, lorsque le Tamagotchi a besoin de quelque chose, un petit signal va apparaître pour avertir l'utilisateur afin qu'il intervienne. L'utilisateur a accès à l'état complet du Tamagotchi à travers divers indicateurs afin d'agir de façon appropriée, par exemple : donner à manger, jouer, entretenir ou éduquer... La Figure 33 est un exemple d'un scénario d'enchainement des situations du jeu Tamagotchi. Nous avons 7 situations qui seront enchainées tout au long de l'exécution du jeu Tamagotchi. Elles sont : Manger, Jouer, Dormir, Soigner, Entretenir, Socialiser et Éduquer. La description en détail de ces situations sera abordée dans la suite du chapitre.

De plus, le Tamagotchi a également besoin de relations sociales. Par conséquent, il faut que l'utilisateur socialise le Tamagotchi en le mettant en contact avec d'autres Tamagotchis virtuels gérés par le système en tant que PNJ (personnages non joueur). Ils peuvent jouer et interagir. Ceux qui ont interagi avec notre Tamagotchi peuvent ainsi devenir ses amis. Nous essayons de qualifier la relation entre les Tamagotchis en analysant leurs interactions. Nous disposons de deux types d'interactions : interaction positive et interaction négative. L'impact de ces interactions sur la relation sociale du Tamagotchi est modélisé comme suit : une interaction positive rend la relation plus amicale, une interaction négative la rend plus hostile. Par ailleurs, le Tamagotchi peut également rechercher un partenaire. Cela est aussi un type de socialisation du Tamagotchi.

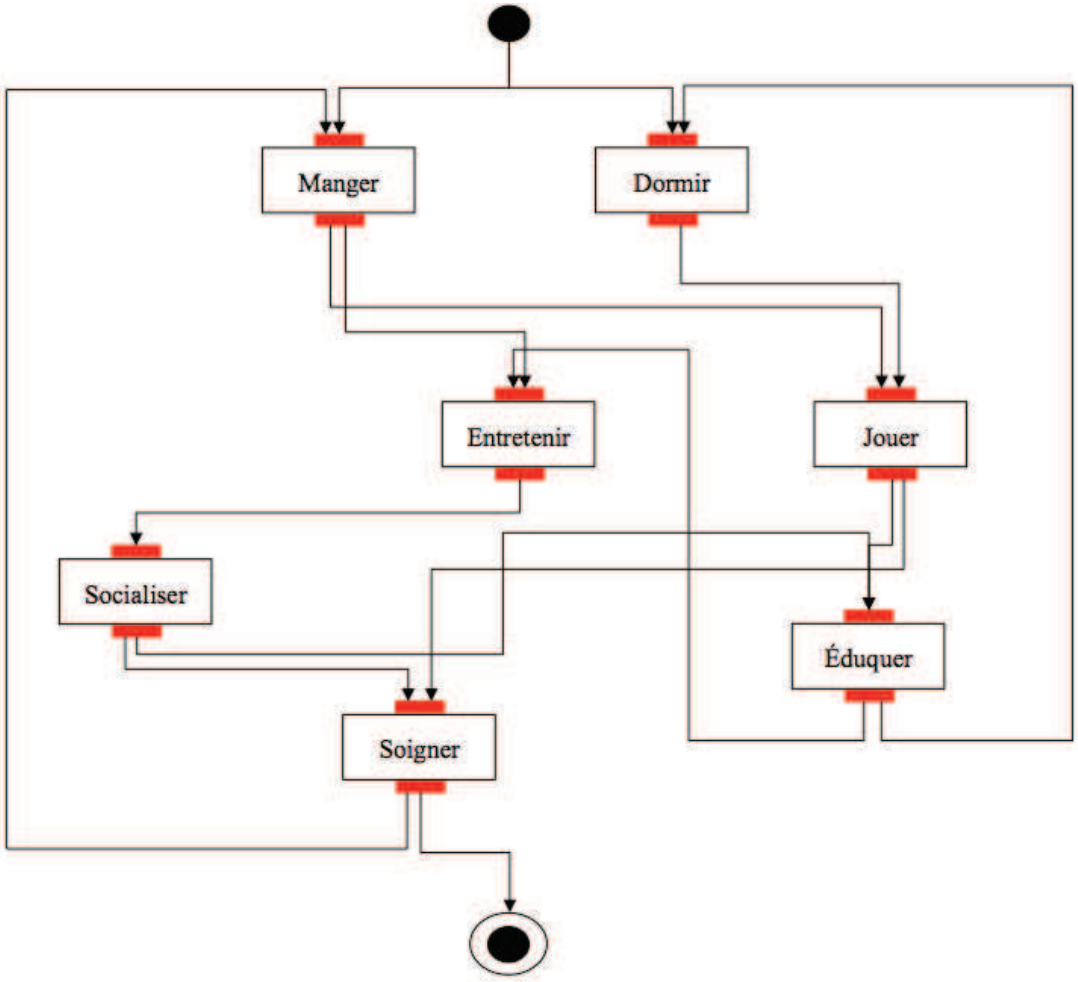

Figure 33. Graphe de situations dans le cas d'étude Tamagotchi

5.2.2 ANALYSE DES ATTRIBUTS UTILISES DANS LE JEU

L'état du Tamagotchi est organisé en plusieurs attributs qu'il faut définir. Chaque attribut de l'état du Tamagotchi est une propriété décrivant une partie de son été global. Les différents attributs se déclinent comme suit :

Tableau 14. Les attributs du Tamagotchi

Attribut	Valeur	Description
satiété (satiety)	[0, 1]	La satiété du Tamagotchi admet une valeur de 0 (il a faim) jusqu'à 1 (il a trop mangé). Si c'est 0, il a très faim donc s'il ne mange pas, il meurt. Si c'est 1, il a une indigestion, il devient malade et ça influe sur sa santé.
fatigue (tiredness)	[0, 1]	Très fatigué → Pas fatigué
ennui (boredom)	[0, 1]	Ennui maximum → Ennui minimum
soin (care)	[0, 1]	Cette propriété décrit le soin à accorder au Tamagotchi ; plus cette valeur augmente, moins il a besoin de soin.
amis (friendly)	Valeur entière	Le nombre total des amis du Tamagotchi
politesse (politesse)	[0, 1]	Impoli → poli

Nous disposons également de l'âge du Tamagotchi et de l'état de l'environnement qui ont un impact sur le déroulement du jeu :

- âge : une valeur entière, elle est supérieure à 0.
- propreté : une valeur variée de 0 (environnement pas propre) à 1 (environnement très propre).
- disponibilité : booléen (vrai/faux), elle décrit la disponibilité du Tamagotchi dans un instant observé.

5.2.3 OBJECTIF DU JEU

Nous définissons 3 critères visant à évaluer l'objectif du jeu :

- Le premier critère vise à maintenir le Tamagotchi en vie jusqu'à la fin du jeu. Ce critère concerne sa santé.
- Le deuxième critère concerne la socialisation du Tamagotchi. Ce critère vise à faire évoluer son niveau social.
- Le dernier critère vise à éduquer le Tamagotchi et permet d'évaluer sa maturité.

La Figure 34 schématise les interactions existant entre les trois critères. La santé, en rouge, se rapporte aux 4 attributs suivants : la *satiété*, le *soin*, la *fatigue* et l'*ennui*. La socialisation se rapporte aux deux attributs : l'*ennui* et l'*ami*. La maturité, en vert, est associée à la *politesse*, la *fatigue* et le nombre d'*amis*. Comme on peut le voir, certains attributs se rapportent à plusieurs critères : l'*ennui* (santé, socialisation), la *fatigue* (santé, maturité), le nombre d'*amis* (socialisation, maturité).

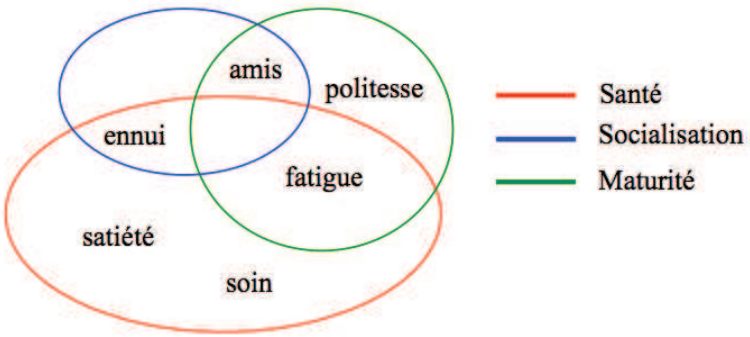

Figure 34. Croisement des attributs des critères

5.2.3.1 Critère de santé

Il faut maintenir le Tamagotchi en vie le plus longtemps possible. Comment déterminer si le Tamagotchi est encore en vie ou non ? Et le plus longtemps possible ? Comment traduire cet objectif en une fonction mathématique ?

Si un Tamagotchi est en vie, nous pouvons observer sa santé. Tant que sa santé est bonne, il pourra vivre. Sinon, le Tamagotchi peut mourir ou l'utilisateur devra le soigner. Afin d'évaluer la santé du Tamagotchi, nous proposons d'utiliser les quatre facteurs suivants : *satiété*, *soin* et *fatigue* et *ennui*.

Nous définissons une fonction de santé $f_{santé}$ élaborée à partir de la somme de quatre facteurs:

$$f_{santé} = \alpha_1 \times satiété + \alpha_2 \times soin + \alpha_3 \times fatigue + \alpha_4 \times ennui \quad (64)$$

Les paramètres $\alpha_1, \alpha_2, \alpha_3$ et α_4 sont des coefficients d'impact dans le calcul de la santé du Tamagotchi. Nous ne pouvons pas considérer les 4 facteurs avec un même niveau d'influence. Nous proposons d'ajouter 4 coefficients d'impact associés à chaque facteur. Pour identifier ces valeurs, nous nous basons sur l'âge du Tamagotchi car nous faisons l'hypothèse que l'impact de ces facteurs sur sa santé dépend de son âge. Si le Tamagotchi est jeune (moins de 4 jours), il est fragile et donc moins résistant ; les paramètres satiété, soin et fatigue seront vitaux. En revanche, s'il a plus de 7 jours, il sera plus résistant. Le Tableau 15 décrit les valeurs des coefficients d'impact correspondant à nos hypothèses. Ces valeurs sont définies par nous-même en fonction de nos intuitions. Elles pourront être ajustées en différentes valeurs par les différents utilisateurs.

Tableau 15. Coefficient d'impact de la santé du Tamagotchi en fonction de l'unité de vie (UT_v)[8]

Âge du Tamagotchi	α_1	α_2	α_3	α_4
1 - 3 UT_v	1	1	1	0,2
4 – 7 UT_v	0,9	0,9	1	0,2
8 – 12 UT_v	0,7	0,8	0,9	0,3
13 – 18 UT_v	0,4	0,5	0,7	0,5
19 – 30 UT_v	0,4	0,4	0,6	0,4
31 – 45 UT_v	0,5	0,5	0,7	0,3
46 – 60 UT_v	0,6	0,7	0,8	0,4
61 – 70 UT_v	0,7	0,8	0,9	0,4
> 70 UT_v	0,9	1	1	0,6

Si le Tamagotchi admis à 80 UT_v, il sera mort.

La fonction $f_{santé}$ est donnée par la relation (64). Nous définissons aussi une règle concernant la santé du Tamagotchi pour fixer le seuil de *bonne santé*. En effet, l'objectif est de maximiser la valeur de fonction $f_{santé}$ tout au long de la vie du Tamagotchi tout en respectant des contraintes sur les variables *satiété*. Pour cela, nous avons fixé des seuils déterminant l'état : *bonne santé*. Nous calculons alors les meilleures valeurs des attributs en calculant la moyenne des 4 coefficients d'impact. Puis, nous les multiplions afin d'identifier quels sont les seuils référencés. Après avoir effectué ces calculs, nous obtenons la santé du Tamagotchi doit être variée dans [0 ; 3,6]. La santé du Tamagotchi est bonne si cet indice dépasse la moitié de valeur maximale de la santé. Pourtant, si cet indice est élevé, cela implique que le Tamagotchi a trop de calories. Par conséquent, nous considérons que si la santé du Tamagotchi est dans l'intervalle $1,8 < f_{santé} < 3,0$, il est en bonne santé.

[8] L'unité de vie sera présentée dans la section « Description des règles du Tamagotchi »

5.2.3.2 Critère de socialisation

Pour évaluer la socialisation du Tamagotchi, nous proposons d'utiliser les deux attributs : *amis* et *ennui* parce qu'elles influencent directement le réseau social. Par exemple, s'il s'ennuie, il aura besoin d'amis pour s'amuser. Il doit interagir avec d'autres Tamagotchis et essayer de devenir amis. Afin de déterminer si deux Tamagotchis sont amis, nous nous basons sur leurs relations. Nous supposons qu'il existe trois relations possibles entre deux Tamagotchis : relation neutre, relation positive, relation négative. Au début, la relation entre deux Tamagotchis est une relation neutre. Nous reprenons alors des règles de relation comme indiqué dans le cas d'étude de thèse de Delmas [8] comme suit :

Tableau 16. Règles de relation en fonction des interactions

	Relation positive (R+)	Relation négative (R-)	Relation neutre (Rn)
Interaction positive	R+	R+	R+
Interaction négative	R-	R-	R-

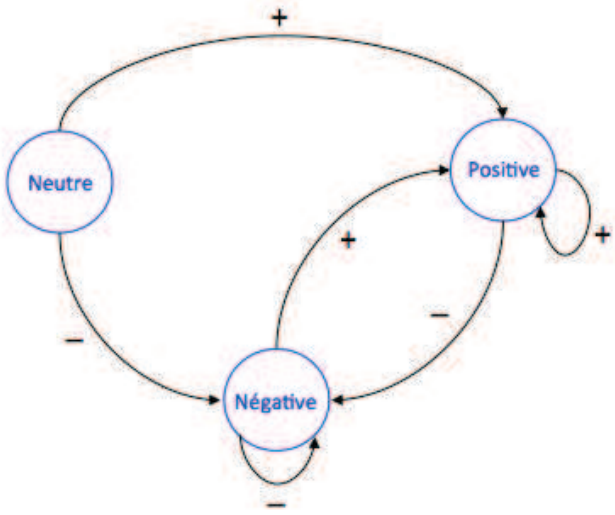

Figure 35. La transition des relations sociales par rapport aux interactions

Nous introduisons la mesure suivante.

$$f_{socialisation} = ennui \times amis \tag{65}$$

Nous trouvons que plus le nombre d'amis du Tamagotchi augmente, plus son niveau de socialisation augmente. Par ailleurs, plus l'ennui est proche que 1, plus le niveau de socialisation augmente. Ainsi la valeur du $f_{socialisation}$ représente la socialisation du Tamagotchi, il faut donc maximiser $f_{socialisation}$ pour atteindre ce critère. Dans ce cas, si la valeur de la socialisation du Tamagotchi est supérieure à 2, nous constaterons que ce critère est satisfait.

5.2.3.3 Critère de maturité

Le troisième critère vise à l'évaluation de la maturité du Tamagotchi. Il combine les trois attributs suivants : *amis*, *fatigue*, *politesse*.

Ainsi les règles de construction de cette fonction peuvent s'appuyer sur les cas suivants. Plus le Tamagotchi est fatigué, plus il fera des actions impolies. Par ailleurs, le nombre d'amis influence aussi la maturité du Tamagotchi car plus le nombre d'amis augmente, plus il est fatigué. Donc, il pourra faire des mauvaises choses ou des actions impolies avec les autres ce qui diminuera sa maturité. Par contre, il peut faire des actions polies même s'il est fatigué, dans ce cas, il peut obtenir une bonne valeur de maturité. S'il n'est pas trop fatigué mais il fait des actions impolies, la maturité diminue rapidement. Nous proposons de calculer la maturité $f_{maturité}$ comme suit :

$$f_{maturité} += fatigue \times amis \times politesse \tag{66}$$

Concernant ce critère, nous disons que si la valeur de la maturité est supérieure à 2, le Tamagotchi sera mature. Le critère est satisfait.

Les trois critères définis, santé, socialisation et maturité, représentent les objectifs globaux de l'application. Ils doivent être pris en compte par l'utilisateur afin d'atteindre les objectifs du jeu. Cependant, quand l'utilisateur cherche à satisfaire ces critères, il doit également atteindre des objectifs locaux que nous appellerons des *post-conditions*. Nous allons les définir dans chaque tranche de la vie du Tamagotchi dans la partie suivante.

5.3 SCENARISATION LE JEU DU TAMAGOTCHI EN SITUATIONS

Toute la vie du Tamagotchi est découpée en situations. En effet, chaque tranche de la vie du Tamagotchi peut être vue comme une situation contextualisée où le tamagotchi voit évoluer un ou plusieurs paramètres de son état.

5.3.1 DESCRIPTION DES SITUATIONS

Au début du jeu, l'utilisateur arrive à l'étape initiale, Naître. Ensuite, nous avons conçu 8 situations qui sont les activités que l'utilisateur doit réaliser avec le Tamagotchi. En effet, certaines actions mèneront à sa mort (par exemple, s'il n'est pas bien soigné, ou s'il est gravement malade).

- Manger (feeding) : donner à manger ou à boire au Tamagotchi ;
- Entretenir (cleaning) : nettoyer l'environnement où le Tamagotchi vit ou l'entretenir ;
- Jouer (playing) : permettre au Tamagotchi de jouer au jeu choisi pour distraire le Tamagotchi ;
- Soigner (healing) : si le Tamagotchi est malade, il a besoin d'être soigné ;
- Dormir (sleeping) : il faut faire dormir le Tamagotchi quand il est fatigué ;

- Socialiser (socializing) : aider le Tamagotchi à avoir plusieurs amis ; on le met en contact avec les autres Tamagotchi. L'observation des interactions au cours de la socialisation permet de constater l'évolution de ses amis ;
- Éduquer (educating) : si le Tamagotchi fait des bêtises, l'utilisateur va l'éduquer.

Le tableau suivant est un récapitulatif des pré-conditions, des post-conditions de ces situations.

Tableau 17. Récapitulatif des situations dans la vie du Tamagotchi

Situation	Pré-conditions	Post-conditions
Manger	*disponibilité = vrai* *satiété < 0,7*	*satiété > 0,1*
Entretenir	*disponibilité = vrai* *âge > 6 jours* *propreté < 0,5*	*propreté > 0,5*
Jouer	*disponibilité = vrai* *âge > 3 jours* *ennui != 1*	*ennui != 0*
Soigner	*soin < 0,2*	*soin > 0,2*
Dormir	*fatigue != 1*	*fatigue != 0*
Socialiser	*disponibilité = vrai* *âge > 6 jours* *amis > 0* *ennui != 1*	*ennui != 0* *amis = 0*
Éduquer	*disponibilité = vrai* *politesse < 0,5* *âge > 3 jours*	*politesse > 0,5*

Nous avons donc présenté la description en détail des situations dans le cas d'étude Tamagotchi. L'exécution du jeu est faite par l'enchainement des situations qui doit avoir des règles de fonctionnement permettant de mettre à jour les attributs du Tamagotchi en temps réel. La section suivante va nous mentionner les règles définies du jeu Tamagotchi.

5.3.2 DESCRIPTION DES REGLES DU JEU TAMAGOTCHI

Quand le joueur est connecté au jeu, le système doit envoyer un message indiquant que le jeu va commencer en définissant quelques valeurs initiales. Nous définissons des unités suivantes pour concevoir les règles du jeu :

- Unité de vie du Tamagotchi : UT_v
- Unité de temps : UT_t
- Unité d'aliments : UT_a

L'âge du Tamagotchi est calculé en fonction d'unité de vie. Nous allons considérer l'exécution du jeu en l'unité de temps. Nous définissons qu'une unité de vie du Tamagotchi équivaut aux A unités de temps : 1 $UT_v = A.UT_t$ (la valeur de A est définie par l'utilisateur).

Les coefficients d'augmentation (incréments) et diminution (décréments) de chaque attribut dépend du l'impact du fonctionnement de chaque situation. C'est-à-dire, dans le cas normal, les valeurs augment ou diminuent selon des petits impacts. Au contrairement, si elles sont dans

le cas spécial, il faut utiliser des grands impacts. Nous définissons α étant un coefficient de petit impact, β étant celui de grand impact, sous condition que $\alpha < \beta$.

Nous allons présenter dans la suite les règles dans chaque situation.

- Manger : La valeur de satiété change en fonction du nombre d'unités d'aliments. Nous considérons que la situation Manger apportera un grand impact à l'attribut satiété et propreté, donc le coefficient β est utilisé pour recalculer la valeur de la satiété et de la propreté. Pourtant, cette situation ne donne qu'un petit impact au ennui, fatigue et soin.

$$satiété = satiété + \beta_{\text{satiété}}\, UT_a$$
$$propreté = propreté - \beta_{\text{propreté}}$$
$$ennui = ennui + \alpha_{\text{ennui}}\, UT_t$$
$$fatigue = fatigue + \alpha_{\text{fatigue}}\, UT_t$$
$$soin = soin - \alpha_{\text{soin}}\, UT_t$$

  Quand la satiété passe à 1, l'annoncer au joueur.

- Entretenir : nous observons la valeur de l'attribut propreté. Cette situation a un grand impact à la propreté du Tamagotchi. Cependant, la fatigue n'augmente pas aussi vite que ci-dessus, pour chaque unité de temps, elle augmente au fur et à mesure → mise à jour :

$$propreté = propreté + \beta_{\text{propreté}}\, UT_t$$
$$fatigue = fatigue + \alpha_{\text{fatigue}}\, UT_t$$
$$satiété = satiété - \alpha_{\text{satiété}}\, UT_t$$
$$ennui = ennui + \alpha_{\text{ennui}}\, UT_t$$
$$soin = soin - \alpha_{\text{soin}}\, UT_t$$

- Jouer : Jouer a un grand impact sur l'ennui qui diminue, un grand impact sur la fatigue qui augmente quand le Tamagotchi s'amuse, un petit impact sur la satiété car jouer donne faim, un petit impact sur la propriété soin qui diminue en jouant et un petit impact sur la propreté car la propreté de l'environnement diminue en jouant. L'évolution des attributs est donnée par les formules suivantes :

$$ennui = ennui - \beta_{\text{ennui}}\, UTt$$
$$fatigue = fatigue + \beta_{\text{fatigue}}\, UTt$$
$$satiété = satiété - \alpha_{\text{satiété}}\, UTt$$
$$soin = soin - \alpha_{\text{soin}}\, UTt$$
$$propreté = propreté - \alpha_{\text{propreté}}$$

- Soigner : Mise à jour :

$$soin = soin + \beta_{\text{soin}}\, UT_t$$
$$ennui = ennui + \alpha_{\text{ennui}}\, UT_t$$
$$fatigue = fatigue + \alpha_{\text{fatigue}}\, UT_t$$
$$satiété = satiété - \alpha_{\text{satiété}}\, UT_t$$

- Dormir : compter le temps du sommeil (un sommeil est environ 5 à 7 unité de temps). Mise à jour :

$fatigue = fatigue - \beta_{\text{fatigue}}\ UT_t$
$satiété = satiété - 1/2.\alpha_{\text{satiété}}\ UT_t$ (pendant son sommeil, le Tamagotchi ne consomme pas beaucoup de calories, ce qui retarde sa faim)

- Socialiser : il existe deux types d'interactions : positive (+) et négative (-).

$fatigue = fatigue + \alpha_{\text{fatigue}}\ UT_t$
$satiété = satiété - \alpha_{\text{satiété}}\ UT_t$
$soin = soin - \alpha_{\text{soin}}\ UT_t$

Nous considérons chaque Tamagotchi PNJ[9] géré par le système et nous nous basons sur la relation et l'interaction entre le PNJ et le Tamagotchi du joueur.

Si la relation est positive :
Interaction positive :
$ennui = ennui - \beta_{ennui}\ UT_t$
$amis = amis + 1$
Interaction négative :
$ennui = ennui + \alpha_{ennui}\ UT_t$
$amis = amis - 1$
Si la relation est négative :
Interaction positive :
$ennui = ennui - \alpha_{ennui}\ UT_t$
$amis = amis + 1$
Interaction négative :
$ennui = ennui + \beta_{ennui}\ UT_t$
$amis = amis - 1$
Si la relation est neutre :
Interaction positive :
$ennui = ennui - \alpha_{ennui}\ UT_t$
$amis = amis + 1$
Interaction négative :
$ennui = ennui + \alpha_{ennui}\ UT_t$
$amis = amis - 1$

- Éduquer : si le Tamagotchi fait des actions impolies, il faut éduquer le Tamagotchi. Cette situation va modifier la valeur de politesse.

$politesse = politesse + \alpha_{politesse}\ UTt$
$satiété = satiété - \alpha_{satiété}\ UTt$
$ennui = ennui + \alpha_{ennui}\ UTt$
$fatigue = fatigue + \alpha_{fatigue}\ UTt$

Si le Tamagotchi est mort, le jeu est fini. Nous mettons à jour :

$satiété = 0, ennui = 0, fatigue = 0, soin = 1, amis = 0, politesse = 0$

[9] Personnage Non Joueur

5.4 ARCHITECTURE DU TAMAGOTCHI INTERACTIF ADAPTATIF

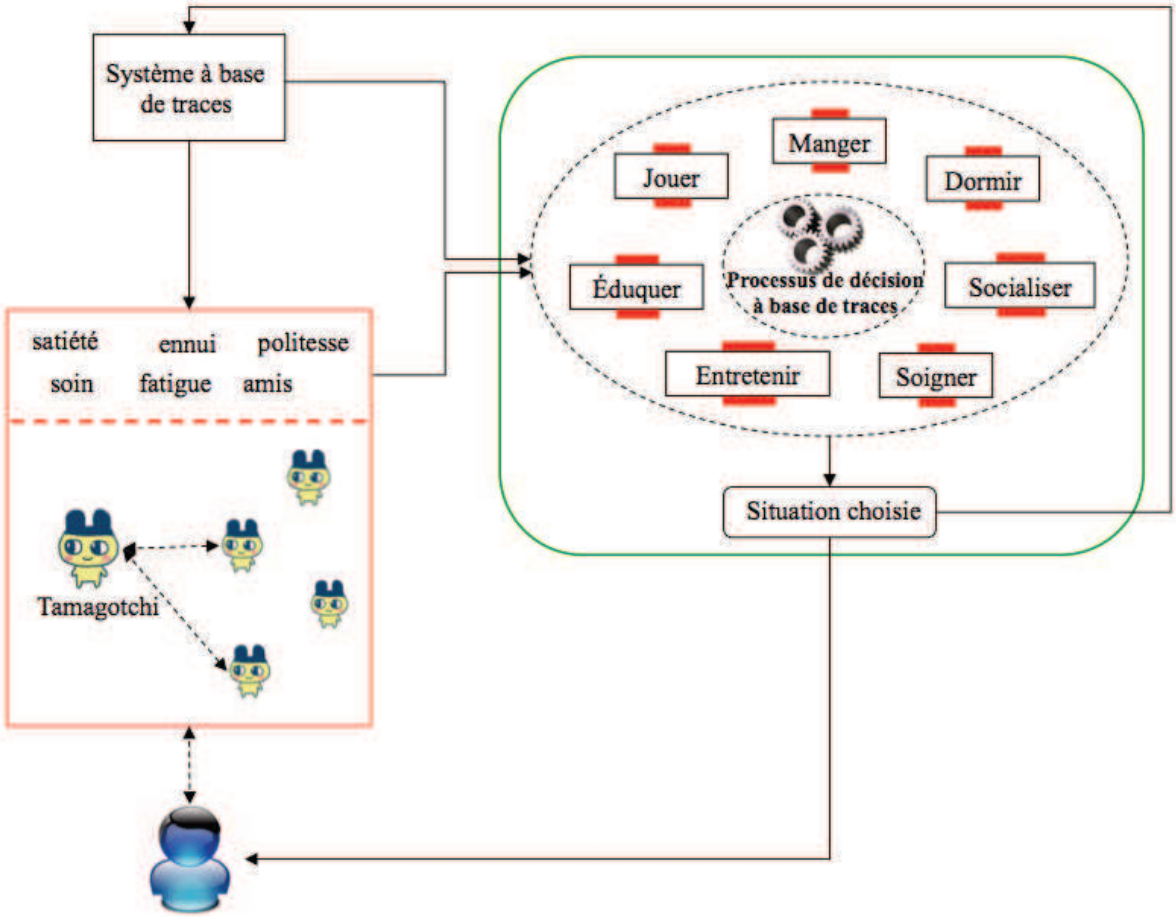

Figure 36. Architecture du système Tamagotchi

Dans la Figure 36, nous trouvons l'architecture générale du jeu Tamagotchi. Nous pouvons retrouver les composants comme la Figure 4 qui décrit l'architecture d'une application interactive à exécution adaptative. Le système qui est entouré en vert est un agent intelligent qui fonctionne lors de déroulement du système Tamagotchi. Il permet de choisir une situation à exécuter en appliquant notre processus de décision multicritère à base de traces. Il est relatif à notre système à base de traces.

En outre, nous avons besoin d'un système à base de traces qui sert à observer les attributs du Tamagotchi ainsi que l'état du système dans la section 5.2.2. Il doit récupérer les valeurs réelles pendant l'exécution du jeu. Les attributs changent à chaque instant du jeu, le système à base de traces mémorise le changement des valeurs des attributs. La collecte, le stockage et la transformation des traces seront effectués par ce système à base de traces intégré lors de la conception du système Tamagotchi.

L'application Tamagotchi est celle qui est entouré en rouge. Il s'agit de l'interface du Tamagotchi et elle permet au l'utilisateur d'interagir avec l'application par les interactions (Annexe B). L'ensemble de tous les modules ci-dessus forme un système Tamagotchi interactif et adaptatif.

5.5 SYNTHESE

Nous avons présenté dans ce chapitre l'étude de cas du Tamagotchi que nous allons utiliser pour tester nos contributions. Il s'agit d'un exemple simple répondant à nos hypothèses de scénarisation en situations. L'exécution de cette application est interactive, elle permet d'avoir des interactions entre l'utilisateur et le Tamagotchi ainsi que l'interaction entre les Tamagotchis. Nous avons défini les règles d'exécution, les formules pour calculer les critères du jeu. L'ensemble de ces éléments nous donne un jeu interactif adaptatif qui sera une nouvelle version du jeu Tamagotchi.

Dans le chapitre suivant, nous illustrons les différentes contributions de nos travaux sur ce cas d'étude.

CHAPITRE 6 MISE EN ŒUVRE DE LA DECISION MULTICRITERE A BASE DE TRACES DANS LE CAS D'ETUDE TAMAGOTCHI

SOMMAIRE

Dans ce chapitre, nous montrons comment nos propositions peuvent être appliquées afin de résoudre les problématiques posées dans le Chapitre 1. Nous expliquons en détails comment adapter une exécution par une décision multicritère en gérant les traces générées dans ce système. Toutes nos contributions seront décrites par un exemple dans le cas d'étude Tamagotchi et l'évaluation de nos propositions sera présentée dans la suite de chaque exemple.

6.1 INTRODUCTION

D'abord, nous présentons le diagramme de classe de l'étude de cas Tamagotchi en tenant compte des règles présentées dans le Chapitre 5 et les contributions de traces et de prise de décision multicritère dans les deux Chapitre 3 et Chapitre 4.

Nous nous intéressons ensuite à la mise en œuvre de la gestion de traces dans le système. Nous définissons un système à base de traces qui permet d'observer, de collecter et de transformer les traces générées pendant l'exécution du jeu Tamagotchi. Nous obtenons à la fin de cette étape une base de traces modélisées nécessaire à la décision multicritère.

Puis, nous démontrons comment nos contributions concernant la décision multicritère sont appliquées dans ce cas concret. Nous cherchons à valider les 3 points suivants :

- La pondération des critères,
- La détermination des alternatives,
- La prise de décision multicritère.

6.2 MISE EN ŒUVRE DES CONTRIBUTIONS DE LA THESE

Afin de pouvoir mettre en place l'étude de cas Tamagotchi, nous avons conçu un diagramme de classe comme le montre la Figure 37. Nous avons l'application Tamagotchi, à chaque connexion de l'utilisateur à l'application, nous disposons une exécution qui est effectuée par la séquence des situations. La classe « Situation » est l'unité fondamentale dans le diagramme de classe. Elle permet de savoir quels sont les éléments dans une situation. Lors de l'exécution, il s'agit d'un observé permettant d'observer tous ce qui s'est passé. Ce module représente la notion de traces qui sera utilisée pour le module de prise de décision. La combinaison de ces modules constitue un cas d'étude Tamagotchi interactive à exécution adaptative.

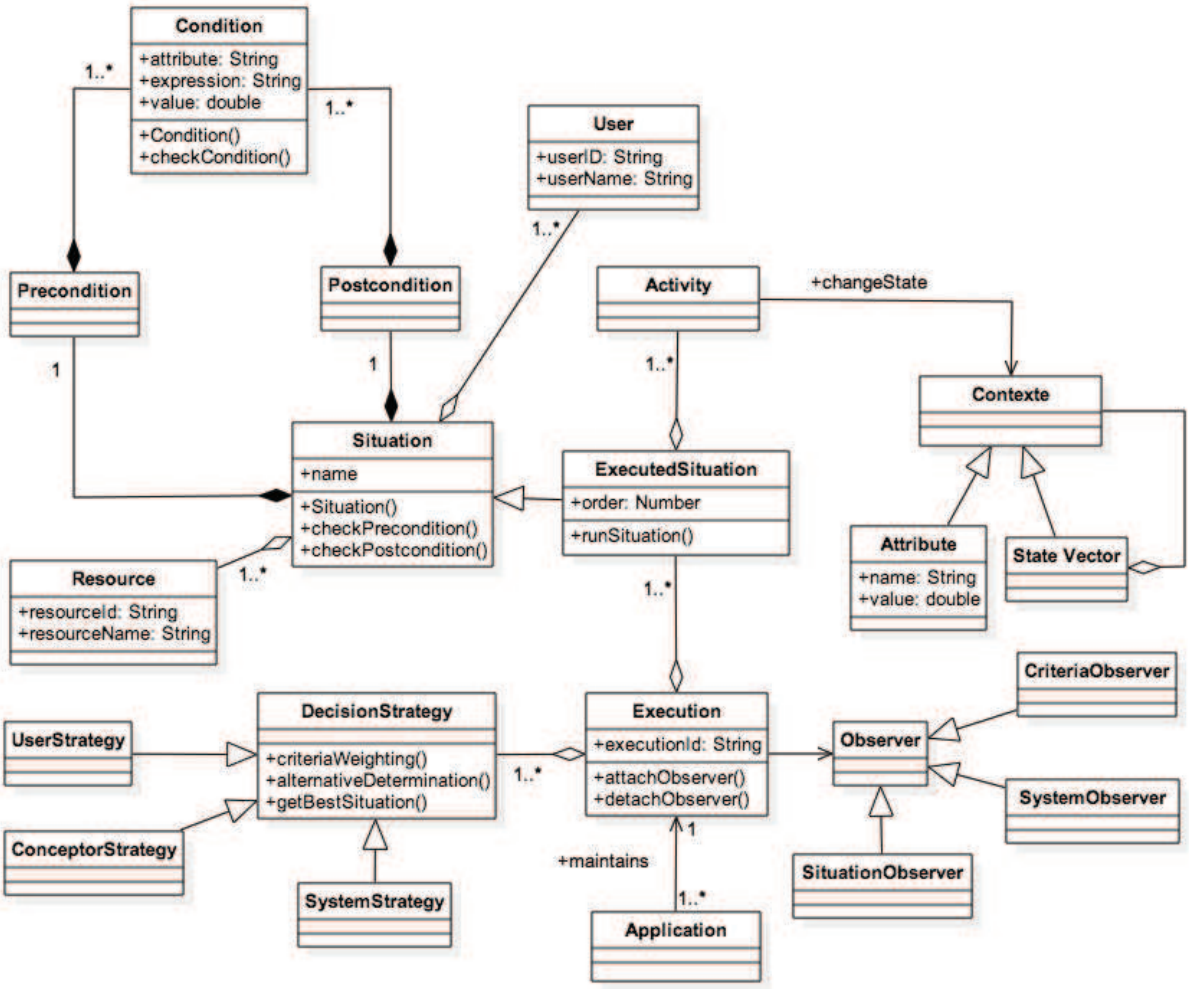

Figure 37. Diagramme de classe de l'étude de cas Tamagotchi

Dans la suite du chapitre, nous allons montrer d'abord la contribution 1 concernant la mise en place d'un système à base de traces dans le cas du Tamagotchi. Nous aborderons dans la suite les trois contributions étant des trois phases dans notre processus d'aide à la décision multicritère à base de traces.

6.2.1 CONTRIBUTION 1 : SYSTEME A BASE DE TRACE POUR LES APPLICATIONS INTERACTIVES A BASE DE SITUATIONS

Nous voulons montrer dans cette section notre système à base de traces dans une application interactive à base de situations. Nous décrivons au fur et à mesure les sous-systèmes dans le SBT conçu comme le principe présenté dans 3.5.

6.2.1.1 Collecte de traces premières

En appliquant le modèle défini dans la section 3.5.1, nous projetons dans le cas d'étude du Tamagotchi pour l'intégrer dans le SBT. Nous décrivons en détail chacun des composants associés dans ce modèle :

- $U = \langle info_1, info_2, \dots, info_j \rangle$: il contient les informations extraites depuis le profil de l'utilisateur.
- $\{V, S, \Delta\}_t$ décrit l'exécution à un moment t dans laquelle :
    - V : le vecteur d'état du système, dans notre cas Tamagotchi, nous associons les états du Tamagotchi comme les éléments dans ce vecteur. Puisque ces états varient en fonction du temps, nous avons besoin de les observer pour tracer ses changements.

      $V = \langle satiété, ennui, fatigue, soin, amis, politesse, propreté, disponibilité, âge \rangle$
    - S : la situation exécutée ; elle admis une parmi des situations suivantes : Manger, Jouer, Dormir, Soigner, Socialiser, Éduquer, Entretenir.
    - Δ : l'ensemble des valeurs des critères. Nous disposons dans ce modèle trois critères à observer qui sont : la Santé, la Socialisation, la Maturité.
- Δ_{fin} est l'ensemble des états des critères à la fin de l'exécution. Quand l'utilisateur a terminé l'exécution, nous observons quels sont les critères qui sont satisfaits (Succès) et ne sont pas satisfaits (Échec).

En nous basant sur ce modèle, pour une exécution complète d'un utilisateur, nous obtenons une trace première. Sa taille dépend du temps d'exécution de l'utilisateur. Plus le temps est long, plus la taille de trace augmente. Nous conservons la trace première dans la base de traces. Si nous voulons utiliser les traces pour décider, il nous faut passer à la transformation.

Concernant la collecte des traces nécessaire pour construire la base de trace utilisée par les autres contributions de la thèse, nous avons développé un prototype de l'application Tamagotchi pour permettre à de utilisateurs d'y jouer. Lors de leurs manipulations, nous avons collecté les traces selon le modèle que nous avons défini. Néanmoins, la diffusion de ce prototype aux utilisateurs a été un peu difficile car l'application est une version hors ligne et la collecte des données réelles a été limitée. Par ailleurs, pour tester notre processus d'aide à la décision à base de traces, nous avions besoin d'une large base de données pour effectuer nos calculs. C'est pour cela que nous avons complété les données réelles par des données issues de simulations. Nous avons mis en place une simulation permettant de générer automatiquement les données selon le format désiré en appliquant nos règles définies pour construire les enregistrements. Ces règles sont définies par une analyse logique : par exemple, quand la valeur *satiété* du Tamagotchi est inférieure à 0,2, nous allons lui donner à manger, *etc*. De plus, nous avons ajouté dans la simulation des bruits pour augmenter la suffisance et la couverture des données. Par exemple, si le Tamagotchi a faim, au lieu de lui donner à manger, nous le faisons dormir, *etc*. Nous avons fusionné les deux bases (la réelle et la simulée) dans la base de trace effective utilisée par notre système et ces mécanismes et toutes les nouvelles exécutions sont ajoutées à cette base là.

La taille de la base finale obtenue (après avoir combiné les données réelles et les données simulées) est de 1014 exécutions (dont 1000 exécutions simulées et 14 exécutions réelles). À partir de ces 1014 exécutions, nous obtenons 1014 traces premières.

6.2.1.2 Transformation de traces

Nous avons trois besoins qui associent à la pondération des critères (contribution 2), la détermination des alternatives (contribution 3) et la prise de décision multicritère (contribution 4). Comme nous avons dit, nous définissons séparément trois modèles de transformation qui correspondent avec les trois besoins. Le modèle de transformation a pour but d'extraire les

informations pertinentes pour analyser selon les besoins particuliers. Nous avons donc décidé de présenter la transformation des traces premières dans chacune des contributions de la thèse.

Le résultat de cette phase est l'ensemble des traces transformées correspondant à nos besoins. Nous avons concrètement :

- Base de traces pour la contribution 2 contenant 1014 traces transformées ;
- Base de traces pour la contribution 3 et 4 contenant 9884 traces transformées (9500 traces simulées et 384 traces réelles).

6.2.2 CONTRIBUTION 2 : PONDERATION DES CRITERES

Nous allons présenter dans cette section l'application de notre méthode de pondération des critères introduite dans 4.2. Nous visons deux objectifs : le premier consiste à montrer comment appliquer la pondération des critères dans le système Tamagotchi ; le second aborde la performance de notre proposition par rapport aux différentes approches de pondération mentionnées dans 4.2.

6.2.2.1 Application de la contribution à la pondération des critères

Nous élaborons dans la suite du manuscrit le processus de pondération des critères lorsqu'il y a un nouvel utilisateur connecté au système Tamagotchi. *L'objectif principal de l'utilisateur est de gagner le jeu le plutôt possible*. Comme nous avons indiqué dans la section 5.2.3, pour évaluer cet objectif, il s'agit de trois critères concernant la *santé*, la *socialisation* et la *maturité* que l'utilisateur doit satisfaire pour finir le jeu Tamagotchi. À un instant donné, l'utilisateur doit choisir une situation pour poursuivre l'exécution. Ce choix est fait par l'application des méthodes de décision multicritère en prenant en compte les critères définis précédemment. Ces méthodes nécessitent la définition des poids des critères afin de calculer la décision. Comme nous l'avons montré dans la Figure 26 présentant notre processus de décision multicritère à base de traces, avant de rentrer dans la phase de pondération, il est obligatoire d'identifier l'objectif à atteindre de l'utilisateur. Un objectif doit respecter le standard SMART indiqué dans la section 4.1.1. Nous allons donc décrire cette phase en indiquant le standard SMART.

Dans le système Tamagotchi, tous les utilisateurs ont envie de gagner le jeu. Cet objectif est alors vérifié par les cinq caractéristiques suivantes :

- Spécifique : l'énoncé de l'objectif est compréhensible ;
- Mesurable : un utilisateur gagne le jeu ou non, cela dépend du calcul du système ;
- Applicable : l'objectif n'est pas en dehors du cadre du jeu ;
- Réaliste : nous ne demandons pas un objectif irréel ;
- Temporelle : un utilisateur ne prend pas trop de temps pour achever le jeu.

Le principe de notre approche de décision demande d'abord de pondérer les critères. Afin d'arriver à mettre des priorités aux critères, nous collectons les exécutions des précédents utilisateurs qui ont utilisé l'application Tamagotchi. Nous observons le résultat final des utilisateurs en fonction de leurs compétences et leurs informations dans leurs profils. En effet, un utilisateur donné a des compétences et des capacités différentes. Il y a une corrélation entre les compétences de l'utilisateur et les résultats qu'il obtient. Avec de meilleures compétences,

l'utilisateur satisfera mieux les critères de l'application. Une mauvaise compétence peut influencer négativement un ou plusieurs critères. Nous enregistrons ces informations (compétences et accomplissement des critères dans les exécutions précédentes) dans une base de données où chaque enregistrement contient 8 champs : les 5 compétences/informations de l'utilisateur et 3 champs correspondant aux états des 3 critères.

Nous avons fait un prototype permettant de collecter des traces des utilisateurs lors de leur utilisation de l'application. Afin de collecter les informations décrites dans la section 4.2.1, nous y avons intégré une base de données dont chaque enregistrement représente d'une fois de l'exécution de l'application d'un utilisateur avec le système Tamagotchi. Chaque enregistrement a deux types d'informations :

- Des informations extraites depuis le profil de l'utilisateur :

Un profil d'utilisateur contient les informations décrites dans la Figure 27. Nous avons extrait depuis le profil les informations de deux catégories : Administration et Compétences. Les informations de la catégorie Administration sont à gauche de la Figure 38 qui présente l'interface utilisateur de notre prototype consacrée au profil utilisateur et celles de la catégorie Compétences sont à droite. Concernant la catégorie « Administration », nous avons extrait deux informations qui sont l'âge et le genre de l'utilisateur. Nous étudions l'âge de l'utilisateur en nous basant sur les études sociologiques (les enfants ne maitrisent pas beaucoup avec les applications compliquées, les adolescents maitrisent bien les jeux, les personnes âgés ont besoin plus de temps par rapport les adolescents pour maitriser les jeux). Nous divisons l'âge de l'utilisateur en 5 groupes qui ont été décrit dans le Tableau 18. En ce qui concerne la catégorie « Compétences », nous avons extrait trois compétences : logique (le niveau logique de l'utilisateur), ordinateur (l'utilisateur peut bien maitriser l'ordinateur ou non), jeu vidéo (l'utilisateur a expérience sur les jeux vidéo ou non). Les valeurs des trois compétences sont variées dans une échelle de 1, 2,… à 5. Plus la valeur de compétence est haute, plus l'utilisateur maitrise bien la compétence.

Tableau 18. Description des informations dans le profil de l'utilisateur

Nom	Valeur	Signification
Âge	{I, II, III, IV, V}	Nous divisons l'âge de l'utilisateur en cinq groupes comme suit I (de 1 à 5 ans), II (de 6 à 12 ans), III (de 13 à 50 ans), IV (de 51 à 65 ans) et V (à partir 66 ans) (étude sociologique)
Genre	Féminin (F), Masculin (M)	
Logique	*1 → 5*	Plus la valeur est proche que 5, plus la compétence est bonne.
Ordinateur	*1 → 5*	
Jeu vidéo	*1 → 5*	

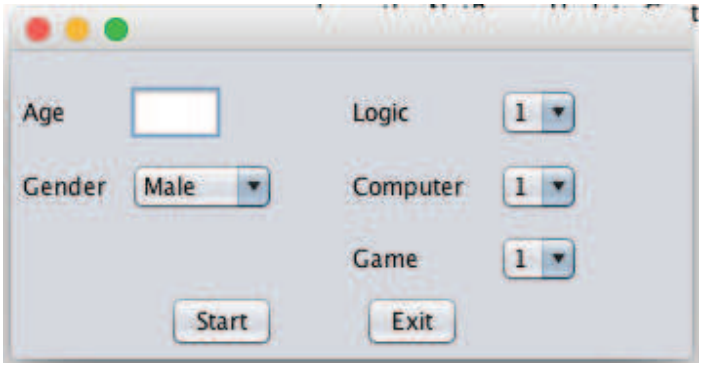

Figure 38. Informations extraites depuis le profil de l'utilisateur

- Une partie des traces de l'utilisateur.

Afin d'utiliser les traces pour pondérer les critères, depuis le modèle de traces défini dans la section 6.2.1. Pour chaque utilisation d'un utilisateur au système Tamagotchi, nous avons observé à la fin si l'utilisateur a satisfait les critères ou non.

La combinaison des deux parties constitue une base de données présentée la Figure 39.

No.	age Nominal	gender Nominal	logic Nominal	computer Nominal	game Nominal	santé Nominal	socialisation Nominal	maturité Nominal
1	V	F	2	4	3	Yes	Yes	No
2	III	M	5	3	3	Yes	Yes	Yes
3	II	M	5	1	3	Yes	Yes	No
4	III	M	5	1	3	Yes	No	No
5	IV	F	2	3	2	Yes	Yes	Yes
6	I	M	5	1	1	No	No	No
7	IV	F	3	5	3	No	Yes	Yes
8	I	M	1	5	3	Yes	Yes	No
9	III	F	3	5	2	Yes	No	No
10	IV	F	2	5	4	No	Yes	Yes
11	IV	F	2	4	2	Yes	Yes	Yes
12	II	F	1	1	4	No	No	Yes

Figure 39. Base de données pour la pondération des critères du système Tamagotchi

Pour construire le moteur de prédiction, nous avons construit une base de 1000 enregistrements correspondant à 1000 utilisations du système Tamagotchi par une simulation. Nous avons programmé une simulation permettant de générer les enregistrements en respectant le format défini. La génération des cinq premiers attributs est faite aléatoirement. À la fin de chaque exécution, nous avons déduit la satisfaction des critères (Yes/No) par un ensemble de règles de simulation prédéfinis. Notre base contient plusieurs cas de simulation ; par exemple, un utilisateur a des bonnes compétences, mais il n'a pas pu gagner le jeu (par manque de concentration par exemple). Ou bien, si un utilisateur n'a pas des bonnes compétences, mais il a joué avec beaucoup d'efforts, il peut satisfaire tous les critères de l'application. Nous combinons tous les cas possibles dans le fonctionnement du système Tamagotchi. La base de données contient des données normales et aussi des bruits, le but est de construire une base de données générale et suffisante.

Afin de prioriser les critères, nous utilisons la base de données créée ci-dessus pour construire le modèle de prédiction afin de prédire la capacité de satisfaction des critères lorsqu'il y a un

nouvel utilisateur qui décide de jouer. Nous construisons séparément les moteurs de prédiction en fonction des critères. Le résultat de chaque modèle est la probabilité de Succès d'Échec du critère correspondant. À partir des 1000 enregistrements, nous appliquons la théorie de Naïve Bayes présentée dans 4.2.4 pour construire le modèle de prédiction. Ce modèle se constitue d'un ensemble de probabilités présentées dans le Tableau 19.

Tableau 19. Description en détail des probabilités dans le modèle de prédiction d'état des critères

	Probabilité
	$P(Succès)$
	$P(\text{Échec})$
Age	$P(age/Succès)$, $P(age/\text{Échec})$
Genre	$P(genre/Succès)$, $P(genre/\text{Échec})$
Logique	$P(logique/Succès)$, $P(logique/\text{Échec})$
Ordinateur	$P(ordinateur/Succès)$, $P(ordinateur/\text{Échec})$
Jeu	$P(jeu/Succès)$, $P(jeu/\text{Échec})$

Pour décrire l'application de notre processus de pondération des critères nous prenons un exemple. Un utilisateur a un profil défini par : I = <*âge* = 24, *genre* = F, *logique* = 3, *ordinateur* = 2, *jeu vidéo* = 1>. Nous voulons calculer pour chaque critère deux valeurs qui estiment la probabilité d'être Succès et Échec. Ainsi, si nous considérons le critère Santé, nous devrons calculer $Predict_{Santé}(Succès)$ et $Predict_{Santé}(\text{Échec})$. Ci-dessous le calcul en détail pour le critère Santé :

$$\begin{aligned} P(Succès/I) = {} & P(Succès)\times P(age = II/Succès)\times P(genre = F/Succès) \\ & \times P(logique = 3/Succès)\times P(ordinateur = 2/Succès) \\ & \times P(jeu\ vidéo = 1/Succès) \end{aligned} \quad (67)$$

$$\begin{aligned} P(\text{Échec}/I) = {} & P(\text{Échec})\times P(age = II/\text{Échec})\times P(genre = F/\text{Échec}) \\ & \times P(logique = 3/\text{Échec})\times P(ordinateur = 2/\text{Échec}) \\ & \times P(jeu\ vidéo = 1/\text{Échec}) \end{aligned} \quad (68)$$

Nous faisons la même façon pour les deux autres critères Socialisation et Maturité. Nous obtenons le résultat comme le Tableau 20.

Tableau 20. Résultat de la prédiction des critères

Critère	Santé	Socialisation	Maturité
Taux Succès	46%	59%	78%
Taux Échec	54%	41%	22%

En appliquant le principe abordé dans 4.2.4, nous nous concentrons sur le taux échec pour calculer les poids des critères comme décrit dans (41). Par exemple, pour le critère Santé :

$$w_{santé} = \frac{0,54}{0,54 + 0,41 + 0,22} = 0,46 \quad (69)$$

La liste des poids des critères est illustrée dans le Tableau 21.

Tableau 21. Résultat de la pondération des critères

Critère	Santé	Socialisation	Maturité
Poids	0,46	0,35	0,19

6.2.2.2 Évaluation et Discussion

Cette section vise à évaluer notre approche de pondération à base de traces. Nous présentons la performance des modèles de prédiction construits pour faire le calcul.

À partir de la base de traces, nous allons tester la capacité de prédiction de l'algorithme Naïve Bayes par rapport aux différentes approches indiquées dans la section 4.2.2 qui sont : k-NN, SVM, Neural Network. Nous avons évalué la performance de la méthode Naïve Bayes à la prédiction des états d'accomplissement des critères en utilisant le logiciel Weka [100]. Le protocole de test est l'utilisation de l'approche k-fold. K-fold est une des façons qui permet de tester la performance des algorithmes dans la fouille de données. L'idée principale de ce protocole est de diviser aléatoirement la base initiale en 10 parties (fold). Neuf folds parmi les 10 sont utilisés pour la base d'apprentissage, le reste est utilisé pour la base de test. Nous faisons réitérons le processus 10 fois afin d'avoir une variété de données. Chaque fois, nous calculons le taux de prédiction. Après avoir effectué 10 fois, nous calculons la moyenne de ces 10 fois. Cette moyenne illustre le taux de prédiction correcte du modèle. Nous testons séparément les moteurs de prédiction des 3 critères selon le protocole 10 folds. Les trois étant indépendants, nous présentons les détails de leurs taux de prédiction dans des tableaux différents : le Tableau 22, le Tableau 23 et le Tableau 24. Dans ces trois tableaux, un unité de temps pour construire le modèle de prédiction est égale à 3 secondes (résultat depuis le logiciel Weka [100] qui consiste à tester les méthodes dans la fouille de données).

Tableau 22. Performance du modèle de prédiction pour le critère Santé

Méthode	**Taux de prédiction correct**	**Temps pour construire le modèle de prédiction**
Naïve Bayes	76,5%	1 unité
k-NN	73,7%	Pas besoin de modèle
Neural Network	76,1%	20 unités
SVM	76,6%	2 unités

Selon le Tableau 22, le taux de prédiction de la méthode Naïve Bayes pour le critère Santé est le meilleur et sensiblement égal à celui de SVM.

Tableau 23. Performance du modèle de prédiction pour le critère Socialisation

Méthode	**Taux de prédiction correct**	**Temps pour construire le modèle de prédiction**
Naïve Bayes	69,7%	1 unité

k-NN	71,5%	Pas besoin de modèle
Neural Network	74,1%	19 unités
SVM	70,5%	2 unités

D'après le Tableau 23, le taux de prédiction correcte de la méthode Naïve Bayes est 69,7%. Il est proche de celui de SVM mais se situe au milieu des 4 méthodes.

Tableau 24. Performance du modèle de prédiction pour le critère Maturité

Méthode	**Taux de prédiction correct**	**Temps pour construire le modèle de prédiction**
Naïve Bayes	74,2%	1 unité
k-NN	79,1%	Pas besoin de modèle
Neural Network	77,6%	19 unités
SVM	74,5%	3 unités

Le Tableau 24 indique que le taux de prédiction correcte de Naïve Bayes et le plus bas des 4 mais atteint néanmoins plus de 74%.

D'après le résultat de comparaison de ces trois tableaux ci-dessus, nous trouvons que la performance de prédiction entre la Naïve Bayes et celle de SVM sont toujours presque similaires. Pourtant, le temps de calcul de SVM est toujours plus grand que celui de Naïve Bayes. La méthode Neural Network a également cet inconvénient. Des fois, le taux de prédiction correcte de cette méthode est très élevé mais elle a besoin d'une durée très longue pour construire le modèle de prédiction. Par conséquent, nous concluons que SVM et Neural Network ne sont pas bien adaptés aux applications en temps réel. Par ailleurs, la méthode k-NN a prouvé sa performance de prédiction mais le résultat dépend strictement du paramètre k (nombre des voisins les plus proches à fixer obligatoirement). Tandis que Naïve Bayes n'a pas besoin de fixer de paramètre additionnel pour les calculs. Nous conclurons que Naïve Bayes est une solution efficace pour notre approche de pondération de critères.

6.2.3 CONTRIBUTION 3 : DETERMINATION DES CANDIDATS POUR LA DECISION

La démonstration de cette contribution sera présentée dans cette section. Nous allons d'abord consacrer à l'application de notre approche à la détermination des situations candidates. Dans la suite du manuscrit, nous évaluons notre approche en donnant des discussions.

6.2.3.1 Application de la contribution à la détermination des situations candidates

Notre approche de détermination est divisée en deux phases : celle d'identification des situations potentielles, celle d'estimation de l'utilité des situations potentielles. Nous disposons d'une base de traces pour manipuler le calcul, elle sera disponible en ligne. Nous extrayons depuis le modèle de trace les deux composants : Ω et Δ. Chacun correspond à une base de données utilisée pour chaque phase au-dessus. La Figure 40 et la Figure 41 représentent deux bases de données pour deux phases afin de déterminer des situations candidates pour la décision à la section 4.3.

Phase d'identification des situations potentielles

Chaque enregistrement dans la Figure 40 a sept champs dont six attributs du Tamagotchi et un dernier élément étant la situation exécutée, par exemple: Jouer, Dormir, etc. Nous avons construit un prototype Tamagotchi pour recueillir des données réelles lors de l'exécution du jeu des utilisateurs. Pourtant, pour avoir un modèle de prédiction efficace, il nous faut une grande base de traces afin de le construire. C'est pour cela que nous avons effectué une simulation de données. Nous avons créé aléatoirement le vecteur d'état, et nous avons appliqué nos règles définies pour choisir une situation pour exécuter. Après avoir combiné les données réelles et les données simulées, nous avons obtenu une base de traces[10] qui est disponible pour tester notre méthode. Statistiquement, nous avons 9884 traces (9500 traces simulées et 384 traces réelles) qui se composent de 2273 situations Manger, de 233 situations Entretenir, de 891 situations Jouer, de 2304 situations Soigner, de 2269 situations Dormir, de 1380 situations Socialiser et de 534 situations Éduquer. Nous avons utilisé ces informations pour construire notre modèle de prédiction des situations potentielles.

Relation: Tamagotchi

No.	satiety Numeric	tiredness Numeric	sadness Numeric	care Numeric	friendship Numeric	politeness Numeric	situation Nominal
1	0.86	-0.32	-0.21	-0.9	2.0	0.34	healing
2	0.01	0.43	0.61	-0.32	0.0	0.23	healing
3	-0.43	-0.23	0.11	-0.06	2.0	-0.12	feeding
4	0.56	0.16	-0.85	0.25	7.0	0.38	socializing
5	0.33	0.16	-0.04	0.14	6.0	-0.08	educating
6	0.27	-0.38	0.15	-0.5	2.0	0.13	healing
7	-0.57	-0.13	-0.73	0.34	8.0	-0.65	socializing
8	-0.55	0.27	-0.65	-0.3	6.0	-0.17	socializing
9	-0.02	-0.59	-0.18	0.28	6.0	0.52	sleeping
10	0.76	-0.33	0.17	-0.04	3.0	-0.09	sleeping
11	0.68	-0.22	-0.55	-0.03	2.0	0.19	socializing
12	-0.71	0.19	0.09	-0.77	0.0	0.01	healing

Figure 40. Base de données pour la phase de prédiction des situations potentielles

Comme décrite dans la section 4.3.2.1, nous avons besoin un modèle de prédiction des situations potentielles parmi les situations possibles. Pour construire ce modèle, il nous faut considérer tous les enregistrements dans la base extraite dans la Figure 40 et appliquer le principe Naïve Bayes. Puisque les attributs dans cette base sont des valeurs numériques, il faut calculer la moyenne et l'écart type de chaque attribut. L'ensemble des moyennes et des écarts type se constitue le modèle de prédiction des situations potentielles.

Afin d'illustrer comment identifier les situations potentielles selon le modèle de prédiction construit, nous prenons un petit exemple comme suit : le contexte du système à un instant donné est défini par le vecteur V = (*satiété* = 0,03 ; *fatigue* = 0,26 ; *ennui* = 0,04 ; *soin* = 0,09 ; *amis* = 2 ; *politesse* = -0,7). Selon V, nous vérifions quelles sont les situations potentiellement exécutables. Nous décrivons en détail le calcul de prédiction de la situation Manger.

Deux valeurs $\mu_{satiété}^{Manger} = 0{,}45$ et $\sigma_{satiété}^{Manger} = 0{,}22$ sont respectivement la moyenne et l'écart type de l'attribut satiété.

[10] Tamagotchi Traces: https://app.box.com/s/5feoqqmsu39stbmq3l0g

$$P(satiété = 0{,}03/Manger) = \frac{1}{\sqrt{2\pi}\times 0{,}22} \times e^{\frac{-(0{,}03+0{,}45)^2}{2\times 0{,}05^2}} \approx 0{,}19 \quad (70)$$

Nous appliquons la même façon pour les autres attributs, nous obtenons : $P(\text{fatigue} = 0{,}26/\text{Manger})$, $P(\text{ennui} = 0{,}04/\text{Manger})$, $P(\text{soin} = 0{,}09/\text{Manger})$, $P(\text{amis} = 2/\text{Manger})$ et $P(\text{politesse} = -0{,}7/\text{Manger})$. La probabilité de la situation manger est égale à $P(\text{Manger}) = 2273/9884$.

$$\begin{aligned} P(Manger/V) &= P(Manger)\times P(satiété = 0{,}03/Manger)\times P(fatigue \\ &= 0{,}26/Manger)\times P(ennui = 0{,}04/Manger)\times P(soin \\ &= 0{,}09/Manger)\times P(amis = 2/Manger)\times P(politesse \\ &= -0{,}7/Manger) \approx 0{,}0012 \end{aligned} \quad (71)$$

Nous continuons le calcul pour toutes les situations. Nous calculons ensuite la prédiction pour chaque situation comme décrit dans 4.3.2.1. En donnant la valeur du seuil s = 10%. Nous obtenons l'ensemble des situations potentielles dans le Tableau 25.

Tableau 25. Résultat de la prédiction des situations potentielles

Situation	Probabilité de prédiction	Résultat
Manger	**21,662%**	**potentielle**
Entretenir	0,0003%	non potentielle
Jouer	**11,128%**	**potentielle**
Soigner	**20,3%**	**potentielle**
Dormir	2,338%	non potentielle
Socialiser	**16,863%**	**potentielle**
Éduquer	**27,127%**	**potentielle**

Selon le Tableau 25, l'ensemble des situations potentielles se compose des situations suivantes : Manger, Jouer, Soigner, Socialiser et Éduquer. Pourtant, en ce moment, ce n'est que la prédiction statistique. Cet ensemble ne prend pas en compte l'aspect multicritère. Actuellement, il y a des situations avec des probabilités de prédiction très élevées mais ce n'est sûr qu'elles puissent satisfaire des critères. Par conséquent, nous passons à la deuxième phase qui consiste à estimer l'utilité des situations potentielles identifiées.

Phase d'estimation de l'utilité des situations potentielles

Nous allons maintenant considérer la phase de calcul de l'utilité par l'analyse des traces comme la Figure 41. Chaque enregistrement dans la Figure 41 représente le changement des valeurs de trois critères. En détail, il contient trois valeurs de l'accomplissement des critères avant d'exécuter la situation choisie et trois valeurs après l'exécution de la situation choisie qui a été mentionnée à la fin de chaque enregistrement.

Relation: Tamagotchi

No.	healthBefore Numeric	socializationBefore Numeric	maturityBefore Numeric	healthAfter Numeric	socializationAfter Numeric	maturityAfter Numeric	situation Nominal
1	1.4	2.0	0.79	1.6	2.04	1.55	healing
2	1.6	2.04	1.8	0.58	1.18	2.84	healing
3	1.18	3.45	3.46	1.4	2.0	0.79	feeding
4	1.18	1.59	3.7	1.18	1.59	3.98	socializing
5	0.72	2.74	10.94	0.64	2.41	12.63	educating
6	1.07	2.33	1.61	1.07	2.33	2.24	healing
7	1.6	2.04	1.8	0.58	1.18	2.84	socializing
8	1.4	2.0	0.79	1.35	1.92	1.57	socializing
9	1.0	2.0	0.79	1.07	2.33	1.41	sleeping
10	1.35	1.92	1.83	1.18	3.45	3.46	sleeping
11	1.22	1.92	2.88	0.71	1.43	3.81	socializing
12	1.0	2.0	0.79	1.3	2.08	1.52	healing

Figure 41. Base de données pour la phase d'estimation de l'utilité des situations potentielles

Dans cette phase, nous devons définir les trois seuils pour trois critères. Dans ce contexte, ses valeurs sont : $p_{santé} = 0{,}4$; $p_{socialisation} = 1$; $p_{maturité} = 1$. Le seuil p doit être défini par le concepteur ou l'utilisateur. Cette valeur dépend strictement du type et nature de chaque critère. Par exemple, le critère Santé, sa valeur se varie de 0 à 2. En considérant cet intervalle de valeur, nous avons défini que si l'exécution de la situation fait augmenter la valeur de la santé de plus de 0,4, cette situation est pertinente pour la prochaine exécution. Nous ne pouvons pas comment quantifier précisément les seuils, nous présentons juste un exemple. En général, les seuils de p sont définis de manière expérimentale. Une fois les seuils ont été définis, nous appliquons (44) et (45) pour calculer l'utilité pour chaque situation potentielle. Le résultat de cette phase est résumé dans le Tableau 26.

Tableau 26. Résultat d'estimation de l'utilité des situations potentielles

Situation	Utilité du critère			Résultat
	Santé	Socialisation	Maturité	
Manger	**1**	**0,2**	0	**alternative**
Jouer	-1	1	-1	non alternative
Soigner	**1**	0	**1**	**alternative**
Socialiser	-1	**1**	**0.6**	**alternative**
Éduquer	0	**1**	**1**	**alternative**

Si nous fixons la valeur de K = 2, c'est-à-dire qu'une situation doit avoir au moins deux valeurs d'utilité positives sur les trois critères. Selon l'estimation de l'utilité présentée dans le Tableau 26, il y a seulement quatre situations (Manger, Soigner, Socialiser et Éduquer) qui font augmenter l'utilité des critères en se basant sur l'analyse des traces. Si nous suivons l'une de ces situations, nous pourrons mieux remplir les critères de l'application. Nous pouvons conclure que le résultat de notre processus de détermination des alternatives est l'ensemble des quatre situations ci-dessus. Nous pouvons maintenant appliquer tout algorithme de décision pour calculer le choix final.

6.2.3.2 Évaluation et Discussion

Tout d'abord, nous avons évalué la performance de la méthode Naïve Bayes pour la détermination des situations potentielles en utilisant le logiciel Weka [100]. Dans le Tableau

27, nous avons résumé le taux de prédiction correcte (mesuré avec Weka) et le temps nécessaire (en unités de temps) pour construire le modèle de prédiction en utilisant les quatre méthodes de fouille de données mentionnées à la section 4.3.2.1. Nous utilisons la base de traces obtenue (décrite dans la Figure 40) pour effectuer l'évaluation.

Nous voyons que le taux de prédiction correcte de la technique k-NN est le plus faible. Cependant cette différence entre les trois autres techniques n'est pas très significative même si Naïve Bayes est le meilleur des trois. Concernant les temps de construction des modèles de prédiction (1 unité de temps est égale à 3 secondes), pour SVM et Neural Network ces temps sont plus longs que celui de Naïve Bayes.

Tableau 27. Comparaison de performance entre quatre méthodes de prédiction: Naïve Bayes, kNN, Neural Network, SVM

Méthode	Taux de prédiction correct	Temps pour construire le modèle de prédiction
Naïve Bayes	83,61%	1 unité
k-NN	76,88%	Pas besoin de modèle
Neural Network	84,03%	14 unités
SVM	84,92%	6 unités

En outre, nous avons testé l'intégration de notre approche dans notre méthode de prise de décision multicritère décrite dans 4.4 pour évaluer la performance de notre approche dans une application réelle. Nous avons réalisé deux tests : l'un avec la décision utilisant notre approche de détermination des situations candidates et l'autre sans notre approche. Nous avons observé le temps de calcul de 20 décisions et nous avons obtenu les résultats représentés comme le montre Figure 42.

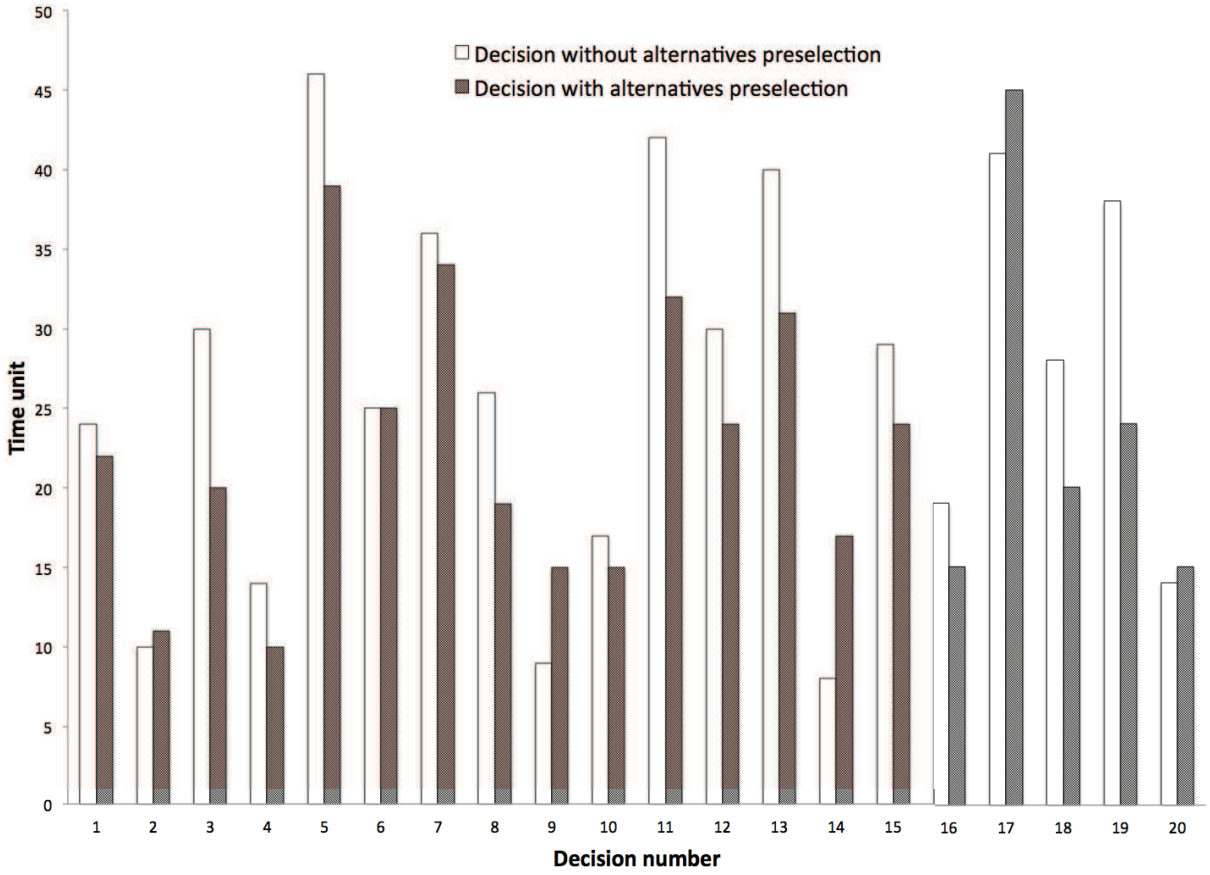

Figure 42. Comparaison du temps d'exécution pour la décision avec et sans approche de détermination des situations candidates

Nous observons que le temps de calcul avec notre approche est souvent inférieur à celui sans l'intégration de notre approche. Si nous appliquons la détermination des situations candidates avant de prendre la décision, nous pouvons réduire le nombre d'alternatives et la prise de décision n'a pas besoin de prendre en compte toutes les situations disponibles mais seulement celles qui sont pertinentes. Toutefois, il existe certains cas où le temps de calcul avec l'intégration de notre approche est plus élevé que la décision sans notre approche (décisions numéro 9, 14, 17, 20 dans la Figure 42). La raison est que, dans ces cas, toutes les situations possibles sont considérées comme des alternatives. Par conséquent, le temps de calcul est plus élevé en raison de la charge de l'exécution de notre approche de détermination des situations candidates. Néanmoins, dans le cas général, nous pouvons remarquer de meilleurs résultats de performance de notre approche de détermination. Bien que la différence dans le temps de calcul pour chaque décision ne soit pas significative entre les deux stratégies, cette différence sera importante si nous considérons le temps cumulé tout au long des décisions au cours de l'exécution de l'application. En outre, cette différence augmente lorsque l'ensemble des situations possibles de l'application augmente.

Cependant, notre approche a des limites. Elle est efficace seulement si nous avons suffisamment de traces dans la base de données. Pendant les premières exécutions, nous ne disposons pas suffisamment de traces pour construire le modèle de prédiction fiable. Dans ce cas, les utilisateurs doivent décider par eux-mêmes. Une autre limite de notre méthode est le réglage des seuils s et p. Nous devons éviter de choisir des valeurs élevées, car ils représentent la limite de pertinence pour vérifier la probabilité et l'utilité. Expérimentalement, la valeur de s

doit être de 5% ≤ *s* ≤ 10% et la valeur de *p* doit être choisi selon le type et la nature des critères pris en compte. Le nombre de situations candidates dépend de ces valeurs.

6.2.4 CONTRIBUTION 4 : APPROCHE DE PRISE DE DECISION

Nous allons décrire dans cette section comment appliquer nos contributions concernant les algorithmes de décision multicritère à base de traces présentées dans la section 4.4. Nous considérons deux types de choix : celui du système, qui est assuré par la méthode PROMETHEE II à base de traces, et celui de l'utilisateur, pris en compte par la méthode de la Logique Subjective à base de traces. Nous présentons d'abord l'application de chacune de ces méthodes séparément avant de combiner le résultat de ces deux choix à la fin de cette section.

6.2.4.1 Choix du système : PROMETHEE II à base de traces

Au fil de l'exécution, l'utilisateur va choisir les situations successives pour essayer d'atteindre l'objectif final combinant les trois critères : Santé, Socialisation, Maturité. L'utilisateur se base sur le vecteur d'état fourni par le système. Nous souhaitons avec notre contribution apporter une aide à l'utilisateur en lui suggérant d'une manière automatique les situations les mieux adaptées à l'état courant en utilisant l'approche à base de traces décrite dans 4.4.1.2. Nous montrons d'abord comment appliquer notre approche, puis nous comparons ses performances à celles de PROMETHEE II classique.

Application PROMETHEE II à base de traces au calcul du choix du système

Nous prenons l'exemple du contexte du système à un instant donné défini par le vecteur : *V = (satiété* = 0,03 *; fatigue* = 0,26 *; ennui* = 0,04 *; soin* = 0,09 *; amis* = 2 *; politesse* = -0,7*)*. Ces valeurs ont été fixées après plusieurs simulations et permettent d'exprimer une dynamique d'exécution qui exploserait si un mécanisme de gestion de l'adaptabilité de l'exécution n'avait pas été mis en place. Avec ce vecteur d'état, nous avons déterminé, en utilisant le résultat de la dernière expérimentation concernant la détermination des situations candidates, l'ensemble *A* des alternatives (situations candidates) à la prise de décision contenant 4 situations : *alimenter*, *soigner*, *socialiser* et *éduquer*. En ce qui concerne la base de traces, nous disposons, comme pour les autres expérimentations, d'une base de traces initiale contenant les traces selon le modèle de traces défini. Depuis cette base, nous extrayons les enregistrements contenant une des situations candidates identifiées. Puis nous créons 3 ensembles de traces correspondant aux 3 critères. Les Figure 43, Figure 44 et Figure 45 représentent respectivement des extraits des 3 bases obtenues pour la *Santé*, la *Socialisation* et la *Maturité*.

satiety Numeric	tiredness Numeric	sadness Numeric	care Numeric	situation Nominal
0.76	-0.55	-0.75	-0.38	socializing
0.3	-0.55	-0.58	0.42	socializing
0.09	-0.55	-0.81	-0.67	socializing
-0.67	-0.55	-0.04	0.02	feeding
-0.73	-0.55	0.22	-0.22	feeding
-0.01	-0.55	-0.7	0.23	socializing
-0.09	-0.54	-0.75	-0.69	socializing
-0.64	-0.54	0.22	0.49	feeding

Figure 43. Base de traces pour le critère Santé

sadness Numeric	friendship Numeric	situation Nominal
-0.19	3.0	treating
0.21	9.0	feeding
-0.01	5.0	feeding
0.69	5.0	feeding
0.0	1.0	feeding
0.08	4.0	treating
0.49	8.0	treating

Figure 44. Base de traces pour le critère Socialisation

tiredness Numeric	friendship Numeric	politeness Numeric	situation Nominal
-0.13	4.0	-0.62	treating
-0.34	8.0	0.69	treating
0.07	3.0	0.15	treating
-0.02	3.0	-0.17	socializing
0.4	9.0	0.16	treating
-0.46	9.0	0.05	socializing
-0.31	9.0	0.17	feeding

Figure 45. Base de traces pour le critère Maturité

Nous décomposons aussi le vecteur d'état V en 3 sous vecteurs, un par critère, reprenant les attributs intervenant dans chaque critère :

- $v_{santé} = (\text{satiété} = 0{,}03\,;\ \text{fatigue} = 0{,}26\,;\ \text{ennui} = 0{,}04\,;\ \text{soin} = 0{,}09)$,
- $v_{socialisation} = (\text{ennui} = 0{,}04\,;\ \text{amis} = 2)$,
- $v_{maturité} = (\text{fatigue} = 0{,}26\,;\ \text{amis} = 2\,;\ \text{politesse} = -0{,}7)$.

Pour chaque critère, nous calculons la distance euclidienne entre le sous vecteur obtenu et chacun des enregistrements de la base correspondante. Nous obtenons pour chaque critère une matrice de distance sur laquelle il faut appliquer l'équation (48). Elle nécessite de déterminer la valeur k pour calculer l'évaluation des alternatives où k est le nombre d'entrées de la matrice ci-dessus que nous voulons utiliser dans les calculs (méthode kNN des k plus proches voisins). k peut prendre n'importe quelle valeur ; néanmoins il est utile de limiter cette valeur pour réduire le temps de calcul. Nous montrons ci-après comment déterminer une valeur pertinente de k avec l'expérimentation sur critère de *Santé*.

Nous avons fait une simulation des calculs des évaluations des alternatives selon le critère *Santé* avec différentes valeurs de k (Tableau 28). On constate qu'à partir de $k = 50$, les valeurs de $E_{santé}$ pour les 4 alternatives ne varient presque plus, Ainsi, dans notre cas, nous pouvons conclure que $k = 50$ (50 voisins les plus proches) est suffisant pour le critère *Santé*. De même manière, nous obtenons pour les critères *Socialisation* et *Maturité* respectivement 100 et 80.

Tableau 28. Observation la variation des évaluations de différentes valeurs de k

k	10	20	30	40	**50**	60	70	80
$E_{santé}$*(Manger)*	0,074	0,092	0,088	0,103	**0,101**	**0,101**	**0,100**	**0,100**
$E_{santé}$*(Soigner)*	0,162	0,192	0,179	0,188	**0,206**	**0,205**	**0,206**	**0,207**
$E_{santé}$*(Socialiser)*	0,359	0,338	0,346	0,335	**0,332**	**0,333**	**0,332**	**0,331**
$E_{santé}$*(Éduquer)*	0,405	0,379	0,387	0,374	**0,360**	**0,361**	**0,361**	**0,362**

Le Tableau 29 récapitule les évaluations des alternatives selon notre approche (application de la formule (47) à base de traces pour les trois critères. En appliquant la suite du processus PROMETHEE II, nous obtenons le classement : Éduquer > Socialiser > Soigner > Manger. Ainsi, avec le vecteur d'état ci-dessus le système suggéra la situation Éduquer pour la suite de l'exécution

Tableau 29. Récapitulatif des évaluations de quatre alternatives sur trois critères dans le cas du Tamagotchi

Critère	Manger	Soigner	Socialiser	Éduquer
Santé	0,1012	0,2065	0,3321	0,3602
Socialisation	0,1293	0,2155	0,3182	0,337
Maturité	0,1144	0,2367	0,3132	0,3357

Évaluation et Discussion

Notre proposition est un support qui permet l'évaluation des situations candidates selon différents critères. Elle aide l'utilisateur à calculer automatiquement les valeurs en se basant sur les traces des exécutions précédentes. Normalement, pour évaluer ces valeurs, il y a 2 moyens :

- quantification explicite par l'utilisateur (i) ;
- définition des fonctions d'utilité ou fonctions d'évaluation pour chaque critère (ii).

L'approche (ii) est difficile à généraliser car chaque application a des critères différents, nécessitant des fonctions d'utilité personnalisées. Notre proposition permet de résoudre ces inconvénients. Elle évite la subjectivité de l'utilisateur et n'a pas besoin d'une fonction d'utilité pour chaque critère. Afin de montrer l'apport de l'utilisation d'une base de traces, nous avons effectué un test qui permet de comparer les performances de la méthode PROMETHEE II classique avec notre proposition d'amélioration à base de traces. Nous avons comparé l'évolution de la valeur des critères et le temps d'exécution total de l'application. Pour PROMETHEE II classique, nous avons tenu compte de l'avis de l'utilisateur qui a donné lui-même des valeurs d'évaluation des alternatives pour chaque critère.

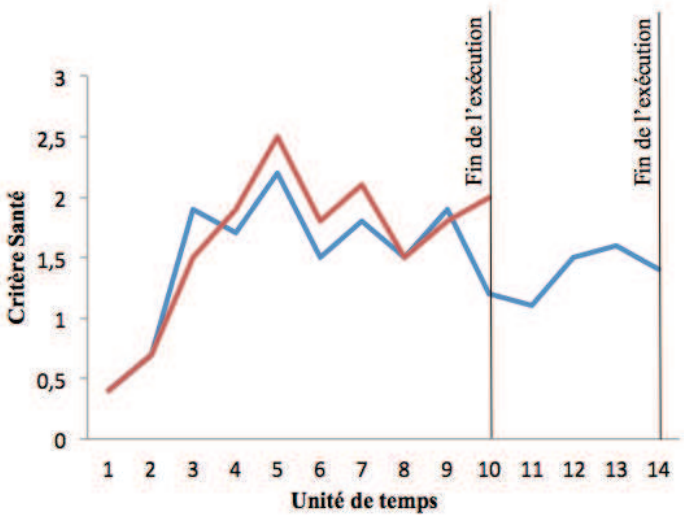

Figure 46. Comparaison sur le critère Santé entre PROMETHEE II classique et à base de traces

La Figure 46 représente l'observation de l'évolution du critère *Santé* en appliquant PROMETHEE II à base de traces (courbe rouge) et PROMETHEE II classique (courbe bleue) pour un utilisateur spécifique. L'axe des abscisses représente l'unité de temps, l'axe des ordonnées représente la valeur du critère *Santé*. Nous trouvons que l'utilisateur finit le jeu au bout de 10 unités de temps avec l'utilisation des traces alors que ça lui prend 14 unités de temps avec PROMETHEE II classique. On voit aussi que la réalisation du critère est globalement meilleure avec PROMETHEE II à base de traces. Nous avons fait le même constat pour les critères *Socialisation* et *Maturité*.

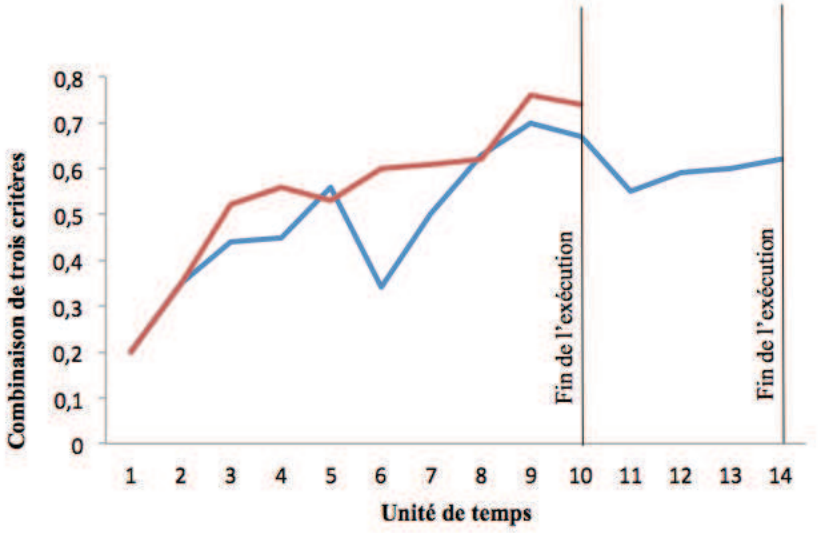

Figure 47. Comparaison sur trois critères entre PROMETHEE II classique et à base de traces

Nous avons combiné les résultats des trois critères et représenté la synthèse obtenue dans la Figure 47 après une normalisation des valeurs dans l'intervalle [0, 1]. Nous constatons là aussi les choix proposés par PROMETHEE II à bases de traces permettent d'atteindre plus vite les objectifs de l'application. Ainsi, notre approche a la capacité d'aider un utilisateur en particulier lors de sa première utilisation de l'application en l'absence d'expérience avec le système. Et durant les exécutions suivantes, elle fournit des suggestions à base d'exécutions antérieures.

6.2.4.2 Choix de l'utilisateur : Logique subjective à base de traces

Dans cette section, nous expérimentons le choix de l'utilisateur par l'application de la logique subjective à base de traces présentée dans la section 4.4.3.3. L'évaluation de notre approche sera mise en place pour valider notre contribution.

Application de la logique subjective à base de traces au calcul de choix de l'utilisateur

Nous extrayons depuis le modèle de trace une base de traces pour appliquer la logique subjective à base de traces permettant d'aider l'utilisateur de faire un choix. La Figure 48 décrit la base de traces utilisée. Nous supposons que la situation actuelle est la situation Jouer et nous souhaitons déterminer parmi les quatre situations candidates (Manger, Soigner, Socialiser, Éduquer), laquelle est la préférence de l'utilisateur.

Relation: Tamagotchi

No.	pre_sit Nominal	pre_health Numeric	pre_socialization Numeric	pre_maturity Numeric	next_sit Nominal	next_health Numeric	next_socialization Numeric	next_maturity Numeric
1	feeding	1.4	2.0	0.79	playing	1.6	2.04	1.55
2	playing	1.6	2.04	1.8	sociali...	0.58	1.18	2.84
3	socializing	0.58	1.18	3.19	cleaning	0.54	2.15	5.34
4	cleaning	0.54	2.15	6.05	playing	0.54	2.2	8.23
5	playing	0.54	2.2	8.96	sleeping	0.41	1.98	11.04
6	sleeping	0.41	1.98	11.74	educat...	0.44	1.98	13.75
7	educating	0.44	1.98	14.42	playing	0.37	2.08	16.72
8	playing	1.4	2.0	0.79	sociali...	1.35	1.92	1.57
9	socializing	1.35	1.92	1.83	sleeping	1.18	3.45	3.46
10	feeding	1.4	2.0	0.79	playing	1.64	2.04	1.56
11	playing	1.64	2.04	1.82	playing	1.41	1.82	2.6
12	playing	1.41	1.82	2.86	sociali...	1.18	1.59	3.7

Figure 48. Base de traces utilisée pour la logique subjective à base de traces

Nous appliquons notre méthode de logique subjective à base de traces pour modéliser l'idée de l'utilisateur en une opinion. Nous prenons par exemple, le calcul suivant :

Entrée : La situation qui vient de finir jouer, l'ensemble de situations candidates $Cand = \{Manger, Soigner, Socialiser, Éduquer\}$, l'idée de l'utilisateur I = la situation Manger est la meilleure situation à exécuter à la prochaine exécution.

Procédure :

1. Est-ce que I est exprimée clairement ? → oui, aller à (2)
2. Avez-vous un point de vue qui est pour ou contre la proposition I? → oui, aller à 3
3. Extraire des transitions dans la base de traces où S_{prev} est égale à jouer

   Nous définissons *r* et *s* qui sont respectivement la conséquence positive et négative de la situation manger. Pour chaque enregistrement dans la base de traces, nous calculons :

$$\epsilon = \sum_{h=1}^{3} w_h \times \left(C_{next}(h) - C_{prev}(h)\right)$$

   Si $\epsilon > 0$ alors $r = r + 1$
   Si non $s = s + 1$

4. Calculer x : le niveau d'évidence de la situation Manger selon les traces : $x = \frac{r}{r+s} = 0{,}4$
5. Calculer y : demander à l'utilisateur à quel point il est d'accord pour choisir la situation Manger, $0 \leq y \leq 1$ → y = 0,3
6. Calculer z : demander à l'utilisateur à quel point il est contre pour choisir la situation $Manger$, $0 \leq z \leq 1$ → z = 0,7
7. Normalisation des résultats :
   $b = \frac{0{,}3}{0{,}7+0{,}3+(1-0{,}4)} = 0{,}1875$; $d = \frac{0{,}7}{0{,}7+0{,}3+(1-0{,}4)} = 0{,}4375$; $u = \frac{1-0{,}4}{0{,}7+0{,}3+(1-0{,}4)} = 0{,}375$
8. Quel est le nombre total des états pour la situation Manger? → 2
9. Combien d'états sont apparus dans la proposition ? → 1 (la situation Manger est la meilleure à exécuter)
10. Calculer a → a = ½

Sortie : L'opinion de la situation manger $\omega_{Manger} = \langle 0{,}1875;\ 0{,}4375, 0{,}375, 0{,}5 \rangle$

Nous appliquons cette procédure pour les trois autres situations candidates : Soigner, Socialiser et Éduquer. Nous allons ensuite calculer la probabilité d'espérance (section 4.4.3.1) de chaque opinion. Par exemple, celle de la situation Manger est égale à : $E(\omega_{Manger}) = 0{,}1875 + 0{,}375 \times 0{,}5 = 0{,}375$. Nous obtenons le résultat comme le montre le Tableau 30.

Tableau 30. Résultat de l'application de la logique subjective à base de traces dans la prise de décision

Situation	Manger	Soigner	Socialiser	Éduquer
Probabilité d'expectation	0,375	0,215	0,621	0,15

En appliquant le règle de classement des opinions, nous obtenons une liste de priorité des situations comme suit :

Socialiser ≻ Alimenter ≻ Soigner ≻ Éduquer

La situation qui sera suggérée à l'utilisateur pour exécuter est la situation Socialiser.

Évaluation et Discussion

Nous allons comparer la performance de notre approche « Logique Subjective à Base de Traces » (LSBT) avec deux autres approches qui sont indiquées dans la section 4.4.3 qui sont l'Approche Collaborative (AC) et la Logique Subjective sans l'analyse de traces (LS). Nous avons observé la situation choisie dans certaines unités de temps (une unité de temps correspond à une décision avec une situation suggérée à l'utilisateur). Après l'exécution de la situation choisie, l'utilisateur évalue sa satisfaction : il observe les valeurs obtenues pour les trois critères et marque ✓ s'il est satisfait. Le Tableau 31 résume les résultats pour les 10 premières unités de temps.

Tableau 31. Comparaison de la satisfaction de l'utilisateur entre trois approches : Logique Subjective à base de traces (LSBT), Logique Subjective sans l'analyse de traces (LS) et approche collaborative (AC)

Unité de temps	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
LSBT	Manger	Entrete	Manger	Socia	Dormir	Manger	Entrete	Jouer	Socia	Dormir
Satisfaction	✓	✓	✓		✓	✓	✓	✓	✓	✓
LS	Manger	Dormir	Jouer	Jouer	Manger	Socia	Dormir	Manger	Entrete	Manger
Satisfaction	✓	✓	✓				✓			✓
AC	Manger	Entrete	Soigner	Dormir	Manger	Manger	Jouer	Éduquer	Manger	Manger
Satisfaction	✓	✓		✓		✓			✓	✓

Tableau 32. Résultat de la satisfaction de l'utilisateur entre les utilisateurs novices et expérimentés

	6 beginners			8 experimented users		
	LSBT	LS	AC	LSBT	LS	AC
Satisfaction number	54	46	51	65	77	62

Le Tableau 31 montre que LSBT obtient plus de satisfaction de l'utilisateur que LS et AC. En outre, si nous considérons uniquement le résultat de LSBT et de LS, nous nous rendrons compte que la satisfaction de l'utilisateur dans le cas à base de traces est supérieure à la LS sans analyse de traces. Nous pouvons conclure que l'intégration des traces dans le cadre de la

logique subjective peut aider les utilisateurs à choisir efficacement une situation à exécuter en respectant leurs satisfactions.

Nous avons également lancé le test avec 14 utilisateurs différents (8 utilisateurs expérimentés et 6 débutants avec le système Tamagotchi) dans les 10 premières unités de temps. Les résultats sont présentés dans le Tableau 32. Nous pouvons observer que la satisfaction des utilisateurs pour les débutants est meilleure avec LSBT qu'avec AC et LS. Pour les utilisateurs expérimentés, la satisfaction dans LSBT est meilleure que celle avec AC, mais moins bonne qu'avec LS seule. La raison est que les utilisateurs expérimentés sont mieux formés dans le cadre de l'application et ont moins besoin d'une aide à la décision. Nous pouvons néanmoins conclure que la logique subjective à base de traces est une approche bien adaptée pour déterminer les préférences des utilisateurs lors de la prise de décision, en particulier pour les utilisateurs non expérimentés pour leurs premières exécutions.

6.2.4.3 Agrégation des choix

Ce processus permet de considérer les résultats de choix venant de 3 logiques : celle du système, celle de l'utilisateur et celle du concepteur. En reprenant le résultat des sections antérieures, nous disposons un récapitulatif comme suit :

Choix du système : Éduquer ≻ Socialiser ≻ Soigner ≻ Manger

Choix de l'utilisateur : Socialiser ≻ Manger ≻ Soigner ≻ Éduquer

Choix du concepteur : Éduquer ≻ Socialiser ≻ Manger ≻ Soigner

Nous appliquons le principe décrit dans la section 4.4.4.2 pour calculer le score de Borda pour chaque alternative :

$$Borda(\text{Éduquer}) = 4 + 1 + 4 = 9$$

$$Borda(Manger) = 1 + 3 + 2 = 6$$

$$Borda(Socialiser) = 3 + 4 + 3 = 10$$

$$Borda(Soigner) = 2 + 2 + 1 = 5$$

Le choix final est obtenu selon l'ordre décroissant des scores de Borda :

Socialiser ≻ Éduquer ≻ Manger ≻ Soigner

6.3 SYNTHESE

Dans ce chapitre, nous avons mis en place toutes nos contributions de la thèse dans le cas d'étude Tamagotchi. L'idée est d'intégrer premièrement le système à base de traces et deuxièmement le processus d'aide à la décision multicritère à base de traces. Concernant le système à base de traces, nous avons défini un modèle de traces à base de situations permettant de collecter tous ce qui est passé lors le l'exécution de l'application. En ce qui concerne le processus d'aide à la décision, nous avons présenté ses trois étapes principales qui

correspondent à trois contributions (de 2 à 4). Nous avons décrit en détail comment appliquer nos propositions dans le cas Tamagotchi en faisant des différents test afin d'évaluer la performance de notre processus d'aide à la décision multicritère à base de traces. Malgré que le résultat final de notre processus est une liste de choix qui a été obtenu par l'agrégation de différentes logiques (système, concepteur et utilisateur), elle pourra satisfaire par cet utilisateur, mais ce n'est pas optimal pour l'autre utilisateur. Nous disons que nous ne pouvons pas trouver une liste qui satisfait à tout le monde, notre résultat de choix est un résultat approximatif et acceptable.

Nous allons partir à la dernière partie de la thèse, qui est « Conclusion et Perspectives ».

Partie 4 : Conclusion

CHAPITRE 7 CONCLUSION

SOMMAIRE

L'ultime chapitre de ce manuscrit est consacré à une conclusion étendue portant sur notre travail de thèse. Nous commençons par un bilan en décrivant les trois différents apports : bibliographiques, méthodologiques et expérimentaux. Nous abordons, dans la suite, les limites de notre travail en donnant les commentaires sur ceux-ci. Enfin, nous suggérons les perspectives que notre travail offre quant à de futures recherches sur la gestion de traces et la décision multicritère basée sur les traces.

7.1 BILAN ET APPORTS DE LA THESE

L'exécution adaptative permet au système de prendre en compte l'état et les comportements de l'utilisateur dans le but d'ajuster sa propre logique d'exécution aux interactions entre l'utilisateur et le système. En quelques mots, l'objectif initial de cette thèse était de déterminer comment intégrer dans un système interactif un processus de décision multicritère, accompagné d'un système de gestion de traces adapté, permettant de choisir la meilleure situation relativement au contexte. Les travaux réalisés durant cette thèse visent à atteindre cet objectif en donnant des apports bibliographiques, méthodologiques et expérimentaux que nous détaillons ci-dessous.

7.1.1 APPORTS BIBLIOGRAPHIQUES

Nous avons porté notre premier effort de recherche sur les mécanismes d'adaptation que nous pouvons utiliser pour mettre en œuvre une exécution adaptative dans une application interactive à base de situations. Nous avons choisi d'appliquer l'approche d'aide à la décision multicritère pour répondre à nos besoins.

Nous avons ensuite établi un large état de l'art des différentes méthodes d'aide à la décision multicritère dans le Chapitre 2. Nous les avons présentées selon leur nature en les classant selon deux approches qui sont : l'approche « agréger puis comparer » et l'approche « comparer puis agréger ». Dans un effort de synthèse, nous avons proposé de considérer toutes ces méthodes, d'une part, du point de vue de l'analyse des avantages des méthodologies de calcul et, d'autre part, du point de vue de l'identification des éléments manquants pour les améliorer. En nous basant sur cette synthèse, nous avons identifié les points sur lesquels devaient porter nos nouvelles propositions. Par ailleurs, en étudiant dans le détail le processus d'aide à la décision multicritère, nous avons remarqué que les traces générées lors des anciennes exécutions peuvent permettent d'affiner le résultat de la décision. Cette approche a été abordée dans le Chapitre 3, sous le terme générique de Système à Base de Traces (SBT), pour lequel nous avons explicité les terminologies employées, les méthodologies et usages associés. En analysant les SBT existant dans la littérature, nous adaptons le principe original du SBT en proposant une méthode intégrant un système à base de traces dans une application interactive à base de situations.

7.1.2 APPORTS METHODOLOGIQUES

7.1.2.1 Sur les traces

Quel est le modèle de trace ? Quels types de données récupérons-nous lors de l'exécution ?

Nous avons défini un modèle de trace adapté à notre contexte à base de situations. Le modèle ainsi élaboré rassemble les traces générées par l'utilisateur et les traces du système. Lors de l'interaction entre l'utilisateur et le système, nous avons mis en place un modèle sous forme d'un vecteur qui permet de récupérer et stocker les traces dans la base de traces. Nous avons proposé de modéliser les traces sous forme d'un vecteur afin de faciliter le calcul dans la phase d'analyse. Dans ce vecteur, le profil de l'utilisateur lorsqu'il s'est connecté et toutes les exécutions de l'utilisateur (enchainement de situations), sont enregistrées en observant les critères définis. À la fin d'une exécution, nous avons vérifié l'état des critères par l'observation des valeurs obtenues pour chaque critère. L'ensemble de ces informations constitue le modèle de traces.

Comment construire un système à base de traces pour les applications interactives à base de situations ?

Nous avons également construit un système à base de traces permettant de gérer les traces depuis leur génération jusqu'à leur exploitation. Notre objectif est de disposer d'un SBT fonctionnant dans un contexte à base de situations. Notre SBT est conçu en respectant le principe original des SBT dont certains des composants ont été adaptés :

- Pour la collecte de traces - nous avons mis en place le modèle obtenu ci-dessus dans le sous-système de collecte. Nous avons collecté les traces générées lors de l'exécution en les modélisant sous forme du modèle de traces défini. Nous avons obtenu les traces premières.
- Pour la transformation de traces - nous avons fait le lien avec un troisième composant, l'extraction de traces, qui est dédié au processus d'aide à la décision multicritère. En tenant compte de ce processus, nous avons défini les modèles de transformation qui conviennent aux besoins du processus de décision. Nous avons trois modèles de transformation qui correspondent aux trois phases de notre processus d'aide à la décision qui sont, le modèle de transformation pour la pondération, celui pour la détermination des alternatives et celui pour la prise de décision. Selon ces modèles, nous avons transformé les traces en de nouvelles traces, appelées traces transformées. Ces traces sont utilisées pour l'extraction des traces qui ont été prises en charge par le processus de décision multicritère.

7.1.2.2 Sur la décision

De quoi décidons-nous ?

Nous avons proposé une architecture présentant différents aspects nécessaires à la mise en œuvre d'une application interactive à base de situations. Cette architecture permet l'exécution d'une application interactive par l'enchaînement des situations conçues par le concepteur.

Quand l'utilisateur a terminé une situation, notre processus de décision va être appelé pour choisir la meilleure situation parmi l'ensemble des situations disponibles. Par ailleurs, nous avons mis en place un SBT qui sert à récupérer les traces générées selon le modèle défini.

Quelles techniques utiliser pour prendre la décision ?

En analysant les approches d'aide à la décision multicritère, nous avons tiré de la synthèse de ces approches un processus d'aide à la décision multicritère composé principalement de trois étapes : (1) la pondération des critères, (2) la détermination des alternatives pour la décision et (3) la prise de décision. La méthodologie ainsi présentée est également la combinaison des algorithmes proposés afin de mettre en œuvre la décision. Pour (1), nous avons proposé un nouvel algorithme utilisant le profil d'utilisateur et les traces pour pondérer les critères. L'intérêt de cet apport est de surmonter un inconvénient concernant le poids des critères qui apparaît dans la plupart des mécanismes de décision. Antérieurement, l'utilisateur devait soit, quantifier lui-même l'importance relative entre les critères, soit faire selon son intuition. Notre algorithme de pondération nous permet de le faire automatiquement. En ce qui concerne (2), nous avons également proposé un algorithme qui a pour but d'identifier les alternatives (les situations potentielles à exécuter) pour la prise de décision. Notre proposition offre comme avantages, d'une part la réduction du temps de calcul pour les algorithmes de décision et, d'autre part, le complément pour avoir un processus de décision automatique. Pour (3), les différentes approches proposées sont détaillées dans la section suivante.

Est-ce que le choix final tient compte des différentes logiques de l'application : celle du concepteur, de l'utilisateur et du système ?

Nous avons modélisé le choix final du processus d'aide à la décision multicritère à base de traces par la combinaison des choix de différentes logiques : celle du concepteur, celle de l'utilisateur et celle du système. Pour le choix du système, nous avons réutilisé la méthode PROMETHEE II classique en ajoutant les traces afin de surmonter l'inconvénient de l'intuition de l'utilisateur dans sa première étape du calcul. Cette nouvelle approche s'appelle PROMETHEE II à base de traces. Pour le choix de l'utilisateur, une approche s'appuyant sur la Logique Subjective à base de traces a été proposée afin de modéliser les préférences de l'utilisateur lorsqu'il veut faire un choix. Cette proposition utilise le principe fondamental de la logique subjective associée à l'intégration de traces pour arriver à construire les préférences de l'utilisateur. Finalement, nous avons appliqué la méthode d'agrégation Borda pour combiner les 3 choix en un seul.

7.1.3 APPORTS EXPERIMENTAUX

Quelle étude de cas choisirons-nous pour illustrer nos contributions ? Comment évaluer nos contributions ?

En parallèle du travail théorique, nous avons conçu et développé une étude de cas, le Tamagotchi, visant à nous permettre de valider toutes nos contributions. L'application Tamagotchi a été développée de manière à respecter le formalisme des situations. Le prototype de Tamagotchi a été testé par plusieurs utilisateurs afin d'évaluer les performances de notre processus d'aide à la décision multicritère. Les traces générées lors de l'exécution sont stockées dans une base de traces afin de pouvoir être réutilisées dans notre mécanisme de décision. Nous avons testé les contributions sur plusieurs aspects tels que : le temps de calcul, la performance, la satisfaction de l'utilisateur. Malgré la simplicité du prototype, la

performance de notre processus d'aide à la décision multicritère à base de traces a montré son efficacité.

7.1.4 PUBLICATIONS

Nos travaux ont donné lieu à plusieurs publications scientifiques dont voici le récapitulatif.

Contribution sur la pondération des critères :

- Conférence internationale : ICCRD 2014 (International Conference on Computer and Development 2014) [83] : article présentant notre approche de pondération des critères dans la prise de décision multicritère, qui est retenu aussi dans une revue internationale.
- Revue internationale : Journal of Software [84].

Contribution sur la détermination des alternatives :

- Conférence internationale : CoDIT 2014 (IEEE International Conference on Control, Decision and Information Technologies) [85] : article présentant une partie de l'approche de détermination des alternatives pour la prise de décision.
- Revue internationale : RAIRO Operations Research (en cours d'expertise) : version étendu de celui de la conférence CoDIT.

Contribution sur la prise de décision multicritères :

- Conférence internationale : LPAR 2015 (International Conference on Logic Programming, Artificial Intelligence and Reasoning) [97] : article présentant la modélisation du choix de l'utilisateur en utilisant la logique subjective à base de traces.
- Conférence nationale :
    - 16e Conférence ROADEF (Société Française de Recherche Opérationnelle et Aide à la Décision) [98]: article court présentant la description de la méthode PROMETHEE II à base de traces.
    - 7e Conférence sur les Environnements Informatiques pour l'Apprentissage Humain [99] : article court (retenu comme poster) présentant l'application de la méthode PROMETHEE II à base de traces pour prendre de décision dans les environnements d'apprentissage.

7.2 LIMITES

Le travail mené dans cette thèse comporte des limites dont certaines ont été évoquées tout au long de ce document. Il nous paraît important cependant de revenir sur les plus importantes d'entre elles.

7.2.1 PERIMETRE DE NOS PROPOSITIONS

Nos contributions devraient être applicables quel que soit le type d'application interactive à base de situations. Cette approche peut toutefois être une contrainte et ne pas permettre une transposition directe de nos travaux sur d'autres types de systèmes. En effet, changer de

représentation du système nécessiterait de revoir le modèle de traces utilisé, ainsi que le processus d'aide à la décision associé.

7.2.2 QUANTITE DE TRACES

« Cette approche a besoin d'un nombre de traces qui est assez grand ». Nous avons été confronté de nombreuses fois à cette remarque lors des différentes relectures et échanges autour de nos travaux. Ceci représente une limite de notre approche. Puisque les calculs sont basés sur l'analyse de traces générées lors des exécutions passées, la taille de la base de traces obtenue est une donnée très sensible. La quantité de traces disponibles influence directement la mise en œuvre des algorithmes et par conséquent la qualité du choix final.

7.2.3 MONTEE EN CHARGE

Nous avons testé nos propositions sur le Tamagotchi qui ne comporte que 7 situations et un vecteur d'état contenant une dizaine d'attributs. Nous avons remarqué que le temps de calcul est compatible avec une exécution en temps réel. Pourtant, nous n'avons pas encore testé nos approches sur une application avec un très grand nombre de situations et un vecteur d'état de grande dimension (beaucoup d'attributs à considérer). Évidemment, le temps de calcul va devenir un point sensible, et s'attaquer aux applications plus complexes est un enjeu important. .

7.3 PERSPECTIVES

À court terme, les perspectives de notre travail s'inscrivent dans les pistes ouvertes par la recherche sur l'aide à la décision multicritère et l'utilisation des traces dans la mise en œuvre des mécanismes d'adaptation dans une application interactive. À plus long terme, les perspectives concernent l'amélioration des performances de nos contributions.

7.3.1 À COURT TERME

7.3.1.1 Amélioration de la modélisation des choix par différentes mesures

Nous aimerions avoir une modélisation du choix conceptuel proposé. En effet, dans notre méthodologie, nous nous concentrons sur les algorithmes et les approches du choix optimal à partir des traces. Nous n'avons pas abordé le problème du point de vue de la structure de l'application, ce qui pourrait nous permettre d'affiner l'ensemble des situations candidates. La scénarisation d'une application au moyen des situations nous mènera à deux mesures qui pourraient être intégrées dans notre contexte :

- Mesure de correspondance

La mesure de correspondance est une valeur basée sur la comparaison entre le vecteur d'état du système et la pré-condition de chaque situation candidate. Toutes les pré-conditions de chaque situation candidate sont déjà vérifiées avec le vecteur d'état lors de la phase de

détermination des situations candidates. Notre travail repose sur le problème de classer les situations candidates par ordre de priorité en fonction des pré-conditions et du vecteur d'état du système. Autrement dit, nous cherchons à évaluer le niveau de correspondance entre le vecteur d'état et chacune de pré-condition. Le calcul de cette mesure pourrait être basé sur les techniques de calcul de dissimilarité. Elles permettent d'estimer de manière approximative le niveau de similarité/dissimilarité entre le vecteur d'état et les pré-conditions. En nous basant sur cette mesure, nous pourrons ordonner les situations entre elles.

- Mesure d'expérience

Afin d'améliorer la sélection des situations candidates, nous introduisons une mesure appelée *expérience* qui représente les tendances des utilisateurs dans les exécutions précédentes. Cette mesure prend en compte : i) la fréquence d'une transition entre deux situations données dans les choix précédents des utilisateurs, et ii) les valeurs des critères à la fin d'une exécution pour vérifier si la stratégie de cette exécution permet d'atteindre de bonnes valeurs des critères. Ainsi, nous disposerons de deux mesures pour classer les situations candidates.

7.3.1.2 Développement d'un jeu sérieux basé sur le Tamagotchi

Nous allons poursuivre le développement du Tamagotchi afin d'en faire un jeu à part entière. Nous gardons la même architecture à base de situations en ajoutant les différentes fonctionnalités. L'objectif est de disposer d'un jeu réellement interactif (permettant aux joueurs d'interagir avec le système de jeu) et adaptatif (aidant les joueurs à adapter le déroulement du jeu pour atteindre l'objectif du jeu). Nous pensons aussi à intégrer un objectif d'apprentissage dans le jeu, tel que : la situation « Apprendre ». Nous allons ainsi concevoir des règles, des interfaces qui permettent au Tamagotchi/joueur d'acquérir des compétences élémentaires par la réalisation des petites missions.

7.3.1.3 Application dans le cas de e-Education

Par ailleurs, nous sommes en train de poursuivre nos travaux en étudiant l'apport de notre approche dans les systèmes de e-Learning. Les principes abordés sont aisément transposables au domaine de l'apprentissage en ligne où la notion de scénarisation pédagogique consiste en une succession d'activités / situations contextualisées (cours, travail dirigé, travail pratique, etc.) comme le montre la Figure 49.

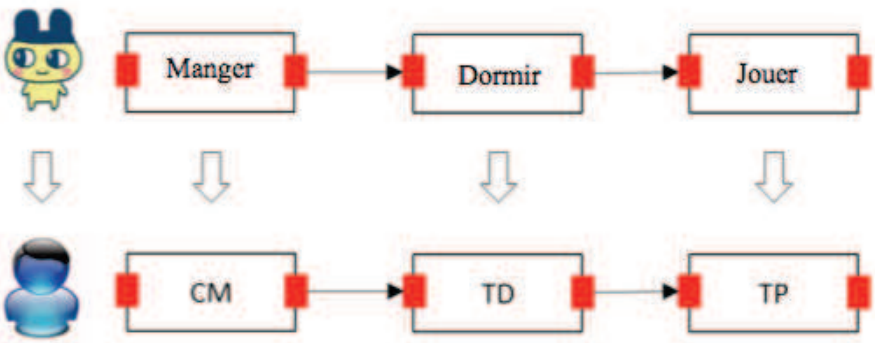

Figure 49. Transposition au cas e-Education

Comparativement à la vie du Tamagotchi, un cours d'apprentissage est vu comme une succession de situations contextualisées. Le déroulement d'un cours est composé de succession de situations générées par l'enseignant et les apprenants en visant les objectifs qui ont été définis par le concepteur. Nous considérons qu'un cours dans le système e-Learning est semblable à un cours classique. Il est composé, par exemple d'une présentation de l'enseignant, des exercices à travailler, des discussions, des tests, etc. Nous avons donc conçu sept situations qui seront intégrées à la plateforme POLARIS[11] [101], qui est un prototype de plateforme d'apprentissage en cours de développement dans notre laboratoire. Les situations sont :

- Situation Nouveau Cours
- Situation Rappel
- Situation Travail Individuel
- Situation Travail Groupe
- Situation Passage au Tableau
- Situation Discussion
- Situation Test

La description de ces situations est présentée dans Annexe C.

Lors de l'exécution d'un cours, nous avons conçu un vecteur d'état permettant de regrouper l'ensemble des états du système, de l'enseignant et des apprenants. Ce vecteur est composé de 42 attributs qui est décrite dans Annexe D.

Grâce aux mécanismes des situations, notre objectif est d'aider l'enseignant à choisir une situation qu'il peut exécuter à un moment donné dans le cours. Le choix est basé sur notre processus d'aide à la décision à base de traces. Une méthode de décision basée sur les expériences passées pourrait permettre de mieux conduire la séquence de situations.

7.3.2 POURSUITE DE LA RECHERCHE

7.3.2.1 Mise en place d'une base de traces pour tester les mécanismes de décision

Nous avons eu l'opportunité de rencontrer la communauté des chercheurs qui est en train de construire une base de données standard pour tester les mécanismes de décision multicritère existant dans la littérature. Nous avons l'idée de modifier les traces en tenant compte du modèle standard disponible pour reconstruire une nouvelle base de données afin de contribuer à la communauté. Nous allons notamment utiliser les données issues de l'étude du Tamagotchi actuel pour mettre en œuvre cette contribution. La prochaine version du Tamagotchi permettra de collecter de nombreuses traces pour construire la base.

7.3.2.2 Amélioration de la performance de notre processus d'aide à la décision multicritère à base de traces

Dans une de nos contributions, nous avons dit que « Les nouvelles approches d'aide à la décision à base de traces ont montré notamment leurs performances pour les utilisateurs

[11] http://foad-l3i.univ-lr.fr/portail/index.php

débutants, pourtant pour les utilisateurs qui ont de l'expérience, le résultat ne leur convient pas toujours ». Dans la suite de notre recherche, nous souhaitons combler cette lacune.

Nous envisageons d'appliquer des méthodes de recommandation actuellement utilisées principalement pour la recommandation de contenus sur des environnements en ligne, pour traiter le problème abordé. Dans notre système, nous caractérisons un utilisateur par son profil. Ce profil est décrit explicitement dans le paragraphe 4.1.3 mais nous ne proposons pas un outil ou une approche permettant d'obtenir un profil d'utilisateur complet caractérisé de manière formelle. Ceci permettra d'identifier quels sont les profils les plus proches d'un utilisateur donné, les profils les plus proches pouvant être extraits de son réseau social (contacts, contexte,...). Il existe de nombreuses techniques qui pourraient permettre de résoudre ce problème, comme par exemple les méthodes de calcul de similarité entre deux utilisateurs.

7.3.2.3 Gestion de grandes masses de données (big data)

Dans cette thèse, nous n'avons considérons que les petites bases de données (en nombre de traces et en quantité d'informations par trace). C'est pour cela que le stockage et l'extraction de ces données ne posent pas de réel problème. Pourtant, l'application pourra être utilisée par plusieurs utilisateurs, il y aura évidemment une augmentation notable du nombre de traces générées en fonction de temps. Par ailleurs, avec des applications de grande dimension, nous aurons besoin de techniques plus évoluées pour gérer les traces collectées. Comment gérer un modèle de traces pertinent ? Comment stocker les traces dans une base en assurant la vitesse d'extraction ? Les techniques d'indexation dans la fouille de données nous permettent d'arriver à indexer une grande base de données et le résultat d'indexation sera utile pour l'analyse ou l'extraction des traces ultérieurement.

Une des applications possibles pourrait concerner les systèmes de gestion des MOOC (Massive Open Online Course). Les MOOC par leur côté massif s'adressent à un très grand nombre d'apprenants (plusieurs centaines de milliers pour certains MOOC). Dans ces conditions, envisager de disposer d'un système de scénarisation des situations d'un MOOC prenant en compte une telle population va nous demander de nous tourner vers des techniques de « big data ».

Annexes

Annexe A. PROMETHEE II à base de traces

La Figure 50 présente la méthode PROMETHEE II à base de traces. Elle combine celui de PROMETHEE II classique et notre proposition à base de traces qui remplace la première étape de cette méthode.

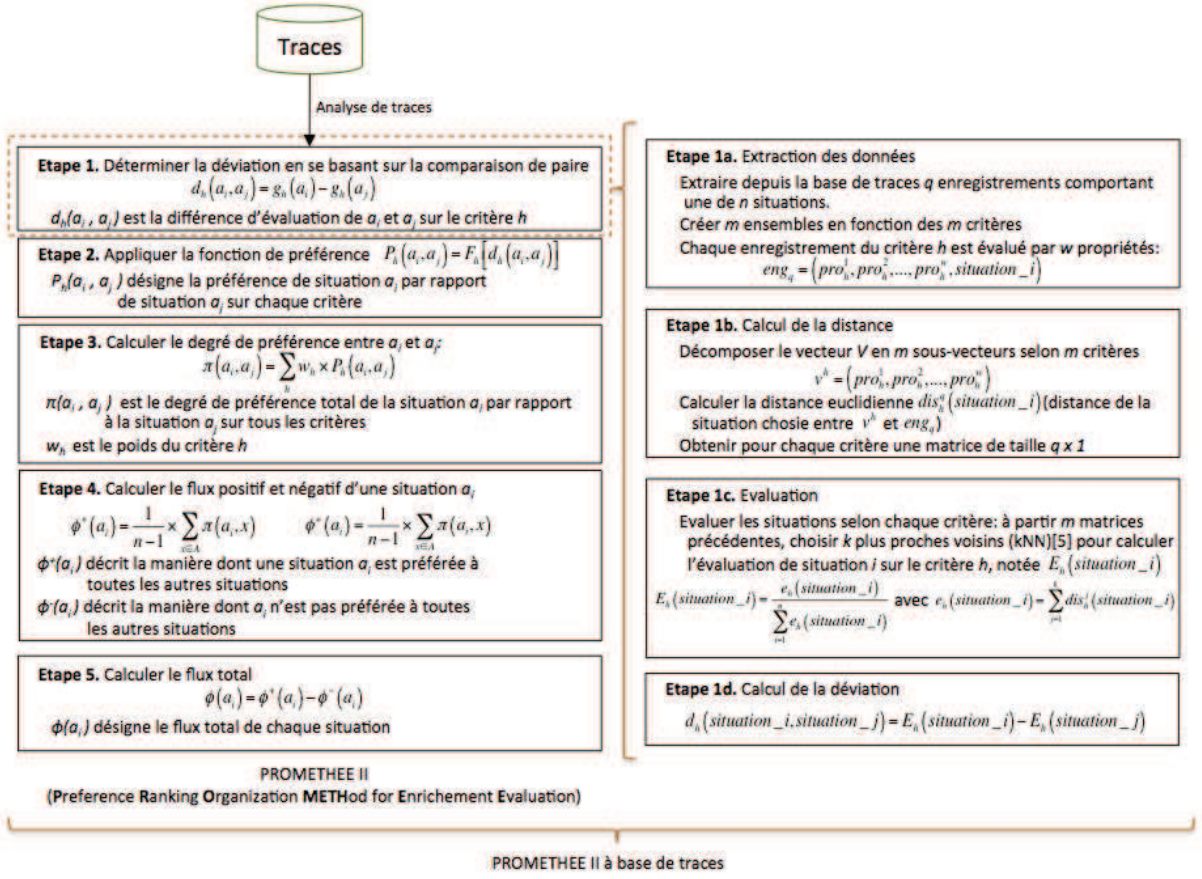

Figure 50. Description de la méthode PROMETHEE II à base de traces

Nous n'utilisons que les traces dans la première étape qui consiste à déterminer la déviation d'évaluation entre deux alternatives en fonction de chaque critère. La méthode commence par la partie droite de la Figure 50, puis reprend le principe classique pour le calcul du choix. Classiquement, PROMETHEE II a besoin de l'intervention de l'utilisateur pour calculer la déviation présentée. Cette proposition supporte un déroulement automatique pour la prise de décision lors de l'exécution d'une application interactive à base de situations.

Nous avons conçu un système Tamagotchi qui est simple et composé des éléments qui nous permettent de montrer les contributions de la thèse. Le système exige que l'utilisateur courant remplisse les informations qui lui sont demandées avant d'entrer dans l'application principale. L'interface de cette partie est déjà présentée par la Figure 38. Quand l'utilisateur finit le remplissage, notre interface principale apparaît en donnant le détail de la pondération des critères comme le montre la Figure 51.

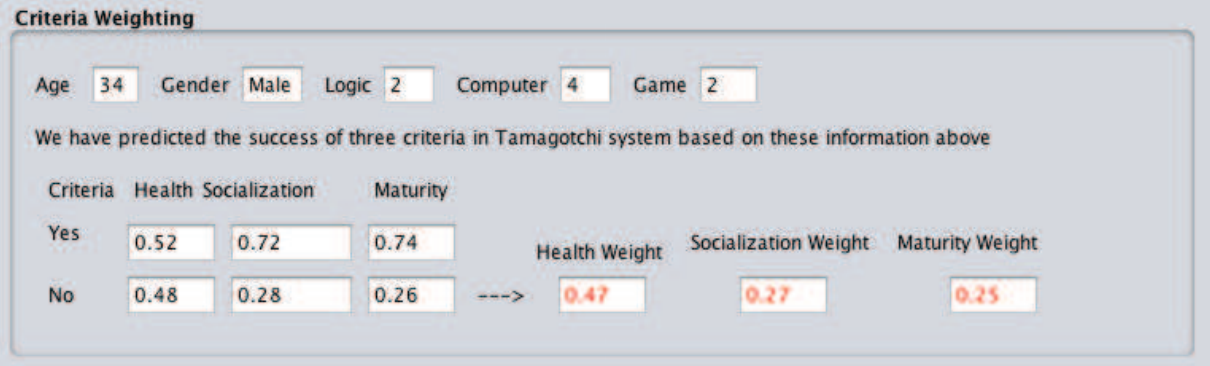

Figure 51. Détail de la pondération des critères dans le système Tamagotchi

La Figure 52 présente l'interface de l'utilisateur de l'étude de cas Tamagotchi. Elle est conçue avec l'IDE de Netbeans. Nous disposons de trois composants :

- Informations générales : il s'agit de l'identifiant du Tamagotchi et de son âge à un instant donné de l'exécution du système. Par ailleurs, il y a aussi les valeurs des trois critères (la Santé, la Socialisation, la Maturité) permettant à l'utilisateur de savoir la valeur atteinte pour chaque critère. Les deux boutons qui suivent ont deux fonctions : celle de déterminer les situations candidates et celle de prendre la décision à base de traces.
- Environnement : il représente simplement le Tamagotchi dans son environnement. Il est situé au milieu des sept situations qui sont placées autour de lui afin d'illustrer l'émergence des situations. Il n'y a aucun lien défini par le concepteur de l'application.
- Exécution : elle sert à visualiser l'enchainement des situations réalisées lors de l'exécution de l'application.

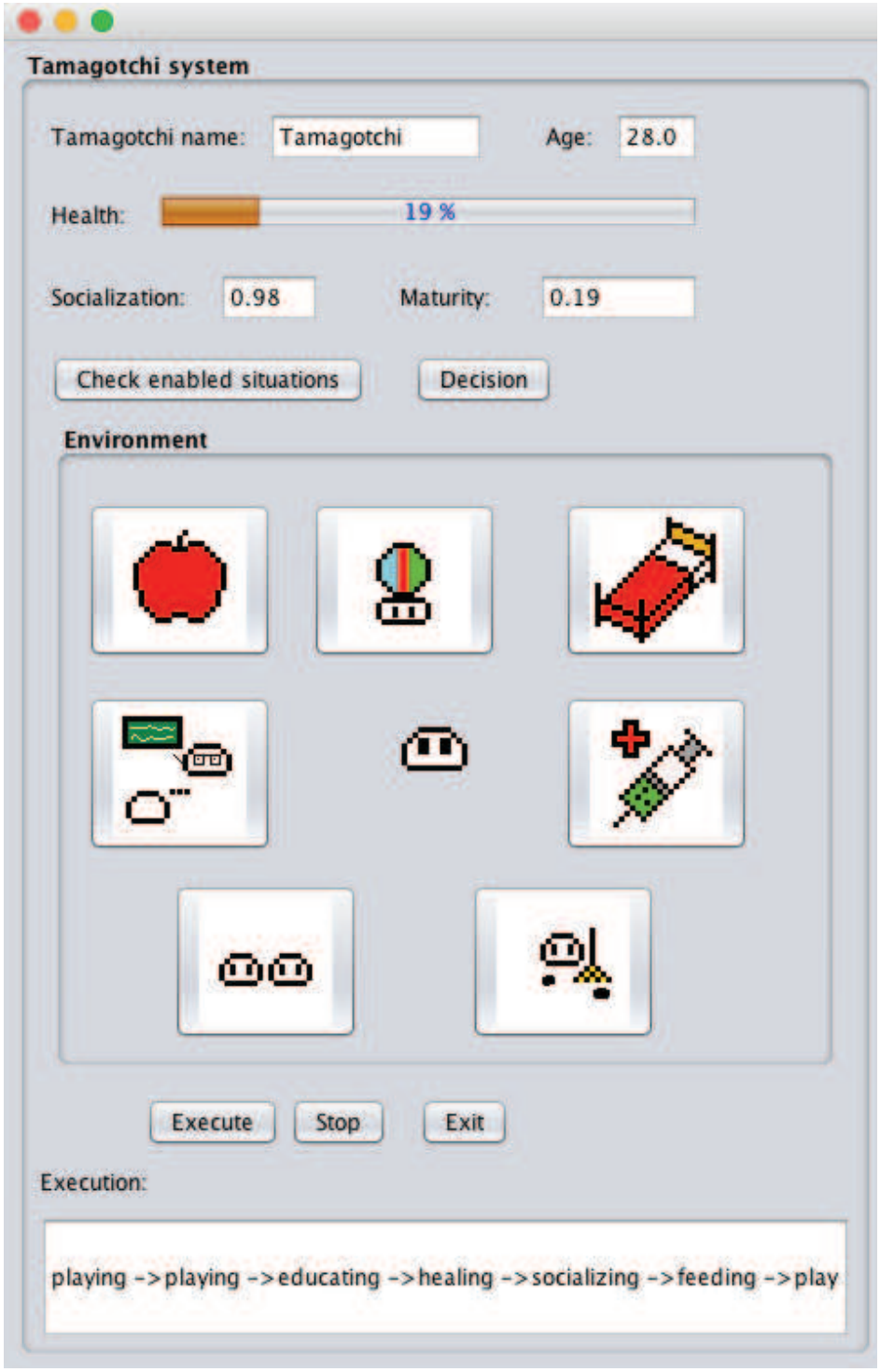

Figure 52. L'interface du système Tamagotchi

Concernant le bouton de la prise de décision, nous disposons de trois principes : celui de l'utilisateur, celui du concepteur et celui du système. L'utilisateur peut choisir une ou plusieurs logiques comme il veut. S'il choisit au moins deux logiques, le système doit combiner les différentes logiques en appliquant la méthode d'agrégation présentée dans 4.4.4.

Annexe C. Description les situations d'apprentissages dans le cas de e-Learning

Nous présenterons ci-dessous une description détaillée des pré-conditions et post-conditions de chaque situation.

Avant d'analyser les conditions des situations, nous supposons que tous les étudiants qui vont participer ne font pas encore ce cours, c'est-à-dire le niveau des étudiants concernant le cours est au niveau débutant. Des conditions requises pour commencer le cours :

- Terminaison du programme du cours : faux
- Terminaison du travail : faux
- Terminaison du passage au tableau : faux
- Terminaison de la discussion : faux
- Terminaison du test : faux

Il existe quelques attributs dans la post-condition que nous mettons en rouge, cela signifie si cette condition est satisfaite, la situation est finie et une autre situation va être exécutée.

Situation	**Nouveau Cours**
Pré-condition	• Terminaison du programme du cours : faux • Présence de l'enseignant : vrai • Nombre des étudiants présents actuels : 70% nombre défini par le concepteur • Présence des étudiants participants au cours : vrai • Durée du cours prévue < Durée du cours restante • Exécution initiale du cours : vrai • Nombre de fois d'exécution du cours : 0 • Disponibilité des diapositives à présenter : vrai
Post-condition	• Durée du cours dépasse la durée maximale

Situation	**Rappel**
Pré-condition	• Terminaison du programme du cours : vrai • Présence de l'enseignant : vrai • Nombre des étudiants présents actuels : 70% nombre défini par le concepteur • Présence des étudiants participants au cours : vrai • Durée prévue < Durée totale • Exécution initiale du cours : faux • Nombre de fois d'exécution du cours != 0 • Disponibilité des diapositives à présenter : vrai
Post-condition	• Durée du rappel dépasse la durée maximale

Situation	**Travail individuel**
Pré-condition	• Présence de l'étudiant : vrai • Terminaison du programme du cours : vrai • Durée du travail < durée restante • Etat des ressources associées au travail = disponible (à voir)

	• Etat des outils permettant à travailler = disponible (à voir)
Post-condition	• Nombre de questions posées > 10 • Durée du travail dépasse la durée maximale

Situation	**Travail en groupe**
Pré-condition	• Présence des étudiants : vrai • Terminaison du programme du cours : vrai • Durée du travail < durée restante • Etat des ressources associées au travail = disponible (à voir) • Etat des outils permettant à travailler = disponible (à voir)
Post-condition	• Nombre de questions posées > 10 • Durée du travail dépasse la durée maximale

Situation	**Passage au tableau**
Pré-condition	• Terminaison du travail : vrai • Présence du présentateur • Durée du passage au tableau < durée restante
Post-condition	• Nombre de questions posées > 10 • Durée du passage dépasse la durée maximale

Situation	**Discussion**
Pré-condition	• Présence du présentateur • Durée de la discussion < durée restante
Post-condition	• Nombre de questions posées > 10 • Durée du discussion dépasse la durée maximale

Situation	**Test**
Pré-condition	• Présence des étudiants • Durée limite de test < durée restante • Terminaison du programme du cours : vrai • Terminaison du travail : vrai • Etat des tests : = disponible
Post-condition	• Nombre de questions posées > 10 • Durée du test dépasse la durée maximale

Des conditions pour terminer le cours :

- Terminaison du programme du cours : vrai
- Terminaison du travail : vrai
- Terminaison du test : vrai
- Dépasser le temps défini par le concepteur.

Annexe D. Description le vecteur d'état dans le cas de e-Learning

Attribut	Type
Présence de l'enseignant	Booléen
Présence des étudiants	Booléen
Nombre minimum des étudiants dont il a besoin pour commencer le cours (paramètre défini par le concepteur)	Numérique
Nombre des étudiants présents actuel	Numérique
Durée du cours prévue (en minutes)	Numérique
Durée du cours estimée (en minutes)	Numérique
Durée du cours restante (en minutes)	Numérique
Exécution initiale du cours	Booléen
Rappel du cours	Booléen
Approfondissement du cours	Booléen
Nombre de fois d'exécution du cours	Numérique
Nombre des diapositives présentées dans le cours	Numérique
Nombre de la diapositive courante (variable interne)	Numérique
Nombre d'étudiants attentifs	Numérique
Terminaison du programme du cours (fini ou non fini)	Booléen
Nombre de question	Numérique
Durée du travail (en minutes)	Numérique
Nombre d'exercices proposés par l'enseignant	Numérique
Nombre d'exercices faits par l'étudiant	Numérique
Nombre d'exercices corrects par l'étudiant	Numérique
Durée estimée de l'exercice (en minutes)	Numérique
Difficulté de l'exercice (une valeur de 0 : très facile à 5 : très difficile) (état lié à l'exercice)	Numérique
Position de l'exercice dans le cours (variable interne)	Chaine
Type de l'exercice (long/ court) (état lié à exercice)	Chaine
Compréhension du cours de l'étudiant (une valeur de 0 : non compris à 5 : totalement compris) (variable interne)	Numérique
Terminaison du travail	Booléen
Durée du passage au tableau (en minutes)	Numérique
Durée limite du passage au tableau (en minutes)	Numérique
Résultat du travail au tableau (pourcentage de justesse)	Numérique

Terminaison du passage au tableau (fini ou non fini)	Booléen
Nombre de question restants	Numérique
Durée limite d'une discussion (en minutes)	Numérique
Nombre nouvelles questions	Numérique
Présentateur du passage au tableau (enseignant, étudiant)	Chaine
Durée de la discussion (en minutes)	Numérique
Durée limite de test (en minutes)	Numérique
Nombre de tests proposés par l'enseignant	Numérique
Nombre de tests faits par l'étudiant	Numérique
Nombre de tests corrects par l'étudiant	Numérique
Niveau de difficulté de test (une valeur de 0 : très facile à 5 : très difficile)	Numérique
Résultat du test (pourcentage de correction)	Numérique
Terminaison du test (fini ou non fini)	Booléen

Annexe E. Définition des notions liées au concept « situation »

Nous présenterons ci-dessous les définitions des composants d'une situation élémentaire comme le montre dans la Figure 3.

- Pré-condition : est un ensemble de conditions devant être vérifiées afin de commencer une nouvelle situation. Ce sont des valeurs pré-requises quand le système veut suivre la logique d'exécution du concepteur. Une pré-condition est constituée d'un ensemble de conditions basiques. Chacune porte une valeur qui permet de quantifier ou mesurer les états des participants contribuant dans cette situation ainsi que l'état du système.
- Post-condition : est le résultat attendu après avoir exécuté la situation. Elle représente les conditions de sortie d'une situation. Au cours de l'exécution de la situation, les états des participants ou du système changent effectivement. Lorsque les conditions sont vérifiées la sortie de la situation permet d'enregistrer tous les changements survenus au niveau des valeurs des états impliqués. La post-condition est organisée aussi en blocs indépendants de conditions basiques.
- Contexte global : il représente une vision globale en se basant sur l'état du monde externe (environnement, ressources), l'état des acteurs et leurs profils associés.
- Déroulement : le déroulement d'une situation représente un cadre pour les interactions et événements déclenchés par les acteurs en utilisant les ressources en vue des objectifs. Les sous-composants du déroulement sont :
    - Acteurs : soit les entités représentant l'utilisateur humain (les joueurs) soit les entités représentant les agents automatisés du système (les personnages non joueurs, les composants du système, *etc*. ;
    - Logique du concepteur : représente la logique de conception qui se compose des règles de déroulement d'une situation et l'enchainement des situations pour construire un scénario tout au long de la trame scénaristique.
    - Ressources : tous les objets informatiques matériels ou logiciels nécessaires à l'exécution d'une situation ou plus globalement dans une application. Les travaux sur la ressource sont aussi traités par une autre thèse au sein de notre équipe [78]. Par ailleurs, d'après [102], une ressource peut être soit :
        - locale : appartient à ou utilisée par un seul acteur ;
        - globale : partagée par plusieurs acteurs et support des interactions.
- Gestionnaire d'événement : il gère les événements qui arrivent de l'extérieur de la situation, par exemple ceux qui arrivent depuis l'environnement ou depuis d'autres situations exécutées en parallèle. Ces événements ne sont pas des conditions pour réaliser une situation, mais ils influencent les interactions des acteurs.
- Gestionnaire des ressources locales : il prend en charge l'accès et l'utilisation des ressources locales de la situation en tenant compte des demandes des acteurs. Les transactions de l'utilisation des ressources doivent passer par ce gestionnaire.
- Gestionnaire des ressources globales : il est similaire au gestionnaire des ressources locales mais il prend en charge la gestion de plus haut niveau. Il gère les ressources ainsi que les transactions liées aux ressources globales.

Après la vérification des pré-conditions et l'accès dans la situation, chaque acteur dans une situation interagit avec les autres en réalisant les actions afin d'atteindre des objectifs de situation ainsi que celui de l'application. Les acteurs emploient les ressources locales et globales pour réaliser leurs activités. Ils doivent respecter la logique définie par le concepteur en effectuant les règles prédéfinies. Les interactions entre les acteurs font progresser le

déroulement de la situation jusqu'à la satisfaction des post-conditions. L'acteur peut choisir de quitter la situation ou de rester encore plus long pour continuer à l'exécuter.

Nous avons évoqué la situation composée dans la section 1.3, nous présentons maintenant la description détaillée de la situation composée.

D'après la thèse de Pham [1], une situation composée est un regroupement de situations d'un point de vue d'abstraction supérieur, selon un contexte global en partie commun entre les situations regroupées. En d'autre terme, « *une situation composée est un assemblage de situations permettant une structuration récursive selon une granularité supérieure à celle des situations élémentaires* » comme la montre la Figure 53.

Une situation composée contient aussi certains éléments présents dans la situation élémentaire mais ils sont plus larges et plus généraux :

- Pré-condition : l'état global du système doit vérifier un ensemble de conditions avant d'entrer dans la situation composée ;
- Post-condition : les conditions doivent être remplies avant de quitter la situation composée ;
- Logique d'enchainement : elle permet d'ordonner des situations composantes de la situation composée de son/ses état(s) initial/initiaux jusqu'à son/ses état(s) final/finaux.

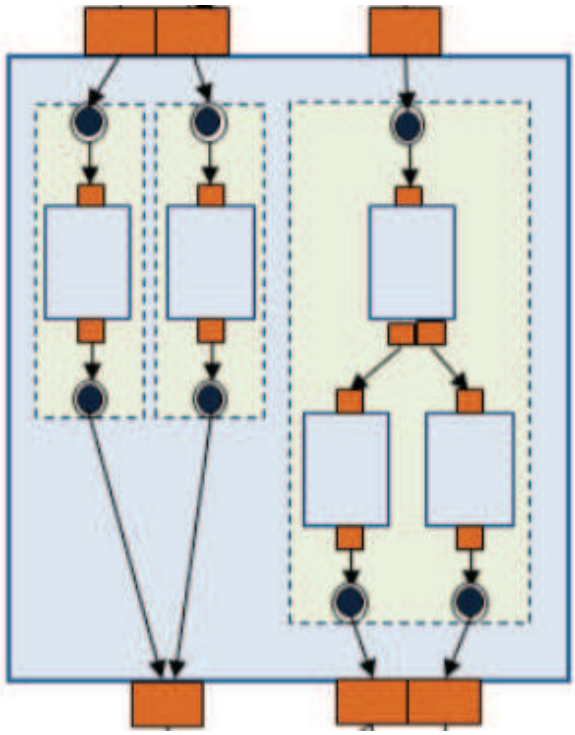

Figure 53. Situation composée [1]

Les situations sont positionnées comme un graphe. À la fin d'une situation, il peut y avoir par exemple, le choix entre plus d'une situation à suivre. L'utilisateur peut faire des choix entre plusieurs situations possibles afin de poursuivre l'exécution. La Figure 54 montre un petit exemple de graphe de situation qui a été extrait depuis le cas d'étude dans la thèse de Pham.

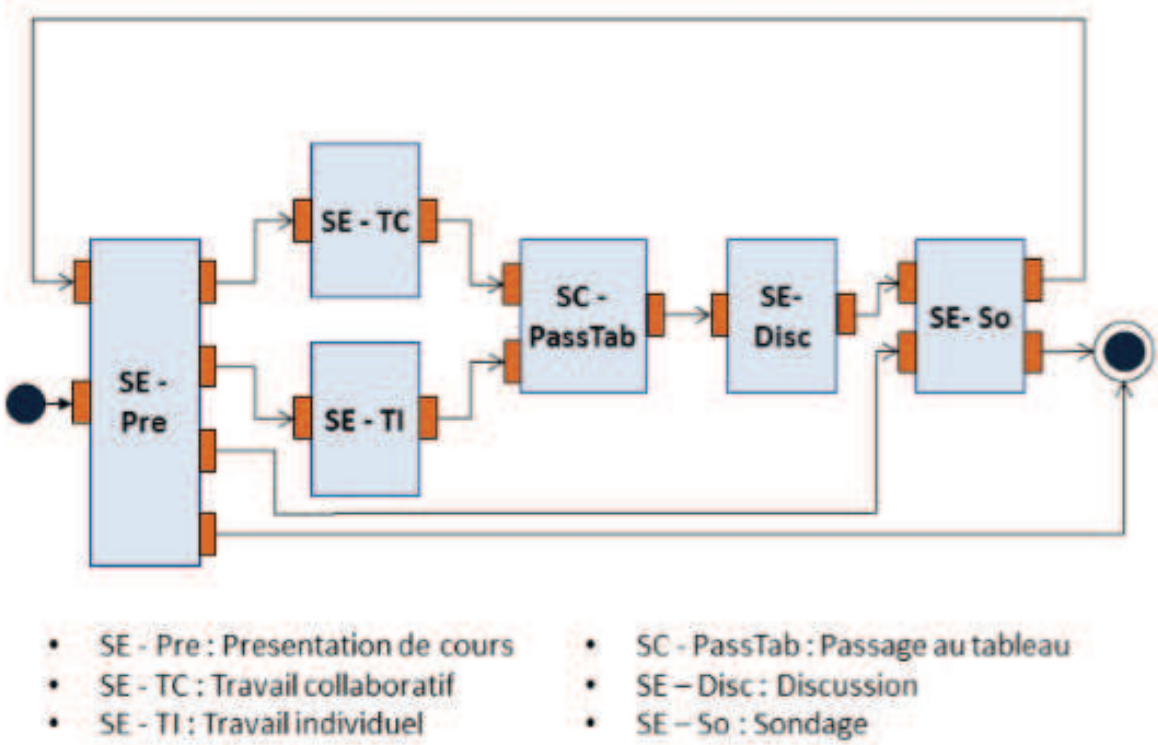

Figure 54. Extrait d'un exemple d'un graphe de situation dans le cas d'étude de Pham [1]

Dans la thèse de Dang [9], l'auteur a fourni un modèle permettant d'aider à la réalisation de scénario interactif qui représente la logique du concepteur en fonction de contexte, des ressources et du comportement de l'utilisateur. Sa proposition qui vise à produire un modèle de scénario de bonne qualité qui soit : riche (le scénario fournit suffisamment d'options pertinentes aux utilisateurs de sorte qu'ils puissent déterminer le déroulement de l'application), valide (tous les possibilités dans le scénario sont cohérentes et répondent aux effets désirés des auteurs), opérationnel (le scénario est exécutable). Ce scénario est utilisé dans le système de pilotage de l'exécution de l'application. Le processus de scénarisation en scénario est effectué lors de la phase de conception de l'application et non pas lors de son exécution en temps réel.

L'auteur a appliqué la logique linéaire pour modéliser le scénario et pour vérifier sa qualité en tenant compte des propriétés abordées ci-dessus. En employant la logique linéaire, l'auteur a proposé une solution qui a été composé de deux étapes :

- Modéliser le scénario en logique linéaire ;
- Évaluer la qualité du scénario en parcourant tous les discours possibles et en analysant les propriétés espérées du scénario.

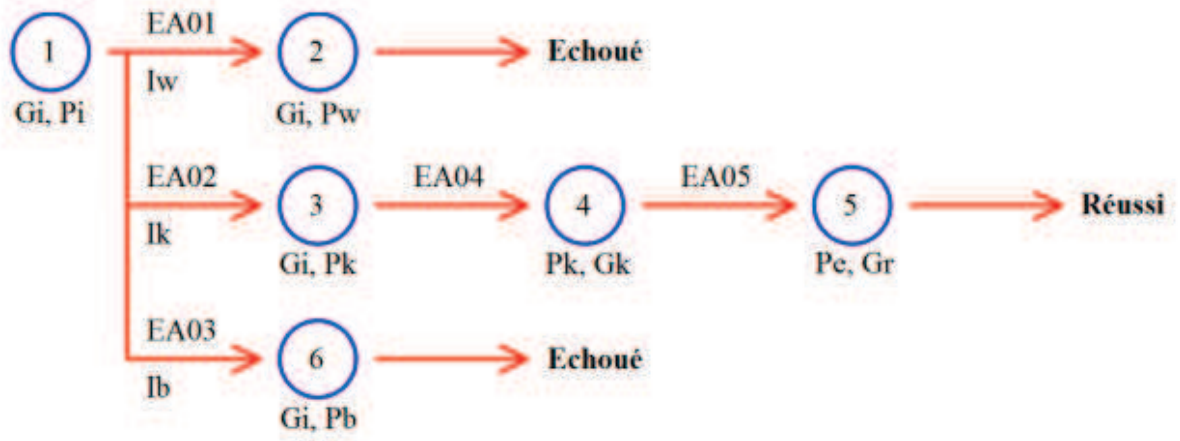

Figure 55. Représentation d'un scénario par l'application la logique linéaire [9]

La Figure 55 décrit un graphe représentant un scénario modélisé par la logique linéaire. Chaque nœud correspond aux états disponibles de l'application à un moment (l'état de l'utilisateur, du système, des ressources, *etc*). Chaque arc représente une action exécutée.

Le scénario garantit les souhaits des utilisateurs mais il ne donne pas exactement l'action appropriée à l'utilisateur. Même si l'utilisateur a la liberté de choisir ce qu'il veut exécuter, c'est mieux de lui suggérer une solution pertinente. Le résultat comme la montre la Figure 55 est celui du concepteur de l'application. Il est fixe et difficile à réorganiser lors de l'exécution de l'application en raison des aléas ou des erreurs non prévisionnelles. Pour cela il faut mettre en place des mécanismes d'adaptation de l'exécution afin de faire le choix parmi les actions possibles pour pouvoir aider l'utilisateur à finir l'application.

Références

1. Pham, P.T.: Thèse de doctorat “Architecture à base de situations pour le traitement des quiproquos dans l’exécution adaptative d’applications interactives,” (2013).

2. Kiousis, S.: Interactivity: a concept explication. New Media Soc. 4, 355–383 (2002).

3. Jameson, A.: Systems that adapt to their users: An integrative perspective. (2000).

4. Paramythis, A.: Thèse de doctorat “Adaptive Systems: Development, Evaluation and Evolution,” (2009).

5. Sehaba, K.: Thèse de doctorat “Exécution adaptative par observation et analyse de comportements - Application à des logiciels interactifs pour des enfants autistes,” (2005).

6. Pham, P.T., Rabah, M., Estraillier, P.: A Situation-Based Multi-Agent Architecture for Handling Misunderstandings in Interaction. Int. J. Appl. Math. Comput. Sci. 25, 439–454 (2015).

7. Champagnat, R.: Habilitation à diriger des recherches “Contribution au pilotage de systèmes: des systèmes de production hybrides aux applications interactives scénarisées,” (2010).

8. Delmas, G.: Thèse de doctorat “Pilotage de récits interactifs et mise en oeuvre de formes narratives dans le contexte du jeu vidéo,” Université de La Rochelle, (2009).

9. Dang, K.D.: Thèse de doctorat "Aide à la réalisation de systèmes de pilotages de narration interactive: Validation d’un scénario basée sur un modèle en logique linéaire, (2013).

10. Alchiekh Haydar, C.: Thèse de doctorat “Les systèmes de recommandation à base de confiance,” (2014).

11. Béchet, N.: Etat de l’art sur les Systèmes de Recommandation, http://people.irisa.fr/Nicolas.Bechet/Publications/EtatArt.pdf.

12. Lops, P., de Gemmis, M., Semeraro, G.: Content-based Recommender Systems: State of the Art and Trends. In: Ricci, F., Rokach, L., Shapira, B., and Kantor, P.B. (eds.) Recommender Systems Handbook. pp. 73–105. Springer (2011).

13. Su, X., Khoshgoftaar, T.M.: A Survey of Collaborative Filtering Techniques. Adv. Artif. Intell. 2009, 4:2–4:2 (2009).

14. Ariely, D., Jr., J.G.L., IV., M.A.: Learning by Collaborative and Individual-Based Recommendation Agents. J. Consum. Psychol. 14, 81–95 (2004).

15. Ardissono, L., Goy, A., Petrone, G., Segnan, M., Torasso, P.: Intrigue: Personalized Recommendation of Tourist Attractions for Desktop and Hand Held Devices. Appl. Artif. Intell. 17, 687–714 (2003).

16. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. 17, 734–749 (2005).

17. Roy, B.: Méthodologie multicritère d'aide à la décision. , Paris (1985).

18. Grabisch, M.: Une approche constructive de la décision multicritère. Trait. du Signal. 22, 321–337 (2005).

19. Triantaphyllou, E., Shu, B., Sanchez, S.N., Ray, T.: Multi-Criteria Decision Making : An Operations Research Approach. Encycl. Electr. Electron. Eng. 15, 175–186 (1998).

20. Fishburn, P..: Additive Utilities with Incomplete Product Set: Application to Priorities and Assignments. Oper. Res. Soc. Am. 15, 537–542 (1967).

21. Bridgman, P.: Dimensional Analysis. , New Haven (1922).

22. Miller, D.W., Starr, M.K.: Executive Decisions and Operations Research. (1969).

23. Fishburn: Utility theory for Decision Making. (1970).

24. Hatami-Marbini, A., Tavana, M.: An extension of the Electre I method for group decision-making under a fuzzy environment. Omega. 39, 373–386 (2011).

25. J P, B., PH, V.: A Preference Ranking Organisation Method: (The PROMETHEE Method for Multiple Criteria Decision-Making). Manage. Sci. 31, 647–656 (1985).

26. Saaty, T.L.: How to make a decision: The analytic hierarchy process. Eur. J. Oper. Res. 48, 9–26 (1990).

27. Saaty, T.L.: Decision making with the analytic hierarchy process. Int. J. Serv. Sci. 1, 83–98 (2008).

28. Saaty, T..: Tha Analytical Hierarchy Process. (1980).

29. Laffy, R.: La méthode MARSAN pour la recherche produits nouveaux. Communication au congrès ESOMAR. , Copenhague (1966).

30. Benayoun, R., Roy, B., Sussman, B.: ELECTRE: Une méthode pour guider le choix en présence de points de vue multiples. (1966).

31. Buffet, P., Grémy, J., Marc, M., Sussmann, B.: Peut-on choisir en tenant compte de critères multiples? une méthode ELECTRE et trois applications. Rev. METRA. 283–316 (1968).

32. Roy, B., Skalka, J.M.: ELECTRE IS: Aspects méthodologiques et guide d'utilisation. (1984).

33. Roy, B., Bertier, P.: La méthode ELECTRE II - Une application au média-planning. In: M., R. (ed.) OR'72. pp. 291–302. North-Holland Publishing Company (1973).

34. Roy, B.: La méthode ELECTRE II. (1971).

35. Roy, B.: ELECTRE III : Un algorithme de classements fondé sur une représentation floue des préférences en présence de critères multiples. Cah. du CERO. 20, 3–24 (1978).

36. Roy, B., Hugonnard, J.C.: Ranking of suburban line extension projects of the Paris Metro System by a multicriteria method. Transp. Res. 16A, 301–322 (1982).

37. Roy, B.: From optimization to multicriteria decision aid: Three main operational attitudes. LNEMS 130, Springer-Verlag, Berlin (1976).

38. Vallée, D., Zielniewicz, P.: ELECTRE III-IV, version 3.x - Aspects méthodologiques. , Université de Paris-Dauphine, Document du LAMSADE no 85 (1994).

39. Roy, B.: A multicriteria analysis for trichotomic segmentation problems. In: Nijkamp, P. and Spronk, J. (eds.) Multiple Criteria Analysis: Operational Methods. pp. 245–257. Gower Press (1981).

40. Roy, B.: Présentation et interprétation de la méthode ELECTRE TRI pour affecter des zones dans des catégories de risque. (2002).

41. Brans, J.P.: L'ingénièrie de la décision; Elaboration d'instruments d'aide à la décision. La méthode PROMETHEE. In: Nadeau, R. and Landry, M. (eds.) L'aide à la décision: Nature, Instruments et Perspectives d'Avenir. pp. 183–213. Presses de l'Universit{é} Laval, Qu{é}bec, Canada (1982).

42. Brans, J.P., Mareschal, B., Vincke, P.: PROMETHEE: a new family of outranking methods in multicriteria analysis. In: Brans, J.P. (ed.) Operational Research, IFORS 84. pp. 477–490. North Holland, Amsterdam (1984).

43. Brans, J.P., Vincke, P., Mareschal, B.: How to select and how to rank projects: the PROMETHEE method. Eur. J. Oper. Res. 24, 228–238 (1986).

44. Brans, J.P., Mareschal, B.: Promethee-V - MCDM Problems with Segmentation Constraints. INFOR. 30, 85–96 (1992).

45. Brans, J.P., Mareschal, B.: The PROMETHEE VI procedure. How to differentiate hard from soft multicriteria problems. J. Decis. Syst. 4, 213–223 (1995).

46. Brans, J.-P., Mareschal, B.: Promethee Methods. Multiple Criteria Decision Analysis: State of the Art Surveys. pp. 163–186. Springer New York (2005).

47. Miettien, K.: Nonlinear multiobjective optimization. (1999).

48. Charnes, A., Cooper, W..: Management models and industrial applications of linear programming. (1961).

49. Dréo, J., Pétrowski, A., Siarry, P., Taillard, E..: Métaheuristiques pour l'optimisation difficile. (2003).

50. Aissanou, F.: Thèse de doctorat "Décisions multicritères dans les réseaux de télécommunications autonomes," (2012).

51. Lund, K., Mille, A.: Analyse de traces et personnalisation des environnements informatiques pour l'apprentissage humain. Presented at the (2009).

52. Settouti, L.S., Prié, Y., Marty, J.-C., Mille, A.: Vers des Systèmes à Base de Traces modélisées pour les EIAH . (2007).

53. Settouti, L.S.: Thèse de doctorat "Systèmes à Base de Traces Modélisées: Modèles et Langages pour l'exploitation des traces d'Interactions," (2006).

54. Settouti, L.S., Prié, Y., Marty, J.-C., Mille, A.: A Trace-Based System for Technology-Enhanced Learning Systems Personalisation. The 9th IEEE International Conference on Advanced Learning Technologies. pp. 93–97. , Riga, Latvia (2009).

55. Cram, D., Jouvin, D., Mille, A.: Visualisation interactive de traces et réflexivité: application à l'EIAH collaboratif synchrone eMédiathèque. Rev. STICEF. 14, 1764–7223 (2007).

56. Loghin, G.-C., Marty, J.-C.: Thèse de doctorat "Observer un Environnement Numérique de Travail pour réguler les activités qui s'y déroulent," (2008).

57. Settouti, L.S., Prié, Y., Mille, A., Marty, J.-C.: Système à base de trace pour l'apprentissage humain. colloque international TICE 2006 «Technologies de l'Information et de la Communication dans l'Enseignement Supérieur et l'Entreprise». , Toulouse, France (2006).

58. Laflaquière, J., Settouti, L.S., Prié, Y., Mille, A.: Un environnement pour gérer des traces comme inscriptions de connaissances. (2007).

59. Djouad, T.: Thèse de doctorat "Ingénierie des indicateurs d'activités à partir de traces modélisées pour un Environnement Informatique d'Apprentissage Humain," http://liris.cnrs.fr/publis/?id=5360, (2011).

60. Lund, K., Mille, A.: Traces, traces d'interactions, traces d'apprentissages: défintions, modèles informatiques, structurations, traitements et usages. , Lyon.

61. Djouad, T., Settouti, L.S., Prié, Y., Reffay, C., Mille, A.: Un Système à Base de Traces pour la modélisation et l'élaboration d'indicateurs d'activités éducatives individuelles et collectives. Mise à l'épreuve sur Moodle. TSI. 29, 721–741 (2010).

62. Sehaba, K.: Habilitation à diriger des recherches "Adaptation dynamique des Environnements Informatiques pour l'Apprentissage Humain interactifs," (2015).

63. Sehaba, K.: Partage d'expériences entre utilisateurs très différents: adaptation des modalités d'interaction. Ingénierie des connaissances. pp. 1–16. , Chambery (2011).

64. Clauzel, D., Sehaba, K., Prié, Y.: Modelling and visualising traces for reflexivity in synchronous collaborative systems. International Conference on Intelligent Networking and Collaborative Systems (INCoS 2009). pp. 16–23. IEEE Computer Society, Barcelona, Spain (2009).

65. Miura, M., Kunifuji, S., Sakamoto, Y.: Practical Environment for Realizing Augmented Classroom with Wireless Digital Pens. Knowledge-Based Intell. Inf. Eng. Syst. 4694, 777–785 (2007).

66. Bouzeghoub, N.-K.-D.: Active sharing of contextual learning experiences among users in personal learning environments using a peer-to-peer network. Proceedings of the International Conference on Advanced Learning Technologies, ICALT'10. pp. 78–82. , Sousse, Tunisia (2010).

67. Bigham, J., Lau, T., Nichols, J.: Traiblazer: enabling blind users to blaze trails through the web. Proceeding of the 14th International Conference on Intelligent user interfaces. pp. 177–186. , Sanibel Island, FL, USA (2009).

68. Karami, A.., Sehaba, K., Encelle, B.: Adaptive and Personalised Robots - Learning from Users' Feedback. IEEE 25th International Conference on Tools with Artificial Intelligence. pp. 626–632. , Herndon, VA, USA (2013).

69. Avouris, N., Komis, V., Fiotakis, G., Margaritis, M., Voyiatzaki, E.: G.: Logging of fingertip actions is not enough for analysis of learning activities. In: Proc. Workshop Usage Analysis in learning systems (AIED 2005. , Amsterdam (2005).

70. Nicaud, J.-F.: Thèse de doctorat "{APLUSIX} : un système expert de résolution pédagogique d'exercices d'algèbre," http://opac.inria.fr/record=b1058618, (1987).

71. Chaachoua, H., Croset, M.-C., Bouhineau, D., Bittar, M., Nicaud, J.-F.: Description et exploitations des traces du logiciel d'alg{è}bre Aplusix. Rev. des Sci. Technol. l'Information la Commun. pour l'Education la Form. Vol 14, 26 (2007).

72. Champin, P., Prié, Y., Mille, A.: Musette: Modeling USEs and Tasks for Tracing Experience.

73. Arana, J., Hassas, S., Prié, Y.: MAZETTE: Multi Agent MUSETTE for Sharing and Reusing Ontologies. In: Meersman, R., Tari, Z., and Corsaro, A. (eds.) On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops SE - 85. pp. 741–752. Springer Berlin Heidelberg (2004).

74. Gagnière, L., Ollagnier-Beldame, M.: Projet Clever@ Réutilisation de traces et processus métacognitifs. , http://liris.cnrs.fr/publis/?id=4954, (2004).

75. Dyke, G., Lund, K., Girardot, J.-J.: Tatiana : un environnement d'aide {à} l'analyse de traces d'interactions humaines. Tech. Sci. Informatiques. 29, pp.1179–1205 (2010).

76. Wexelblat, A.: History-rich Tools for Social Navigation. CHI 98 Conference Summary on Human Factors in Computing Systems. pp. 359–360. ACM, New York, NY, USA (1998).

77. Maders, H.-P.: Conduire une équipe projet. World, Paris (2000).

78. Sawadogo, D.: Thèse de doctorat "Architectures Logicielles et Mécanismes pour la Gestion Consolidée de Ressources Numériques dans une Application Interactive," (2015).

79. Domingos, P., Pazzani, M.: On the optimality of the simple Bayesian classifier under zero-one loss. Mach Learn. 29, 103–130 (1997).

80. J.Hand, D., Yu, K.: Idiot's Bayes - not so stupid after all? Int. Stat. Rev. 69, (2001).

81. Tan, P.-N., Steinbach, M., Kumar, V.: Introduction to data mining. Pearson Addison, Wesley (2006).

82. Vapnik, V.: The Nature of Statistical Learning Theory, (2000).

83. Ho, H.N., Rabah, M., Estraillier, P., Nowakowski, S.: A process for Trace-Based Criteria Weighting in Multiple Criteria Decision Making. 6th International Conference on Computer Research and Development. pp. 77–84. , Hanoi, Vietnam (2014).

84. Ho, H.N., Rabah, M., Nowakowski, S., Estraillier, P.: Trace-Based Weighting Approach for Multiple Criteria Decision Making. J. Softw. 9, 2180–2187 (2014).

85. Ho, H.N., Rabah, M., Nowakowski, S., Estraillier, P.: Trace-Based Decision Making in Interactive Application: Case of Tamagotchi systems. IEEE International Conference on Control, Decision and Information Technologies. pp. 123–127. , Metz, France (2014).

86. Behzadian, M., Kazemzadeh, R.B., Albadvi, A., Aghdasi, M.: PROMETHEE: A comprehensive literature review on methodologies and applications. Eur. J. Oper. Res. 200, 198–215 (2010).

87. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2011).

88. Zenebe, A., Zhou, L., Norcio, A.F.: User preferences discovery using fuzzy models. Fuzzy Sets Syst. 161, 3044–3063 (2010).

89. Jøsang, A., Hayward, R., Pope, S.: Trust Network Analysis with Subjective Logic. Proceedings of the 29th Australasian Computer Science Conference - Volume 48. pp. 85–94. Australian Computer Society, Inc., Darlinghurst, Australia, Australia (2006).

90. Oren, N., Norman, T.J., Preece, A.: Subjective logic and arguing with evidence. Artif. Intell. 171, 838–854 (2007).

91. Jøsang, A.: Legal reasoning with subjective logic. 289–315 (2001).

92. Rowe, G., Wright, G.: The Delphi technique as a forecasting tool: issues and analysis. Int. J. Forecast. 15, 353–375 (1999).

93. Shi, S., Cao, J., Feng, L., Liang, W., Zhang, L.: Construction of a technique plan repository and evaluation system based on AHP group decision-making for emergency treatment and disposal in chemical pollution accidents. J. Hazard. Mater. 276, 200–206 (2014).

94. Truchon, M.: Borda and the maximum likelihood approach to vote aggregation. Math. Soc. Sci. 55, 96–102 (2008).

95. Lepelley, D.: On the probability of electing the Condorcet. Math. Soc. Sci. 25, 105–116 (1993).

96. Hudry, O.: Votes et paradoxes: les élections ne sont pas monotones! Mathématiques Sci. Hum. 163, 9–39.

97. Ho, H.N., Rabah, M., Nowakowski, S., Estraillier, P.: Application of Trace-Based Subjective Logic to User Preferences Modeling. 20th International Conference on Logic Programming, Artificial Intelligence and Reasoning. , Suva, Fiji (2015).

98. Ho, H.N., Rabah, M., Nowakowski, S., Estraillier, P.: Aide à la décision multicritère à base de traces avec PROMETHEE II. 16ème Conférence ROADEF. , Marseille, France (2015).

99. Ho, H.N., Rabah, M., Nowakowski, S., Estraillier, P.: Personnalisation interactive de parcours pédagogiques par une méthode de décision multicritère à base de traces. 7ème Conférence sur EIAH. pp. 429–431. , Agadir, Maroc (2015).

100. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explor. Newsl. 11, 10–18 (2009).

101. Trillaud, F.: Thèse de doctorat “Architecture pour le contrôle des interactions et le pilotage d’une application interactive multi-utilisateurs à exécution adaptative: application à un environnement de FOAD,” (2013).

102. Champagnat, R., Prigent, A., Estraillier, P.: Scenario building based on formal methods and adaptative execution. ISAGA, Atlanta (USA). 6, (2005).