CELTIC/EDAIN : une approche de modélisation et de supervision d'expériences interactives

Damien Mondou, Armelle Prigent, Arnaud Revel

CELTIC/EDAIN : une approche de modélisation et de supervision d'expériences interactives

Damien Mondou[1] Armelle Prigent[1] Arnaud Revel[1]

[1] La Rochelle Université - Laboratoire Informatique, Image, Interraction (L3i)
23 Avenue Albert Einstein, 17000 La Rochelle

prenom.nom@univ-lr.fr

Résumé

Cet article présente une nouvelle approche de modélisation et de supervision d'expérience interactive. Cette approche a été utilisée pour concevoir un jeu sérieux avec un robot Nao. Ce jeu permet aux plus jeunes, de découvrir de manière ludique l'exposition dédiée à l'ethnographie au muséum d'histoire naturelle de La Rochelle et l'exposition consacrée à l'archéologie au musée Sainte Croix de Poitiers. La phase de modélisation du jeu est divisée en deux parties. La première consiste à définir des comportements atomiques, regroupable en patterns. La seconde étape vise à définir des agents implémentant les patterns précédemment définis, en spécifiant les contenus et les comportements réels exécutés sur le procédé contrôlé, dans notre cas le robot Nao. Le premier objectif de cette approche est d'externaliser les contenus du jeu et les comportements réels du procédé (les différents postures et gestures de Nao) en base de données. Le second objectif est de pouvoir définir un jeu sérieux sans aucune contrainte concernant le procédé piloté.

Mots Clef

Modélisation formelle, réseaux d'automates, robotique, interaction homme-robot.

Abstract

This paper presents a new approach to designing and supervising an interactive experience. The approach is implemented by creating a serious game with a Nao robot. This game allows youth to discover in a fun way the exhibition dedicated to the ethnographic artifacts of La Rochelle's natural history museum and the exhibition dedicated to archaeology at the Sainte Croix Museum in Poitiers. The design phase of the game is divided into two steps. The first step defined the atomic behaviors grouped within the pattern. In the second step, the agents implementing these patterns were created ; they specified the contents and behaviors to be executed on the controlled process, in our case the Nao robot. The first objective is to externalize the contents of the game (e.g. robot speech) and the real behaviors of the process (e.g. the different postures and gestures of the Nao) in a database. The second objective is to be able to define a serious game without constraints on the process piloted.

Keywords

Formal model, automatas network, robotic, human-robot interaction.

1 Introduction

Les musées sont continuellement en recherche de nouvelles manières d'améliorer l'expérience des visiteurs, de leur permettre de découvrir les collections et d'acquérir de nouvelles connaissances. Ainsi, depuis plusieurs années, le jeu sérieux a démontré son potentiel pour créer de l'engagement et amener les utilisateurs à améliorer leur compréhension d'un sujet [15]. Les avantages de l'apprentissage par le jeu ont donc été démontrés [33, 13] et de nombreux sites culturels se sont équipés de fonctionnalités ludiques dans cet objectif. Ces enjeux sont particulièrement forts pour le jeune public qui peut montrer un désintérêt pour les musées. Les dispositifs digitaux ludiques représentent un vecteur d'engagement efficace pour ces générations.
Ainsi, on trouve aujourd'hui, dans ces lieux de culture, des systèmes basés sur des écrans mobiles [9], des approches de réalité augmentée [24], de transmedia et des solutions d'immersion virtuelle [7] pour permettre aux visiteurs de naviguer dans les contenus.
Intégrer des systèmes à base de robots est une nouvelle perspective particulièrement intéressante. En effet, les robots deviennent de plus en plus présents dans notre vie quotidienne (bien souvent dans un contexte de support à l'humain). Il peut s'agir de robots d'accueil comme au King's College de Londres où la réceptionniste est un robot nommé Inkha ou dans les supermarchés pour guider les consommateurs dans leur recherche de produits ou fournir des conseils nutritionnels, tels que LoweBot [28], Aiko Chihira [31], et Pepper [5]. Les robots ont également intégrés des oeuvres artistiques [2, 18] dans le cadre de projets transdisciplinaires entre artistes et scientifiques.
Ainsi, les robots investissent aujourd'hui les musées et les lieux de culture pour constituer une alternative aux dispo-

sitifs numériques de diffusion d'information ou de création d'engagement auprès des visiteurs. L'intérêt de cette nouvelle approche tient dans plusieurs points. Elle permet d'une part une interaction plus naturelle avec les visiteurs (il est plus aisé de visiter le musée et jouer sans avoir à être équipé d'une tablette, d'un téléphone ou d'autres équipements qui pourraient interférer avec la visite et la découverte des oeuvres). D'autre part, elle ouvre la voie à une dynamique sociale entre les visiteurs (pouvant échanger entre eux quand bien même ils sont de simples spectateurs de l'interaction). Enfin, les systèmes de vision par ordinateur, intégrés au robot, permettent de détecter la présence de visiteurs et le robot peut alors les interpeller pour les inviter au jeu. Ainsi, un simple visiteur qui n'a pas fait la démarche de jouer sur tablette ou de s'équiper d'un dispositif de réalité virtuelle peut devenir un joueur pour une partie de sa visite.

1.1 L'interaction robotique dans les lieux de culture

Un certain nombre d'expérimentations ont déjà été menées pour analyser la faisabilité et l'impact de l'introduction des robots dans les lieux de culture. Ainsi, depuis 2014, deux robots sont en charge d'accueillir les visiteurs au Musée National des Sciences et Innovations émergentes à Tokyo et peuvent également produire du contenu verbal. A l'exposition Eppur Si Muove au Musée d'Art Moderne de Luxembourg, un robot Nao sur une base de robot mobile est chargé de guider le public et de présenter les œuvres [16]. En France, dans le cadre d'un projet sur l'esthétisme artificiel, le robot Berenson a été déployé au musée du quai Branly. Ce robot se comporte comme un critique d'art ; il exprime une émotion lorsqu'il se trouve devant une oeuvre. Son opinion évoluera grâce aux réactions des visiteurs présents autour de lui [6]. De plus, certains musées, en raison de leur architecture, ne sont pas accessibles aux personnes à mobilité réduite (et ne peuvent réaliser les travaux nécessaires). Le musée du château d'Oiron en France a appréhendé ce problème et, au premier étage du musée, a présenté un robot (Norio) qui peut être contrôlé à distance par un joystick à partir du rez-de-chaussée, ce qui permet aux personnes handicapées de découvrir les œuvres qui leur sont inaccessibles [17]. Sous une autre forme, un projet actuellement mené au L3i consiste à piloter un robot à partir d'un casque de réalité virtuelle [1]. Cette plateforme, destinée actuellement à la recherche et l'aide aux victimes en milieu hostile à l'homme pourrait être adaptée à la visite d'espaces culturels pour les personnes à mobilité réduite.
Nous avons donc, au travers de notre projet "Musées 3.0" et en collaboration avec le muséum d'Histoire Naturelle de La Rochelle et le musée Sainte Croix de Poitiers, développé une expérience interactive pour les jeunes visiteurs au travers d'un jeu sérieux. Ce jeu leur permet de découvrir, d'une manière intéressante et ludique, une partie des collections, au travers d'un quiz oral proposé par un robot Nao. Le robot Nao, initialement développé par la société française Aldebaran (aujourd'hui Softbank Robotics), dispose d'un ensemble de capteurs (notamment une caméra, un sonar et des capteurs tactiles) qui lui permettent de percevoir l'environnement dans lequel il évolue. Ce robot est également capable d'interagir verbalement avec l'humain au travers de ses microphones et haut-parleurs (le robot intégrant les API de reconnaissance et de synthèse vocale). Le jeu, quant à lui, consiste en un jeu de piste dirigé par le robot qui pose des questions aux jeunes joueurs. Ces derniers partent alors dans le musée à la recherche de la réponse et retournent vers Nao pour répondre lorsqu'ils l'ont trouvée.

1. Tests réalisés sur un robot Pepper et un casque HTC Vive.

1.2 Une approche générique de modélisation et de supervision

Concevoir une scénarisation de qualité pour une expérience interactive peut soulever un certain nombre de défis. D'une part, il est parfois complexe de produire une arborescence de cas suffisamment riche pour offrir à l'utilisateur une sensation de liberté dans ses choix d'interaction. De la même manière, une trop grande liberté de l'utilisateur dans une arborescence importante réduit la capacité du concepteur à analyser la qualité de chacune des exécutions possibles. Que ce soit dans l'univers des jeux [11], des systèmes de communication [20] ou de pilotage de robots [25], la nécessité d'utiliser une modélisation de l'activité est primordiale. Cette dernière permet de disposer des méthodes nécessaires à l'anticipation des chemins possibles de l'utilisateur, à la composition dynamique de comportement favorisant une arborescence plus large et à l'analyse des traces d'interaction à posteriori. Ainsi, l'utilisation de réseaux de Petri par exemple [3, 10, 36] ou d'automates à états finis [29, 35] est répandue dans l'état de l'art. En effet, ces approches permettent de garantir une haute qualité d'expérience par la vérification de certaines propriétés (de sécurité, d'accessibilité ou de vivacité notamment).
C'est dans cette dynamique que [27] propose CITE, un framework dédié complet destiné à faciliter la phase de modélisation d'un système interactif et garantissant une souplesse suffisante pour atteindre les objectifs de complexité et d'extensibilité. Il s'agit ici de prendre en compte toutes les dimensions de l'interaction (le temps, l'espace, l'interaction et le contenu) en utilisant un modèle formel capable de construire dynamiquement l'arborescence du scénario à partir de la description des agents, de leurs comportements et des contextes de leur exécution. Nous décrivons ici la structure de CELTIC implémentant les dimensions CIT (Contenu, Interaction et Temps) du modèle CITE basée sur deux couches de description (la dimension E, correspondant à la prise en compte des lieux de l'interaction n'est pas présentée ici). Le processus de supervision dynamique consiste à contrôler l'expérience interactive au regard du modèle formel (basé sur des automates à états finis). L'architecture complète de pilotage (appelée EDAIN) a été définie et implémentée. Nous avons prouvé l'effica-

cité de ce modèle dans le cadre d'expériences en présence de public.
L'un des avantages de ce modèle est qu'il permet de produire, après exécution, une trace complète de l'expérience de chaque utilisateur, modélisée sous la forme d'un automate temporisé. Nous montrerons ici, l'intérêt de cette information pour l'analyse des comportements et pour le potentiel qu'elle peut représenter pour l'apprentissage dynamique en vue de l'amélioration du modèle. En effet, elle constitue une première étape d'intégration d'un processus visant à affiner la valeur des paramètres du modèle en fonction des exécutions passées via un bouclage de pertinence.

2 CELTIC : éditeur de modélisations d'expériences interactives

CELTIC (Common Editor for Location Time Interaction and Content) est un éditeur générique de production de modèle pour des expériences interactives. Il peut s'agir d'un jeu, d'une application ou comme dans notre cas d'étude, d'une expérience de jeu avec un robot. L'objectif est de proposer une modélisation générique et modulaire qui permette de mettre en oeuvre une supervision et une analyse des traces des utilisateurs. Ce modèle vise de plus à prendre en compte les différentes composantes de l'interaction : le temps, l'espace et le contenu. La littérature propose différentes approches de modélisations permettant une supervision et une adaptation que ce soit dans l'univers du web [12, 34] ou de la scénarisation de jeu vidéo [21, 22, 23]. Ces travaux reposent pour la plupart sur des modélisations formelles garantissant une plus grande souplesse dans le contrôle de l'exécution. Ainsi, comme cité précédemment, les réseaux de Petri sont souvent utilisés pour modéliser et vérifier les activités asynchrones, en particulier les jeux sérieux [1, 26]. Des approches de logiques linéaires ont été mises en œuvre pour représenter la consommation de ressources dans le contexte du jeu [8], ou des techniques à base d'automates ont été utilisées pour garantir à l'exécution d'atteindre un état souhaité par analyse d'accessibilité [30]. Cependant, ces modèles se révèlent souvent complexes à manipuler car ils restent bas-niveaux et monolithiques (ils nécessitent, en effet, de modéliser l'intégralité de l'arborescence d'interaction). Une solution est alors de permettre une modélisation offrant une construction dynamique du modèle. Une première approche de supervision dynamique basée sur un modèle générique a été proposée dans [29], et a donné naissance à un cadre de supervision appelé *#Telling*. La modélisation, ici divisée en trois couches, peut superviser l'activité en fonction des scènes d'exécution. Les comportements atomiques des entités sont modélisés par des automates à états finis ensuite composés dynamiquement pour représenter le comportement global d'une entité dans une scène. Dans cette approche, l'objectif est de laisser une possibilité importante de choix à l'utilisateur tout en garantissant la qualité du cadre narratif et ce en facilitant la tâche du concepteur. Bien que l'efficacité de ce modèle ait été prouvée, elle est limitée par certains points. Elle ne permet pas de prendre en compte la dimension temporelle, ni dans la définition du comportement, ni dans le passage d'une situation à une autre. Enfin, la gestion de contenu n'est pas prise en charge et reste intégrée au modèle, ce qui complexifie lourdement la tâche de conception. Nous présentons ici une extension de cette dernière approche qui facilite le contrôle et permet l'externalisation du contenu et la représentation des contraintes temporelles.

2.1 Principe du modèle à deux couches

Notre approche est, elle aussi, basée sur des réseaux d'automates temporisés à états finis , très adaptés pour représenter des systèmes synchrones avec des contraintes de temps. Nous simplifions la tâche de modélisation par une méthode en deux étapes. La modélisation d'une expérience interactive est divisée en deux parties :

— tout d'abord, la description de l'expérience est réalisée de manière abstraite, en définissant les entités réutilisables (les comportements et les patterns), modélisées par des automates temporisés. Cette étape crée la *couche déclarative* ;
— par l'instanciation des entités génériques dans des agents, regroupés dans des contextes d'exécution ordonnés, la *couche d'implémentation* est alors définie.

Le processus de création d'un système interactif est illustré dans les figures 1 et 2 et détaillé dans les sections 2.2 et 2.3.

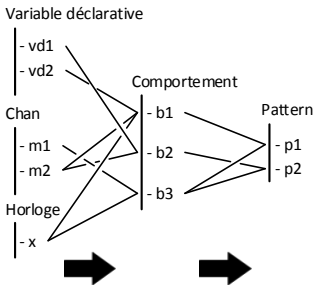

FIGURE 1 – Couche déclarative du modèle.

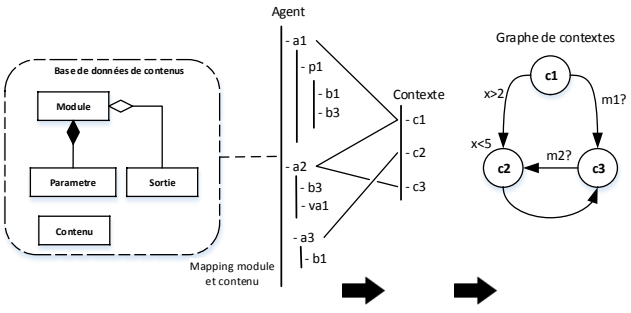

FIGURE 2 – Couche d'implémentation du modèle.

2.2 La couche déclarative

La couche déclarative est celle dans laquelle nous définissons les entités génériques du système modélisé, à savoir les comportements atomiques qui peuvent être exécutés. Ces comportements sont associés à des variables déclaratives (des entiers, des chaînes de caractères ou des booléens) et à des messages (channels), qui sont notamment les signaux permettant de synchroniser plusieurs comportements pour la composition dynamique. Les comportements sont ensuite regroupés en patterns qui représentent un ensemble de comportements, spécifiques à l'entité, qui peuvent être réutilisés par le principe d'héritage multiple.
L'enjeu majeur de cette couche déclarative est de permettre la représentation de comportements atomiques et de patterns de comportements. Ceux-ci peuvent être réutilisés et composés dynamiquement au sein d'agents dans la couche d'implémentation.

Les comportements et les patterns. Dans notre approche, un comportement atomique est une transition d'automate temporisé [19] qui viendra s'agréger avec tous les autres comportements possibles de l'agent. Par exemple, nous souhaitons représenter un premier comportement permettant au robot de se lever (dans un délai compris entre 2 et 5 unités de temps) et un second lui permettant de réaliser une reconnaissance vocale. Les modèles de ces comportements sont représentés sur la figure 3.

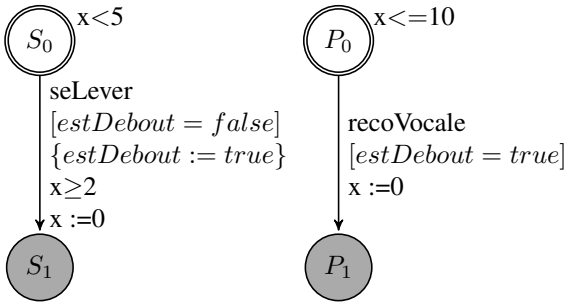

FIGURE 3 – Automates temporisés des comportements *seLever* et *recoVocale*

Les patterns décrivent un ensemble de comportements susceptibles d'être exécutés conjointement au sein de l'application et sont implémentés par des agents. Un automate temporisé du pattern est alors obtenu par le produit synchronisé de l'ensemble des automates des comportements qui le composent (selon l'algorithme de synchronisation décrit dans [4]). Cette composition permet au système d'évoluer par la synchronisation de deux entités ou par leur évolution individuelle. Un exemple d'automate obtenu à partir des comportements définis précédemment est donné à la figure 4.

L'éditeur. La conception du modèle dans cette couche déclarative repose sur un éditeur actuellement en cours de développement dans sa seconde version basée sur des technologie web afin de permettre une édition accessible à tous

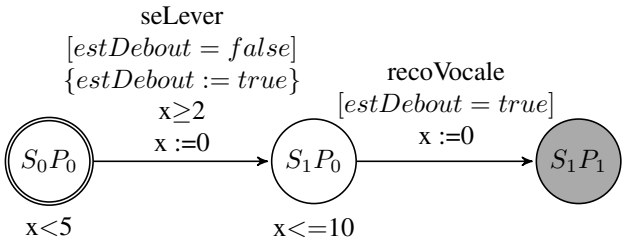

FIGURE 4 – Automate temporisé du pattern

sur un serveur distant. Cet éditeur produit un fichier XML de scénario. Ainsi, pour les comportements et le pattern décrits précédemment, l'éditeur produit la couche déclarative présentée à la figure 5.

```
1  <declarative_layer>
2    <clock id="X"/>
3    <declarative_variable id="estDebout" type="bool"/>
4    <behavior id="seLever" .../>
5    <behavior id="recoVocale" .../>
6    <behavior id="bouger" .../>
7    <behavior id="parler" .../>
8    <pattern id="recoUp" ...>
9      <behavior id="seLever" priority="1"/>
10     <behavior id="recoVocale" priority="2"/>
11   </pattern>
12 </declarative_layer>
```

FIGURE 5 – Déclaration d'un pattern

2.3 La couche d'implémentation

La couche d'implémentation utilise les comportements et patterns décrits en amont pour concevoir les entités impliquées dans l'expérience interactive : les agents et les contextes d'exécution. Il s'agit, dans un premier temps, de construire des agents qui implémentent des comportements et des patterns. Ces agents sont ensuite liés dans des contextes d'exécution représentant une situation spécifique dans lequel les agents interagissent entre eux.

Les agents. Un agent peut implémenter certains des comportements atomiques décrits dans la couche déclarative. Il peut aussi implémenter le rôle d'un ou plusieurs patterns grâce au mécanisme d'héritage multiple.
Chaque comportement composant l'agent doit correspondre à une spécification de la part du concepteur. Ainsi, grâce aux variables d'agents, le concepteur va pouvoir appliquer un ensemble de contraintes, conditionnant l'exécution du comportement. La dernière tâche a effectuer est l'association d'un module à chaque comportement. La base de données possède une table *Module* contenant l'ensemble des modules exécutables sur le processus piloté. Par exemple, pour le comportement *recoVocale* (ligne 9 de la figure 6), le concepteur va associer le module correspondant (dans notre cas le module 1). Ce module, pour être exécuté, impose la spécification de plusieurs paramètres (présent dans la table *Parametre* de la base de données) :

— *dictionnary* (ligne 10). Ce dictionnaire correspond

au vocabulaire proposé à Nao pour la reconnaissance vocale ;

— *time* (ligne 11). Une durée d'écoute pendant laquelle le joueur pourra répondre à la question posée (durée pendant laquelle le robot sera à l'écoute).

Chaque module renvoie à la fin de son exécution une liste de valeurs de sorties (présent dans la table *Sortie*). Le concepteur doit alors indiquer dans quelle variable du modèle chaque valeur retournée sera sauvegardée. Dans notre exemple, le module associé au comportement *recoVocale* retourne le mot reconnu (*wordRecognized*) et un taux de confiance (*confidence*). Ici, le concepteur a donc déclaré deux variables d'agent (ligne 3 et 4) dans lesquels vont être sauvegardés les données reçues à la fin de l'exécution, grâce à l'instruction *saveIn*. Enfin, le concepteur peut indiquer quel contenu est utilisé lors de l'exécution d'un module (en utilisant la table du même nom). Par exemple, lors de l'utilisation du module de synthèse vocale, le concepteur peut indiquer l'identifiant du discours à prononcer par le robot (ce discours étant présent dans la table des contenus).

Le concepteur peut spécialiser un agent grâce aux différents comportements qu'il implémente. L'automate de comportement de l'agent sera alors obtenu par le produit synchrone de l'ensemble des comportements. L'enjeu majeur de cette couche est de garantir la réutilisabilité des comportements. Un agent spécifique à une expérience interactive donnée peut utiliser des comportements atomiques conçus pour une autre expérience. C'est là l'un des enjeux majeurs de notre modèle.

Les figures 6 et 7 donnent une représentation XML de la déclaration d'un agent dans la couche d'implémentation et de son automate de comportements.

Les contextes et le graphe de contexte. Le contexte décrit une situation dans laquelle un certain nombre d'agents va être impliqué. Sa description se fait au travers de la liste des agents impliqués (figure 8). L'automate de contexte sera alors construit par synchronisation de l'ensemble des automates des agents présents.

Le scénario global d'exécution est produit par le graphe de contexte. Il s'agit d'un automate temporisé de haut niveau qui représente les passages entre contextes d'exécution (un exemple est donné à la figure 9).

Cette couche d'implémentation agrège d'abord les comportements dans les agents, puis les agents dans les contextes et produit dynamiquement les automates de comportement de ces derniers par synchronisation.

3 EDAIN : l'architecture de supervision

L'expérience interactive ayant été modélisée, notre outil de supervision (EDAIN) assure la gestion de l'expérience d'une part au travers d'une exécution pilotée par le parcours du graphe de contexte (et suivi des comportements générés dynamiquement au sein des contextes via la syn-

```
1  <global_variable id="out" type="string" value=""/>
2  <agent id="agentAccueil">
3    <agent_variable id="wordRecognized" type="string"
         value=""/>
4    <agent_variable id="confidence" type="double" value
         ="0.0"/>
5    <pattern id="recoUp">
6      <behavior id="seLever" rename="" module="18">
7        <moduleParameter id="posture" table="Behavior"
             id_bdd="20" saveIn=""/>
8      </behavior>
9      <behavior id="recoVocale" rename="" module="1">
10       <moduleParameter id="dictionnary" table="
             ContentDependance" id_bdd="questionID"
             saveIn="reponseCorrecte"/>
11       <moduleParameter id="time" value="10"/>
12       <output id="wordRecognized" saveIn="
             wordRecognized"/>
13       <output id="confidence" saveIn="confidence"/>
14     </behavior>
15   </pattern>
16   <behavior id="bouger" rename="" module="3">
17     <moduleParameter id="gesture" table="Behavior"
           id_bdd="9"/>
18     <output id="out" saveIn="out"/>
19   </behavior>
20   <behavior id="parler" rename="accueilPublic" module
         ="2">
21     <moduleParameter id="speed" value="80"/>
22     <moduleParameter id="text" table="Content" id_bdd
         ="7"/>
23     <output id="out" saveIn="out"/>
24   </behavior>
25 </agent>
```

FIGURE 6 – Déclaration de l'agent

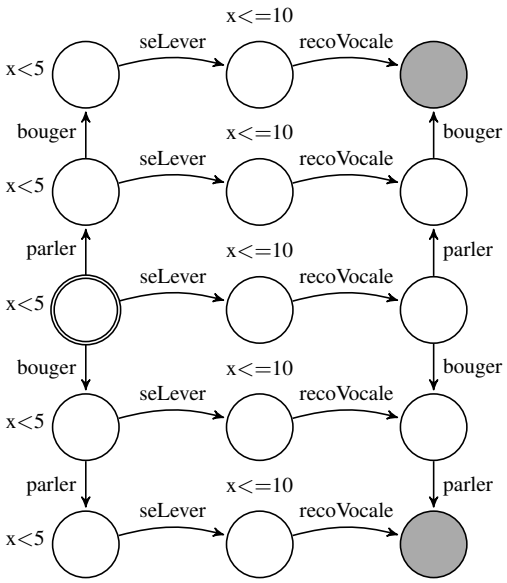

FIGURE 7 – Automate temporisé (simplifié) de l'agent

```
1  <context id="Waiting">
2    <agent id="agentAccueil"/>
3    <agent id="personneDetection"/>
4  </context>
```

FIGURE 8 – Déclaration d'un contexte dans la couche d'implémentation.

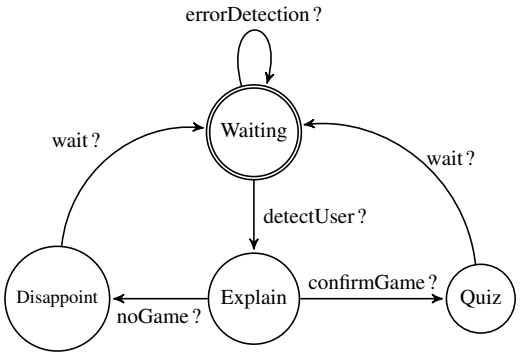

FIGURE 9 – Graphe de contexte

chronisation des agents) et d'autre part via des appels distants des comportements sur le procédé supervisé (dans notre cas : le robot Nao).

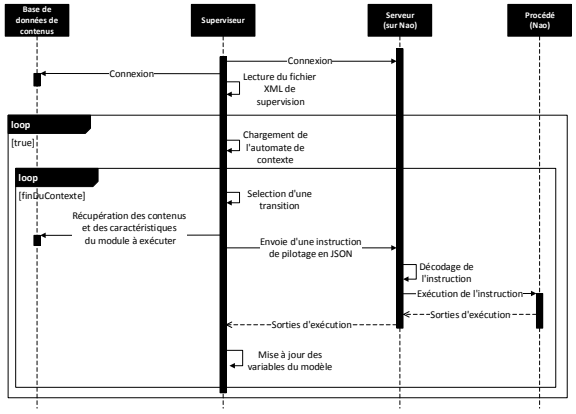

FIGURE 10 – Diagramme de séquences de la supervision.

L'objectif de notre plateforme est de dissocier la modélisation et la supervision de l'activité. Il était alors possible d'utiliser le même modèle pour contrôler différents procédés, à condition de développer un client spécifique pour chaque processus. Un diagramme de séquences de la supervision est détaillé dans la figure 10.
Cet outil de supervision utilise le fichier de spécification précédemment généré en entrée et charge le graphe de contexte. La transition d'un contexte à un autre est réalisée au travers de la réception des signaux ou via des passages de transitions spontanés par satisfaction des gardes (c'est ici la sémantique dynamique de l'automate temporisé du graphe de contexte qui détermine la réalisation de l'exécution).

4 Expérimentations publiques et analyses des situations types

Les expérimentations menées dans les différents lieux de culture ont plusieurs objectifs. Tout d'abord, elles nous permettent d'éprouver la méthode de modélisation proposée et de vérifier la généricité du modèle de scénario, la réutilisabilité des éléments de la couche déclarative d'une expérimentation à une autre et la facilité avec laquelle les contenus sont décorrélés du modèle (au travers de l'architecture s'appuyant sur la base de données présentée à la figure 2). Une autre validation porte sur l'exactitude de notre algorithme de composition dynamique des comportements pour construire les agents et les comportements du système au sein des contextes et sur la qualité du processus de supervision permettant d'exécuter le graphe de contexte. Enfin, nous souhaitons pouvoir récolter des données sur les exécutions de tous les utilisateurs afin d'exploiter la structure du modèle et mettre en oeuvre des méthodes de fouille de données pour extraire automatiquement des informations sur les comportements des usagers.
Nous avons réalisé deux expérimentations publiques pour des évènements tels que la fête de la science (Muséum d'Histoire Naturelle de La Rochelle) ou les Off de la Gamer Assembly (Poiters).

FIGURE 11 – Expérimentation au Muséum d'Histoire Naturelle - 2017.

La première expérimentation a eu lieu lors de la fête de la science 2017 (figure 11). Le scénario créé pour l'occasion avait pour objectif de faire découvrir certaines oeuvres exposées, sous forme de quiz, proposé par les robots Nao. Un total de 46 joueurs a pu découvrir les oeuvres au travers de ce jeu. La seconde expérimentation, dédiée au musée Sainte Croix de Poitiers, s'est déroulée lors de la Gamers Assembly 2018. Nous proposions à cette occasion, un nouveau scénario interactif, faisant intervenir les robots Nao et des jeux développés sur tablettes pour l'occasion. Au travers de deux séances, nous avons reçu un nombre total de 11 joueurs.

4.1 Hypothèse de réutilisabilité

Les comportements et patterns de comportements définis pour le robot Nao concernent un certain nombre d'actions liées aux capteurs et effecteurs de cette plateforme. Il peut s'agir de détecter une présence humaine, de parler, d'écouter une réponse de l'utilisateur par exemple. Les comportements décrits dans la couche déclarative pour la première expérimentation ont été réutilisés sans difficulté pour la seconde expérimentation (environ 70% de réutilisation). Nous avons donc pu redéfinir de manière relati-

vement simple un nouveau scénario sur la base des éléments existants. De plus, d'un lieu de culture à un autre, les contenus manipulés dans le scénario changent (puisqu'ils s'appuient sur les collections des musées). La structure de stockage des textes dans la base nous permet aujourd'hui d'utiliser un comportement générique (par exemple parler) et de le connecter rapidement à un nouveau contenu intégré dans la base de données. Cette nouvelle architecture nous préserve des difficultés de construction des scénarios d'exécution que pouvaient poser l'outil propriétaire de Nao (Choregraphe), qui présentait une dépendance au contenu très forte et posait de grandes problématiques pour la réutilisabilité des scénarios.

4.2 Algorithme de supervision

Au cours de la première expérimentation publique, nous avons validé l'algorithme de supervision dans sa version non-temporisée (les comportements et le graphe de contexte ne contenaient pas d'horloges). L'algorithme s'est avéré efficace et a permis de réaliser un pilotage de l'activité cohérent avec la modélisation réalisée. L'extension mise au point au cours de la seconde expérimentation a donc consisté à intégrer ces contraintes temporelles dans le modèle et à en tenir compte pour la supervision.

4.3 Analyse des comportements utilisateurs

Nous avons intégré, dans notre superviseur, un système d'observation de l'expérience pour générer des traces d'exécution. Dans l'exemple de la figure 12, les lignes 1, 3 et 5 correspondent à l'exécution d'un comportements sur le robot avec une liste de paramètres propre à chaque module (à noter que les paramètres *text* et *gesture* de la ligne 1 ont des valeurs entières, correspondants aux identifiants de la base de données). Les lignes 2, 4 et 6 correspondent quant à elles au retour d'exécution des modules sur le robot. Par exemple, la ligne 2 nous apprend que le joueur 3 a répondu "oui" à la question posée avec un taux de reconnaissance de 52 %.
L'objectif à terme est d'utiliser ces traces d'exécution pour analyser le comportement de l'utilisateur final. En effet, lors de la conception de l'expérience interactive, il est difficile pour le concepteur de prévoir tous les cas d'utilisation possibles en situation réelle. Nous souhaitons donc, grâce à ces traces, utiliser un système de clustering visant à effectuer une analyse qualitative de l'expérience a posteriori. Nous utilisons pour cela les traces issues de la première expérimentation (muséum). En effet, lors de celle-ci, des cas d'erreurs sont apparus à plusieurs reprises :

- des erreurs de reconnaissances vocales. Ces erreurs étaient en particulier dues à la résonance de la salle et du nombre importants de personnes qui s'y trouvaient ;
- des abandons. Une longue file d'attente s'est créée au fil de l'après midi qui a découragé un certain nombre de joueurs ;
- des erreurs de reconnaissance faciale liées à l'éclairage ou au mauvais positionnement du joueur.

```
1 13 oct. 2017 14:26:52; joueur:3 question; parameters: [
    speed: 100, text: 8, dictionnary: oui,non, time:
    10, gesture: 9]
2 13 oct. 2017 14:27:06; joueur:3 questionBack;
    parameters: [confidence:0.521499991417,
    wordRecognized:oui]
3 13 oct. 2017 14:27:07; joueur:3 reponseOut1;
    parameters: [gesture: 9, speed: 100, text: 9]
4 13 oct. 2017 14:27:12; joueur:3 reponseOut1Back;
    parameters: [out: ok]
5 13 oct. 2017 14:27:13; joueur:3 poserQuestion;
    parameters: [speed: 100, text: 1, gesture: 9]
6 13 oct. 2017 14:27:23; joueur:3 poserquestionBack;
    parameters: [out: ok]
```

FIGURE 12 – Exemples d'observations obtenus lors de l'expérimentation.

L'objectif est d'utiliser un algorithme pour identifier automatiquement des clusters. Ceux-ci seront ensuite utilisés afin d'identifier dans quelle situation un joueur se trouve lors d'une prochaine expérimentation afin d'intégrer le bouclage de pertinence et améliorer le modèle (notamment le seuil de confiance pour la reconnaissance).
Nous disposons de 4000 traces représentant les parties de 46 joueurs. A partir de ces traces, nous recréons le graphe modélisant l'évolution d'un joueur dans notre jeu. Nous identifions ainsi quatre clusters émergeant, représentant les parties terminées, les abandons, les erreurs de reconnaissance vocale et les erreurs d'identification. Leur répartition est donnée dans le tableau suivant, sachant qu'un graphe peut appartenir au plus à deux clusters (par exemple, la partie a pu être terminée mais des problèmes de reconnaissance vocale sont apparus).

Parties terminées	Problèmes de reco. vocale	Problèmes d'identification	Abandons
39%	28%	7%	54%

TABLE 1 – Vérité terrain de la première expérimentation.

Nous utilisons un algorithme d'apprentissage non supervisé pour effectuer la clusterisation qui doit pouvoir identifier automatiquement le nombre optimal de clusters. Toutes ces contraintes nous ont amenées à utiliser l'algorithme de clusterisation Affinity propagation [14]. Nous déterminons trois vecteurs caractéristiques différents pour effectuer la clusterisation :

- vecteur "states" : vecteur identifiant pour chaque graphe, le nombre d'occurrences de chaque états du graphe ;
- vecteur "transitions" : vecteur identifiant pour chaque graphe, le nombre d'occurrences de chaque transitions du graphe ;
- vecteur "states + transitions" : couplage des deux vecteurs précédents.

Pour chaque vecteur de caractéristiques, nous ajoutons une donnée concernant la durée finale de l'expérience.
L'utilisation des deux premiers vecteurs par l'algorithme

Affinity propagation permet d'identifier 5 clusters. Cependant, aucune sémantique particulière n'en ressort. En effet, la donnée sur le temps de la partie a beaucoup influencé l'algorithme. Nous avons donc utilisé le troisième vecteur qui permet quand à lui d'identifier 6 clusters, que nous avons étiqueté a posteriori de la façon suivante :

- Abandon sans erreur particulière ;
- Partie terminée avec erreur de reconnaissance vocale ;
- Abandon avec problème d'identification des joueurs (le joueur était mal placé devant le robot) ;
- Abandon avec erreur de reconnaissance vocale ;
- Erreur d'identification ;
- Partie terminée sans erreur particulière.

Sur la base de cette clusterisation, la prochaine étape consistera à identifier dynamiquement dans quelle situation le joueur se trouve afin de parer aux problèmes éventuellement rencontrés, de façon dynamique.

5 Conclusion

Nous avons présenté ici notre modèle de conception d'expériences interactives et la mécanique de supervision permettant de l'utiliser pour piloter le système. Des expériences publiques ont été menées. Elles nous ont permis de valider notre approche et d'entamer des travaux sur l'analyse des comportements utilisateurs via la détection de clusters dans les traces.

De plus, ces expérimentations nous ont prouvé à quelle point la prise en compte du temps est importante pour obtenir une interaction de qualité. En effet, plus encore que dans les systèmes informatiques classiques, le paramétrage des temps de parole du robot et des délais au cours desquels il écoute une réponse verbalisée par le public est primordiale. Par exemple, nous avons constaté que la rapidité de réponse d'un utilisateur est la plupart du temps dépendante de son âge. Ainsi, un enfant va répondre très rapidement (parfois même avant que la séquence d'enregistrement du robot soit déclenchée), alors qu'une personne plus âgée prend davantage de temps avant de verbaliser sa réponse (et souvent alors que cette même séquence d'enregistrement est terminée). De plus, le choix des délais d'attente du robot avant de réagir à la présence d'un visiteur, sa vitesse de parole ou de mouvement doivent être paramétrés avec précaution car ces éléments ont un impact fort sur la perception du public [32]. C'est dans cette dynamique que nous avons intégré au framework une prise en compte des contraintes temporelles pour les comportements des agents et pour la gestion des passages entre contextes. Au delà de cela, nous envisageons d'utiliser les traces utilisateurs pour déterminer par apprentissage automatique les délais d'interaction appropriés aux types de visiteurs et mettre en place un raffinement dynamique du modèle via un bouclage de pertinence. C'est ce qui constitue aujourd'hui la perspective principale de nos travaux. Enfin, nos premiers tests concernant le pilotage d'un robot Pepper via la plateforme CELTIC/EDAIN ont été concluant. En effet, une partie du jeu a pu être déployé sur le robot Pepper sans aucune modification du modèle.

Références

[1] Manuel Araújo and Licínio Roque. Modeling games with petri nets. *Breaking New Ground : Innovation in Games, Play, Practice and Theory - Proceedings of DiGRA 2009*, 01 2009.

[2] Elise Aspord, Joffrey Becker, and Emmanuelle Grangier. *Link human/robot*. Van Dieren eds, 2014.

[3] Franciny M. Barreto, Joslaine Cristina Jeske de Freitas, and Stéphane Julia. A timed petri net model to specify scenarios of video games. In Shahram Latifi, editor, *Information Technology - New Generations*, pages 467–473, Cham, 2018. Springer International Publishing.

[4] Johan Bengtsson and Wang Yi. Timed automata : Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*. Springer Berlin Heidelberg, 2004.

[5] Daniel Van Boom. Pepper the humanoid robot debuts in france. publié sur le site Cnet le 21/10/2015.

[6] Sofiane Boucenna, Philippe Gaussier, Pierre Andry, and Laurence Hafemeister. A robot learns the facial expressions recognition and face/non-face discrimination through an imitation game. *International Journal of Social Robotics*, 6(4) :633–652, Nov 2014.

[7] Marcello Carrozzino and Massimo Bergamasco. Beyond virtual museums : Experiencing immersive virtual reality in real museums. *Journal of Cultural Heritage*, 11(4) :452 – 458, 2010.

[8] Ronan Champagnat, Pascal Estraillier, and Armelle Prigent. Adaptative execution of game : Unfolding a correct story. In *International Conference on Advances in Computer Entertainment Technology*, ACE 06, New York, 2006. ACM.

[9] Tanguy Coenen, Lien Mostmans, and Kris Naessens. Museus : Case study of a pervasive cultural heritage serious game. *J. Comput. Cult. Herit.*, 6(2) :8 :1–8 :19, May 2013.

[10] Hugo Costelha and Pedro Lima. Robot task plan representation by petri nets : Modelling, identification, analysis and execution. *Autonomous Robots*, 33, 11 2012.

[11] G. W. de Oliveira, S. Julia, and L. M. Soares Passos. Game modeling using workflow nets. In *2011 IEEE International Conference on Systems, Man, and Cybernetics*, pages 838–843, Oct 2011.

[12] Roberto De Virgilio. AML : A modeling language for designing adaptive web applications. *Personal and Ubiquitous Computing*, 16(5) :527–541, 2012.

[13] Diane Dietze. *Playing and learning in early childhood education*. Wadsworth Publishing Company, 2011.

[14] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, 315(5814) :972–976, 2007.

[15] Juho Hamari, David J. Shernoff, Elizabeth Rowe, Brianno Coller, Jodi Asbell-Clarke, and Teon Edwards. Challenging games help students learn : An empirical study on engagement, flow and immersion in game-based learning. *Computers in Human Behavior*, 54 :170 – 179, 2016.

[16] Patrick Henaff. Entre art et science : Guido, un robot guide espiègle au musée d'art moderne de luxembourg. *session vidéo, Journées Nationales de la Recherche en Robotique (JNRR)*, october 2015.

[17] Mathilde Khlat. Norio, the robot guide of the oiron castle. Publié sur le site de Tourmag le 11/12/2014.

[18] Matthieu Lapeyre, Pierre Rouanet, and Pierre-Yves Oudeyer. Poppy : a New Bio-Inspired Humanoid Robot Platform for Biped Locomotion and Physical Human-Robot Interaction. In *Proceedings of the 6th International Symposium on Adaptive Motion in Animals and Machines (AMAM)*, Darmstadt, Germany, March 2013.

[19] Kim G. Larsen, Paul Pettersson, and Wang Yi. Uppaal in a nutshell. *STTT*, 1 :134–152, 1997.

[20] Thi Thieu Le, Luigi Palopoli, Roberto Passerone, and Yusi Ramadian. Timed-automata based schedulability analysis for distributed firm real-time systems : A case study. *Int. J. Softw. Tools Technol. Transf.*, 15(3) :211–228, June 2013.

[21] A. Liapis, H. P. Martínez, J. Togelius, and G. N. Yannakakis. Adaptive game level creation through rank-based interactive evolution. In *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, pages 1–8, Aug 2013.

[22] Y Louchart and Ruth Aylett. Emergent narrative, requirements and high-level architecture. In *In Proceedings of the 3rd Hellenic Conference on Artificial Intelligence*, page 308, 2004.

[23] Brian Magerko. Building an interactive drama architecture. In *In First International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, pages 226–237, 2003.

[24] T. Miyashita, P. Meier, T. Tachikawa, S. Orlic, T. Eble, V. Scholz, A. Gapel, O. Gerl, S. Arnaudov, and S. Lieberknecht. An augmented reality museum guide. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '08, pages 103–106, Washington, DC, USA, 2008. IEEE Computer Society.

[25] A. Miyazawa, P. Ribeiro, W. Li, A. Cavalcanti, and J. Timmis. Automatic property checking of robotic applications. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3869–3876, Sep. 2017.

[26] Stéphane Natkin and Liliana Vega. A Petri Net Model for the Analysis of The Ordering of Actions in Computer Games. In *GAME ON 2003*, France, January 2003. London, October 2003.

[27] Armelle Prigent and Arnaud Revel. CITE – Content Interaction Time and spacE : a hybrid approach to model man-robot interaction for deployment in museums. *EAI Endorsed Transactions on Creative Technologies*, 4(13), November 2017.

[28] Prnewswire. Lowe's introduces lowebot - the next generation robot to enhance the home improvement shopping experience in the bay area. Publié sur le cite PnNewsWire le 30 août 2016, 2016.

[29] Nicolas Rempulski. *Synthèse dynamique de superviseur pour l'exécution adaptative d'applications interactives*. PhD thesis, Université de La Rochelle, 2013.

[30] Nicolas Rempulski, Armelle Prigent, Vincent Courboulay, Matthieu Perreira Da Silva, and Pascal Estraillier. Adaptive Storytelling Based On Model-Checking Approaches. *International Journal of Intelligent Games & Simulation (IJIGS)*, 5(2) :33–42, November 2009.

[31] Reuters. Humanoid robot starts work at japanese department store. Publié sur le site de Reuters le 20 avril 2015.

[32] Arnaud Revel and Pierre Andry. Emergence of structured interactions : from a theoretical model to pragmatic robotics. *Neural Networks*, 22(2) :116–125, January 2009.

[33] Ingrid Pramling. Samuelsson and Marilyn. Fleer. *Play and learning in early childhood settings : international perspectives*. Springer Dordrecht ; London, 2008.

[34] Chian Wang, Dao Zhi Wang, and Jia Li Lin. ADAM : An adaptive multimedia content description mechanism and its application in web-based learning. *Expert Systems with Applications*, 37(12) :8639–8649, 2010.

[35] Rui Wang, Yong Guan, Houbing Song, Xinxin Li, Xiaojuan Li, Zhiping Shi, and Xiaoyu Song. A formal model-based design method for robotic systems. *IEEE Systems Journal*, PP :1–12, 09 2018.

[36] V. A. Ziparo, L. Iocchi, D. Nardi, P. F. Palamara, and H. Costelha. Petri net plans : A formal model for representation and execution of multi-robot plans. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '08, pages 79–86, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.