Vers une utilisation éco responsable des objets connectés par la mutualisation de leurs composants physiques : Une approche basée sur le concept d'artefact

Richard Fontaine, Rémy Courdier, Denis Payet

Richard Fontaine, Rémy Courdier, Denis Payet

Vers une utilisation éco responsable des objets connectés par la mutualisation de leurs composants physiques : Une approche basée sur le concept d'artefact

Volume 3, nº 3-4 (2022), p. 393-413.

<http://roia.centre-mersenne.org/item?id=ROIA_2022__3_3-4_393_0>

Revue Ouverte d'Intelligence Artificielle
Volume 3, n° 3-4, 2022, 393-413

Vers une utilisation éco responsable des objets connectés par la mutualisation de leurs composants physiques : Une approche basée sur le concept d'artefact

Richard Fontaine[a], Rémy Courdier[a], Denis Payet[a]

[a] Université de La Réunion, rue Josehp Wetzell – Parc Technologique Universitaire, Laboratoire d'Informatique et de Mathématiques (LIM), 97490, Sainte-Clotilde, La Réunion, France
E-mail : richard.fontaine@univ-reunion.fr, remy.courdier@univ-reunion.fr, denis.payet@univ-reunion.fr.

Résumé. — À l'échelle planétaire, l'empreinte environnementale du numérique représente un continent de 2 à 3 fois la taille de la France et 5 fois le poids du parc automobile français. Une cause majeure de cette catastrophe écologique serait la surabondance d'objets connectés, liée notamment à un usage irraisonné, amplifié par une logique de surconsommation. Afin de limiter l'impact négatif de cette surabondance et d'optimiser le potentiel informatique émanant de notre environnement connecté, nous proposons dans cet article un modèle d'architecture, basé sur le concept d'artefact, visant à favoriser la mutualisation des composants présents au sein de nos appareils connectés. Pour montrer sa faisabilité et les avantages liés à son utilisation, nous proposons une implémentation de ce modèle pour la plateforme Android.

Mots-clés. — Green IT, Informatique Ubiquitaire, Mutualisation, Meta-Model Agent & Artefact, Système multi-agents en temps réel.

1. Introduction

Nous sommes au début d'une nouvelle ère de l'informatique qui se manifeste par une tendance à l'intégrer dans les objets physiques de la vie quotidienne [44, 3, 36]. Bien qu'apportant de nombreux avantages, cette avancée technologique apporte un ensemble de problématiques environnementales [5, 7].

Selon l'Union européenne [12], en 2020, il existait 30 matières premières critiques (contre 14 en 2011) et parmi elles, de nombreux éléments sont directement liés aux

nouvelles technologies : le Cobalt (batteries lithium-ion) ; le Germanium (fibres optiques) ; le Hafnium (processeur) ; l'Indium (écran tactile) ; le Tantale (écrans à cristaux liquides, les puces de mémoire vive dynamique (DRAM)). De plus, l'année dernière la Bauxite (principal minerai permettant la production d'aluminium.), le Lithium (batterie) et le Titane (Industrie) ont été ajoutés à cette liste pour la première fois.

Ces éléments étaient en général liés à la fabrication des ordinateurs, des imprimantes et d'autres objets numériques usuels. Mais depuis 2015, on remarque qu'un basculement s'est opéré : [5] :

- Les télévisions représentaient 5 % à 15 % des impacts en 2010 contre 9 % à 26 % d'ici à 2025 ;
- Les smartphones représentaient 2 % à 6 % des impacts en 2010 contre 4 % à 16 % d'ici à 2025 ;
- Les objets connectés représentaient 1 % des impacts en 2020 contre 18 % à 23 % d'ici à 2025.

À la vue de ces chiffres et de la masse d'objets connectés fabriqués chaque année (IdO ou IoT en anglais)[6, 24], nous pouvons constater qu'un épuisement de nos réserves mondiales arrive à grands pas.

Ce constat alarmant nous révèle qu'il est nécessaire d'adopter une nouvelle vision de l'éco-conception qui ne se focalise pas uniquement sur l'impact énergétique, mais plutôt sur une sauvegarde directe de nos matières premières. Pour cela, nous pensons qu'il est nécessaire de voir dans nos objets connectés plus loin que leurs fonctionnalités initiales. Ces fonctionnalités cachées et actuellement inexploitées, que l'on va nommer dispositions, représentent le cœur de notre approche. Nous proposons ainsi, au travers d'un modèle d'architecture, de les révéler et de les mutualiser virtuellement afin de faire émerger des nouveaux services sans ajout systématique d'objets connectés. Ce modèle a été implémenté au sein d'un framework nommé Agent Framework For Omnipresent Real Device (AFFORD).

Avant de présenter les solutions théoriques et techniques qui sont proposées afin de rendre cette mutualisation des dispositions possibles, nous allons illustrer notre concept afin de mieux comprendre ses avantages.

2. Un environnement connecté riche en opportunités

Dans le cadre d'un système ambiant, la démarche actuelle consiste à ajouter de nouveaux équipements à chaque fois que l'on doit produire un nouveau service.

Cependant, notre environnement est déjà empli de divers capteurs et effecteurs. Pour nous en rendre compte, nous pouvons citer l'exemple des smartphones dont la liste de capteurs embarqués est impressionnante. Dans la Figure 2.1, nous pouvons voir une liste non exhaustive, des capteurs présents dans un Samsung Galaxy s8, commercialisé en 2017. De la même manière, comme nous pouvons le voir dans la Figure 2.2 notre environnement fourmille d'éléments capables de transmettre de l'information ou d'agir sur son environnement.

Type de capteurs	Détection
HRM	Rythme cardiaque
Microphone	Son/Volume sonore/souffle
Luxmètre	Mesure de luminosité
Iris/Empreinte	Authentification
Accéléromètre	Mouvement
Capteurs météorologiques	Humidité/température /pression atmosphérique

FIGURE 2.1 – Différents capteurs présents dans le Samsung galaxy s8

Type d'effecteur	Objets caractéristiques	Utilisations
Lumière	Téléphone, télévision, ampoule connectée	Attire l'attention sur un emplacement
Chenillard	Électroménager	Transmettre un message simple
Écran	Électroménager, téléphone, télévision	Transmettre un message visuel
Écran tactile	Tablette, smartphone	Apporte un message complet, avec possibilité d'interaction avec l'utilisateur
Haut-parleurs	Télévision, smartphone, alarme	Attirer l' attention sur un problème détecté ou transmettre un message
Vibration	Smartphone, tablette, pda	Attirer l'attention vers l'écran de l'appareil

FIGURE 2.2 – Ensemble d'éléments réutilisables présents dans l'habitat

Habituellement, nous ne voyons dans ces objets que leurs fonctionnalités initiales imposées par leurs concepteurs. Cependant, même si leurs possibilités peuvent être limitées par les composants électroniques, leurs usages quant à eux peuvent être bien plus variés. Ainsi, nous montrons que chacun de nos objets connectés possède des capacités potentielles et inexploitées à effectuer des tâches génériques que nous nommerons des dispositions. Dans le but de mettre à profit ce potentiel, nous proposons ici de les mutualiser virtuellement afin de dissocier leurs capacités des contextes où elles peuvent être exploitées.

2.1. Exemple illustratif d'une mutualisation des composants : Un agent de surveillance

Pour illustrer nos propos, intéressons-nous à l'exemple d'un agent A, chargé de surveiller et d'alerter en cas de chute. Nous nous concentrons sur l'un des objectifs de cet agent, qui consiste à lancer une alerte à un proche lorsqu'une chute a été détectée. Cet agent A possède, parmi tous ses objectifs, l'objectif Ω de prévenir un proche lorsque la personne est tombée. Cet objectif Ω peut être divisé en trois objectifs, dont l'accomplissement est nécessaire pour satisfaire Ω, qui sont :

- ω_1 : Détecter une chute de la personne
- ω_2 : Alerter le proche par un son
- ω_3 : Alerter le proche à l'aide d'une lumière clignotante

Exemple 1 – Liste des objectifs disponibles

Ainsi, les objectifs ω_2 et ω_3 sont inactifs au démarrage du système. Ils seront activés lorsque l'objectif ω_1 sera atteint, c'est-à-dire lorsqu'une chute sera effectivement détectée. En outre, cet agent A a accès à un ensemble de dispositifs disséminés dans son environnement. Dans notre exemple, nous allons nous intéresser à un sous-ensemble composé de trois objets connectés : une ampoule, une télévision, ainsi qu'un détecteur de chute. Ainsi, il est possible de décrire ces objets Obj^i de l'environnement par leurs dispositions n, qu'ils présentent :

- Obj^1 : Le détecteur de chute
  - n_1 détecter une chute de la personne assistée
- Obj^2 : La télévision de la personne proche
  - n_2 produire du son
  - n_3 afficher une image
- Obj^3 : L'ampoule connectée de la personne proche
  - n_4 faire de la lumière

Exemple 2 – Liste des objets disponibles

Dans le cas usuel, le service serait géré par une seule entité qui intégrerait l'ensemble des objectifs Ω et les limiterait à ceux-ci. Dans notre cas, il est possible de dissocier le service de surveillance, la captation d'information et la réaction du système. Lorsque le système démarre, l'agent A cherche à accomplir l'objectif Ω. À la vue de la présence du détecteur de chute Obj^1 dans son environnement et de sa disposition n_1 à détecter une chute, l'objectif ω_1 est rendu actif en permanence. Si une chute est détectée, l'agent doit réagir et ses objectifs ω_2 et ω_3 deviennent actifs. Dans le cas, où l'ampoule Obj^3 et la télévision Obj_2 sont accessibles, l'agent a la possibilité de déclencher les comportements associés aux objectifs d'alerte (ω_2 et ω_3). Si, dans un autre cas, la lampe n'est plus utilisable par l'agent, la télévision Obj_2 offre toujours la possibilité d'utiliser l'écran pour déclencher une alerte clignotante. Dans cet environnement, l'alerte sonore, qui est nécessaire à l'objectif ω_2, pourra toujours être déclenchée grâce à la télévision. De cette façon, l'attention de la personne proche est dans tous les

cas attirée sur le problème de la chute de la personne qui est assistée. De la même manière, cette même ampoule Obj^3 et cette même télévision Obj_2 peuvent servir de source d'alerte pour d'autres types de suivi tels que l'intrusion, la température, l'hydrométrie, etc. Il est à noter que nous avons ici un contexte particulier mettant en avant certains dispositifs et que l'agent reste toujours en capacité de s'adapter en fonction d'autres contextes présentant les bonnes dispositions. Par exemple, il est possible pour un agent de s'adapter à l'absence de détecteur de chute dédié en profitant des capteurs présents au sein d'un smartphone. Sans avoir à ajouter forcément de nouveaux éléments dans notre environnement, nous sommes donc en capacité de mener à bien plusieurs services simultanément qui a priori ne sont pas proposés dans la conception initiale et individuelle des objets. Cette vision de notre environnement nous permet ainsi de diminuer la redondance des appareils au sein d'un système en privilégiant une utilisation plus intelligente et raisonnée des objets qui nous entourent, tout en renforçant sa robustesse en augmentant la redondance des dispositions.

2.2. Un défi écologique, mais aussi un défi technique

Il existe de nombreux travaux liés à l'informatique ambiante appliqués à divers domaines tels que : la santé [23] ; la gestion intelligente des villes[3] ; le bureau intelligent [36].

Cependant, pour rendre la vision précédemment présentée réelle et amorcer ce changement d'habitude, quatre points doivent cependant être pris en compte :

- D'un point de vue conceptuel, nous pensons qu'il est nécessaire de proposer une solution de haut niveau, qui permettrait un déploiement rapide sur un maximum d'appareils existants.
- D'un point de vue interaction, les entités en charge de fournir les services (humains ou agents) doivent pouvoir interagir facilement avec les dispositions.
- D'un point de vue de la mise en avant des dispositions, nous voulons privilégier la présence d'entités autonomes pouvant décider individuellement des dispositions qu'elles souhaitent mutualiser.
- D'un point de vue opérationnel, compte tenu de l'aspect ubiquitaire, la solution doit être suffisamment légère pour permettre son déploiement sur des appareils à faibles capacités de calcul.

3. Travaux connexes

3.1. IoT

Les plus proches travaux connexes à notre proposition sont principalement liés aux domaines de l'informatique des Objets (IdO ou IoT en anglais) et au Web des Objets (WdO ou WoT en anglais). Tout d'abord, en ce qui concerne l'IoT, il est apparu comme une évidence de relier la nouvelle masse d'objets connectés miniaturisés au réseau Internet [6, 24]. C'est ce réseau d'appareils hétérogènes que l'on appelle l'Internet des Objets. Cependant, est demeuré le problème d'interaction avec et entre

ces objets connectés, demandant sans cesse de s'adapter aux technologies associées. Afin de lutter contre les problématiques liées à cette hétérogénéité, l'intergiciel est généralement considéré comme la couche générique qui fournit des fonctions de base permettant d'abstraire la complexité de l'environnement. [2, 18]. Cette approche est couramment utilisée [16, 29], cependant une limite à l'utilisation des intergiciels vient du fait que, généralement, un système a autant d'intergiciels qu'il y a de problèmes de communication [26, 32]. Par conséquent, plus des entités hétérogènes sont impliquées dans un système, plus celui-ci peut contenir un grand nombre d'intergiciels afin de masquer les problèmes de communication. Même si certains mettent en avant la mise en place d'un « middleware de middlewares » afin de proposer une interface unique aux applications [41], le système sera incapable de s'adapter à l'apparition de nouvelles technologies. En ce qui concerne ce problème, la tendance se dirige vers une communication avec ces objets par le biais de requêtes HTTP [25]. C'est ce constat qui a été le prémice de ce que nous appelons aujourd'hui le Web des Objets [43]. Le WoT, standardisé depuis par le W3C, vise donc à réutiliser les normes du Web pour mettre à disposition les fonctionnalités des dispositifs de l'IoT. Cette démarche permet de s'appuyer sur des normes déjà existantes (comme HTTP ou REST) pour résoudre le problème d'interopérabilité [11]. Par rapport à nos travaux, le WoT se situe sur une couche d'abstraction différente et complémentaire. Notre désir de découplage entre la conception des services et la conception des objets connectés n'entre donc pas en contradiction avec l'utilisation de protocole comme ceux cités précédemment. Malheureusement, aucun de ceux présentés ne résout directement notre problématique de mutualisation de dispositions.

3.2. Une autonomie grâce à l'agent

Dans un contexte d'intelligence ambiante, une autre option pour surmonter l'hétérogénéité de l'environnement peut être l'utilisation d'agents [13, 19, 40, 45] En effet, grâce à leurs capacités à être autonomes, à agir en temps réel, à être sociaux et proactifs les agents deviennent une alternative solide aux logiciels classiques. De plus, grâce à l'augmentation des capacités de calcul, un objet connecté peut directement être utilisé comme support à l'exécution d'un agent gérant directement les traitements associés. Cependant, si l'on veut monter en charge, il est inconcevable de faire porter le poids des descriptions de tous les objets connectés et de leurs possibilités au niveau des agents en charge des services. En se basant sur ce constat, certaines propositions tentent de dépasser les problèmes de mise à disposition des objets connectés en les liant à des « agents-objets » spécialisés [28, 30, 31, 35]. Ainsi chaque objet connecté est virtuellement représenté par un agent « interface » qui fait office de médiateur entre l'objet et les agents « standards » (Figure 3.1). Cette option se heurte malheureusement à un problème conceptuel. En effet, il n'est pas conseillé de modéliser un objet en utilisant le concept d'agent, car il ne possède ses caractéristiques [8, 9, 19, 45]. Ces difficultés rencontrées par le tout agent peut être surmonté par la notion d'environnement de travail des agents qui propose une séparation entre agents et environnement.

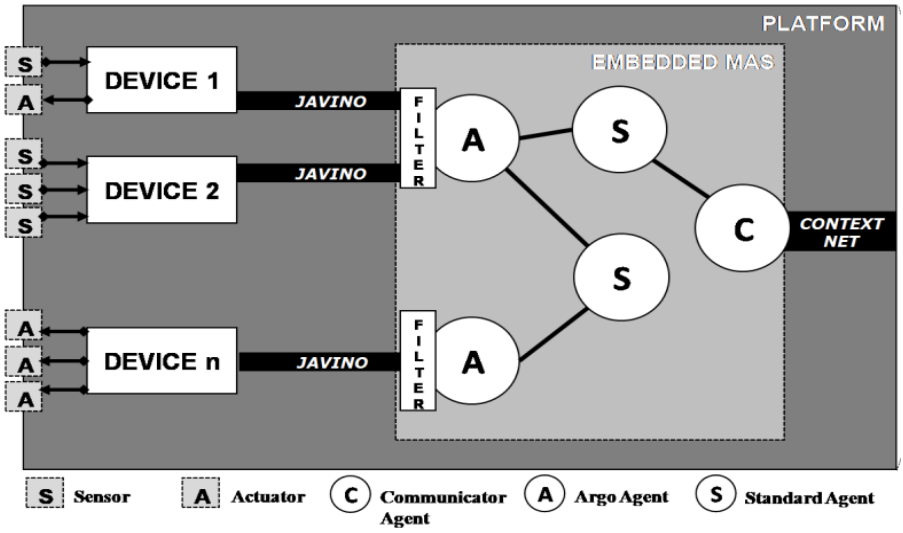

FIGURE 3.1 – Exemple d'utilisation d'agents comme médiateur entre objets et agents [35]

Dans ce cadre, un exemple proche de nos travaux est celui du méta-modèle A&A (Agent & Artifact) et sa caractérisation des interactions via trois abstractions principales [33] :

- Agents : composants proactifs de systèmes, encapsulant l'exécution d'activités dans un environnement donné ;
- Artifacts : composants passifs des systèmes, tels que les ressources et les médias, destinés à être utilisés par les agents pour soutenir leurs comportements ;
- Workspaces : conteneurs conceptuels d'agents et d'artefacts, utiles pour définir les topologies de l'environnement et les notions de localité.

On peut donner un exemple de son utilisation au travers du framework Cartago [37] au sein de JaCaMo [4] pour la programmation de systèmes multi-agents opérant dans des environnements distribués [39]. Ce concept ne résout malheureusement pas directement le problème de la mutualisation des dispositions des objets présents dans notre environnement mais nous pouvons, entre autres, remarquer sa cohérence en termes de modélisation agent/objet faisant défaut au « tout agent » ainsi que l'idée d'encapsulation de ressources au sein d'une composante virtuelle. De plus, certains des éléments de conception proposés par le modèle peuvent être simplifiés. On peut citer par exemple des notions telles que le mode d'emploi, présente dans le modèle A&A, qui devient dans les faits obsolètes à la vue des dispositions que l'on pourrait considérer comme primaires ou atomiques (e.g lumière, vibration, données capteur).

3.3. Vers une interaction plus directe : l'affordance

Il demeure un aspect lié aux méthodes d'interaction avec ces dispositions d'aspect atomiques qu'il reste à éclaircir. Au plus proches de nos problèmes d'interaction, les domaines de la psychologie et de l'intelligence artificielle peuvent nous éclairer via le concept d'affordance [17]. Celui-ci suggère que la conception des dispositifs présents dans l'environnement informe les utilisateurs sur la façon de les utiliser. Quelques exemples sont les formes d'une poignée de porte qui correspond à la forme d'une main humaine, une surface rigide et horizontale qui encourage une personne à marcher dessus ou une chaise à s'asseoir. Ce concept d'affordance a inspiré de nombreux axes de recherche qui se focalise sur la modélisation/formalisation du lien entre perceptions et actions. On peut citer, entre autres, les domaines de la simulation à base d'agents, où l'affordance est utilisée dans plusieurs domaines où l'interaction est centrale [1, 27] ou de la robotique, où l'affordance est utilisée pour détecter des actions possibles sans nécessiter une quantité importante d'entraînement étiqueté [20, 21, 22]. Au sein de cette littérature, un domaine reste cependant grandement inexploré, celui de l'intégration de l'affordance pour les agents virtuels dans un contexte ubiquitaire. L'affordance, peut toutefois y jouer un rôle clé pour modéliser l'interaction entre un agent et son environnement en permettant aux objets d'exposer directement leurs dispositions.

4. Description du concept

À la vue des contraintes et des éléments soulevés ci-dessus, nous pouvons déduire que les solutions actuelles ne répondent pas aux exigences liées à notre proposition. Afin de représenter les dispositions présentes au sein de notre environnement, nous proposons de l'encapsuler au sein d'un artefact dispositionnel. Même si le terme d'artefact est le même que celui utilisé dans le modèle A&A [33], des différences conceptuelles subsistent. Dans notre cas, nous définissons l'artefact de manière plus abstraite en tant que composant passif du système encapsulant une ressource.

Suite à ces précisions conceptuelles sur la nature de l'artefact, nous proposons de modifier notre vision de l'environnement en proposant une décomposition virtuelle de nos objets [14] sous forme d'artefacts dispositionnels que l'on va définir ainsi :

Définition 4.1 (Décomposition en artefact dispositionnel). — *Chaque objet Obj, peut être représenté par un ensemble d'artefacts dispositionnels Art selon l'usage tel que :*

$$Decomp(Obj) -> \{Art_n\}$$

avec Obj un objet connecté, Art un artefact virtuel, n la disposition de l'objet encapsulée par l'artefact et Decomp une fonction de décomposition en artefacts dispositionnels.

À partir de ces éléments, nous pouvons alors formaliser la notion de virtualisation des dispositions ou chaque objet peut être représenté virtuellement par un ensemble d'artefacts dispositionnels. Sur la Figure 4.1 nous pouvons voir un exemple de décomposition sous forme d'artefacts dispositionnels reprenant l'exemple illustratif vu précédemment (Exemple 2.1).

Algorithm 1 Création d'artefacts via une liste de disposition

```
list < artifact > ListArtifacts ←
for disposition : listDisposition do
    Artifact art ←
    if disposition is Effector then
        art ← new Effector(disposition)
    end if
    if disposition is Sensor then
        art ← new Sensor(disposition)
    end if
    if disposition is Service then
        art ← new Service(disposition)
    end if
    ListArtifacts.add(art)
end for
```

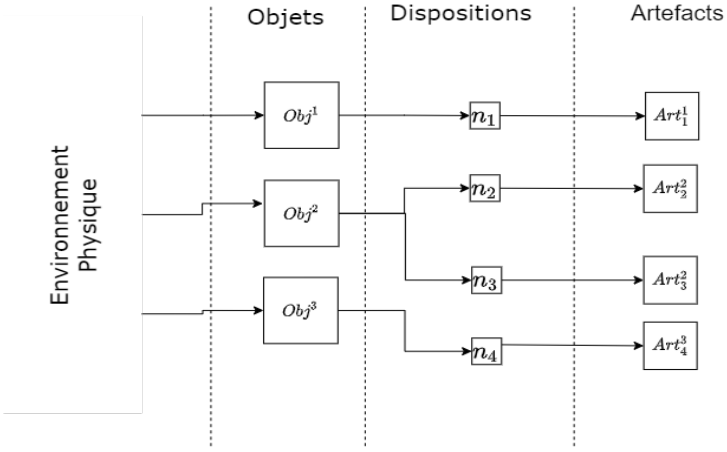

FIGURE 4.1 – Une mise en valeur des dispositions sous forme d'artefacts

Du fait qu'une disposition est une propriété intrinsèque de l'objet, l'artefact qui l'encapsule existe dans l'environnement virtuel sans pour autant être lié immédiatement à un workspace. Il reste donc à la charge du propriétaire de l'objet de l'exposer au sein d'un ou plusieurs workspaces (*Expose&Unexpose*). Un worpkspace est alors défini par l'ensemble d'artefacts $Art_0, ...Art_n$ exposé qui le compose. Une fois que le workspace est créé un agent agent A avec une capacité q (parmi ces m capacités) peut alors être liée à un objet Obj proposant une disposition p pour former un système W_{pq} = interest(Obj_p, A_q). Suite à l'encapsulation au sein de l'artefact dispositionel Art on notera finalement W_{pq} = interest(Art_p, A_q). On peut le formuler autrement comme le système « agent A (possédant la capacité p) intéressé par une disposition q (représenté par l'artefact Art) ». Cette fonction *interest* peut être décrite comme

interest	m_1	m_2
n_1	Ø	Ø
n_2	Ø	τ

FIGURE 4.2 – Fonction de juxtaposition *interest*

une jonction ou une juxtaposition, entre les capacités de l'agent et les dispositions présentes dans le workspace, qui permet à celui-ci d'actualiser W_{pq} en filtrant les $m * n$ opportunités qui lui sont possibles (Algorithm 4.2).

Algorithm 2 Analyse d'un workspace

```
list < artifact > ListArtifacts ← workspace.getArtifacts
for Artifact artifact : ListArtifacts do
  if Interest(artifact) then
    ArtifactList.add(artifact);
  end if
end for
```

À partir de cette décomposition et de la perception de l'agent, il est alors possible de voir émerger de nouvelles possibilités au sein de l'environnement que nous nommerons opportunités dispositionnelles notées τ.

DÉFINITION 4.2 (Émergence d'opportunité dispositionnelle). — *Soit W(un système agent-objet) = (Obj, A) et W_{pq}(un système agent-artefact) = (Art_p, A) composé d'un artefact Art possédant une disposition p. Une opportunité dispositionnelle existe si et seulement s'il existe une opportunité τ, telle que :*

1. $W = interest(Obj, A)$ *ne possède pas* τ
2. $W_{pq} = interest(Art_p, A_q)$ *possède* τ

avec $Art_p \in Obj$

Ces opportunités dispositionnelles représentent alors un moyen de proposer des services supplémentaires sans l'ajout d'objets.

5. Modèle d'architecture

Sur la base de ces définitions, nous proposons un modèle d'architecture qui fait intervenir trois types d'entités basées sur le concept A&A : l'agent, l'espace de travail et l'artefact. Les rôles de l'agent et de l'espace de travail étant globalement inchangés, nous allons directement nous concentrer sur les ajouts conceptuels liés à la décomposition des objets en un arbre d'artefact dispositionnel. Nous nous intéresserons donc aux huit classes principales d'artefact (Figure 5.1) : Artifact, Artifact composite, Artifact principal, Artifact final, Artifact capteur, Artifact actionneur, Artifact service.

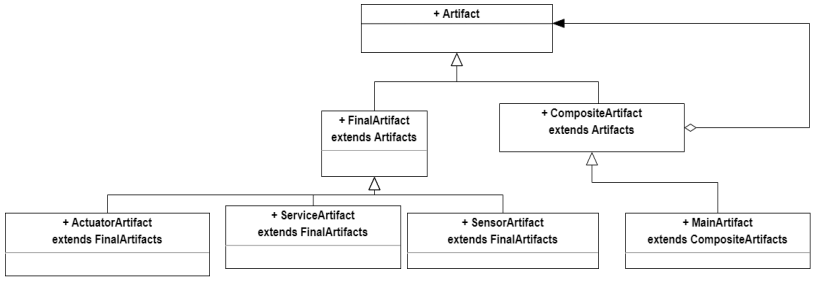

FIGURE 5.1 – Hiérarchie entre les classes proposées

- **Artifact.** Cette classe est le composant principal de la décomposition. Globalement, elle définit toutes les primitives utiles à la programmation du comportement observable des artefacts en fonction des frameworks ou des environnements d'implémentation. Cette première classe permet d'établir le lien de dépendance entre les artefacts, ainsi que son état observable au sein des espaces de travail. Comme précisé dans l'état de l'art, les artefacts mis en avant ici ne sont pas identiques au modèle A&A. Nous avons opté pour une interface unifiée afin de simplifier l'interaction entre l'agent et son environnement. Par exemple dans le cas d'un artefact de type light une action se soldera par un allumage/extinction là où pour un type sonnerie, cela résultera un retentissement de la sonnerie. Cette particularité est possible grâce au fait que la décomposition sous forme de dispositions permet de mettre en avant des capacités que l'on peut qualifier d'atomiques.
- **Composite Artifact.** La classe Composite Artifact est avant tout un conteneur d'artefact. Son rôle est de garder une trace de la structure de l'objet connecté d'origine. Il est également possible qu'elle soit elle-même sous la gestion d'un artefact si elle est aussi un élément composant.
- **Main Artifact.** La classe Main Artifact représente virtuellement l'objet connecté et contient les informations relatives à l'objet (nature, état, adresse mac, etc.). Elle est aussi responsable de l'accès aux artefacts.
- **Final Artifact.** Les éléments de cette classe représentent les artefacts directement rendus accessibles dans l'espace de travail. Toutes les classes qui en héritent (Sensor Artifact, Actuator Artifact, Service Artifact) peuvent être vues comme les feuilles de l'arbre de décomposition de l'objet originel et ne peuvent donc pas avoir d'artefacts qui les composent.

5.1. PROPRIÉTÉS

Grâce aux classes d'artefact vues précédemment, il devient possible de décrire virtuellement les dispositions d'un objet sous la forme d'un arbre. Au sein de cette description, les feuilles représentent alors les dispositions de l'objet et les nœuds internes, la structure de sa décomposition. La Figure 5.2 illustre une décomposition virtuelle d'un smartphone en un arbre dont la racine représente l'objet originel (MainArtifact),

les nœuds intermédiaires la structure de la décomposition (compositeArtifact) et les feuilles les dispositions pouvant être mis à disposition (finalArtifact). Suite à cela,

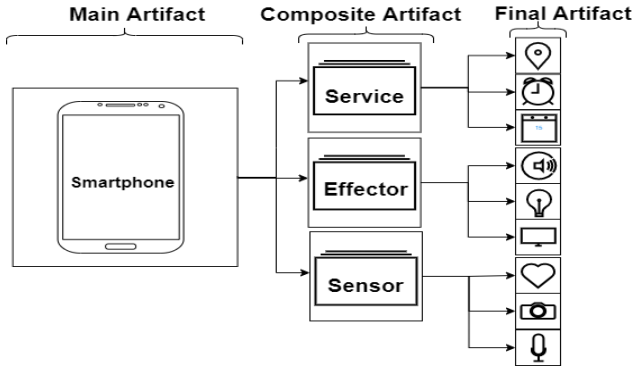

FIGURE 5.2 – Exemple de décomposition

l'objet n'est plus représenté de manière monolithique, mais est décomposé en un ensemble d'artefacts dispositionnels. Il devient alors possible de les mutualiser au sein de workspaces afin de proposer des services qui utilisent pleinement les possibilités de notre environnement connecté. D'un point de vue conceptuel, contrairement à l'utilisation d'agents, la mutualisation repose sur un paradigme cohérent. De plus, le haut niveau d'abstraction permet d'éviter la surabondance d'intergiciels et ouvre les possibilités d'interaction avec ces dispositions, que cela soit par des canaux hétéroclites (via zigbee, wifi, etc.) ou via des normes existantes de plus haut niveau comme le Web des Objets (par le biais de requêtes HTTP) [25, 43, 11]. Cette caractéristique permet au concept d'être utilisé au sein de n'importe quelles solutions permettant la mise en relation de dispositifs distants.

6. Une implémentation sur Android

Une implémentation sur Android a été réalisée afin de mettre à l'épreuve la faisabilité du modèle. Le choix de la plateforme Android n'est pas anodin. En effet en 2019, Google a annoncé avoir atteint les 2,5 milliards de terminaux Android actifs. Ce qui en fait le système d'exploitation le plus répandu. De plus, le système est conçu pour les objets connectés tels que les smartphones, tablettes tactiles, ordinateurs, télévisions (Android TV), voitures (Android Auto), Chromebook et autres montres connectées (Wear OS). Ce panel de produits illustre parfaitement les types d'appareils les plus à même d'être porteur de capteurs et d'effecteurs dont les usages ne sont pas mutualisés. Afin de mettre en place ce modèle générique, nous avons basé la phase d'implémentation sur la plateforme « Software Kit for Ubiquitous Agent Development » (SKUAD) qui permet de créer des agents ambiants capables de manipuler des capteurs et des effecteurs [10] dont les performances et la conception permettent un usage embarqué. Nous allons en premier lieu décrire les spécificités de cette implémentation sur Android avant de les illustrer par la phase de test.

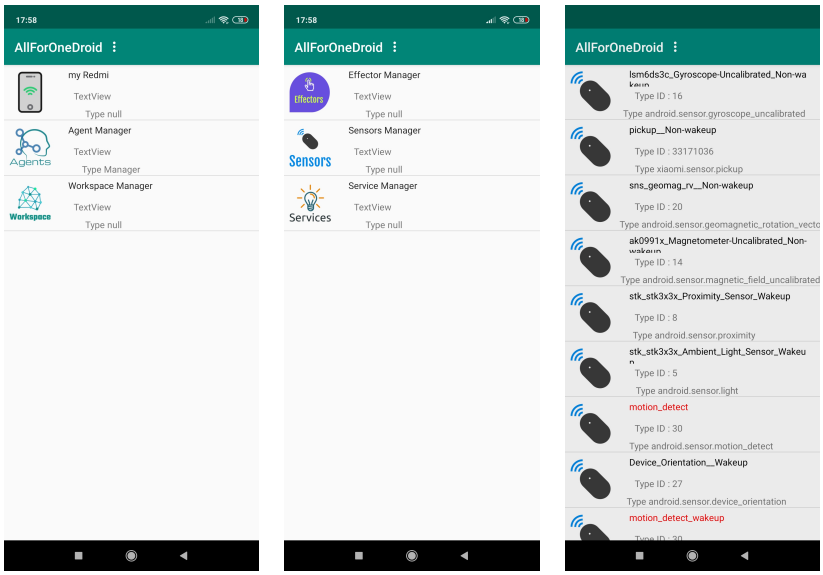

FIGURE 6.1 – Une application sur la plateforme Android

6.1. IMPLÉMENTATION

L'application (format APK) (Figure 6.1), est suffisamment légère (9.13 Mo), pour une consommation de 30 Mo de RAM en moyenne pour une séance de 3 h d'utilisation. Elle permet, la détection de tous les capteurs embarqués au sein de l'appareil et propose, pour l'instant, sept effecteurs et deux services.

Lors de sa mise en route, si un composant est disponible et que les autorisations le permettent, le système procède alors à la création des artefacts en tant qu'instance de la classe associée. Il revient alors à l'utilisateur de choisir, via l'interface graphique, quels artefacts mettre à disposition et quels agents activer au sein du workspace.

Lors des phases de test, AFFORD a été testé sur différents appareils Android (Smartphone et tablette) afin de mettre en avant la présence de dispositions. Cette première phase a permis de confirmer que de nombreuses possibilités étaient cachées au sein de nos smartphones. Outre les effecteurs et services pouvant être implémentés, nous pouvons voir dans le tableau (Fig.6.2) que nativement ils étaient capables de proposer à minima une vingtaine d'artefacts de type Sensor.

Du point de vue de l'implémentation, la classe Artifact comporte, en plus des primitives de base liées au comportement observable des artefacts, l'ensemble des méthodes spécifiques liées aux contraintes d'Android telles que la création d'IHM, les accès aux permissions et au stockage de l'appareil. En héritant de la classe Artifact, nous avons implémenté les autres classes décrites dans le modèle. Dans le but d'avoir une vue globale, nous allons nous concentrer sur la classe AndroidMainArtifact (classe

Marques	Samsung	Motorola	Samsung	Samsung	Xiaomi
Modèle	Galaxy s8	one zoom	SM-T870	SM-N960F	Redmi Note 8T
RAM (Go)	4	4	8	6	4
SensorArtifact	26	44	20	27	22

FIGURE 6.2 – Quelques exemples de smartphones testés

fille de MainArtifact) ainsi que sur celles héritant de FinalArtifact. Conformément à l'algorithme (Algorithm 1) et afin de s'adapter à tous types d'appareils Android, la classe AndroidMainArtifact a pour fonction principale d'analyser les possibilités offertes par l'objet connecté et de demander à l'utilisateur les autorisations associées à leurs fonctionnements. Les artefacts liés aux dispositions sont alors créés dynamiquement et automatiquement en introspectant l'appareil. Les créations d'artefacts ont été déléguées à trois managers (CompositeArtifact) qui sont spécialisés dans la création d'un type d'artefact. On peut voir un exemple de création de sensorArtifacts au sein du manager associé dans l'exemple 6.3.

```
public static ArtifactImplSensorsManager CreateDefaultArtifactSensorArtifactsManager (String
    name, Activity activity ) {
      ArtifactImplSensorsManager. activity = activity ;
      SensorManager sensorManager = (SensorManager)
          activity . getSystemService(SENSOR_SERVICE);
      ArtifactImplSensorsManager AASM = new ArtifactImplSensorsManager(name,sensorManager);
      List<Sensor> list = sensorManager. getSensorList (Sensor.TYPE_ALL); //On liste
          l'ensemble des capteurs ( dispositions )
      if ( list != null ) {
          Iterator iterator = list . iterator ();
          while ( iterator .hasNext()) { // On parcourt la liste des dispositions de type
              capteur
              Sensor sensor = (Sensor) iterator .next ();
              AASM.nbmax++;
              ArtifactSensorImpl devicesensor = new ArtifactSensorImpl (sensor .getName(),
                  Context.SENSOR_SERVICE, sensor.getType(),
                  SensorManager.SENSOR_DELAY_NORMAL,sensor.getStringType()); //On
                  encapsule la disposition dans un artefact
              devicesensor . setSensor (sensor );
              devicesensor .addDeviceMaster(AASM);
          }
          return AASM;
      }
      return null ;
  }
```

FIGURE 6.3 – Méthode de création des SensorArtifact

```
public interface SensorArtifact extends FinalArtifact {
    void setObservableState ( String state ); // Modification des valeurs observables
    void setObservableState ( byte [] state );
    byte [] getObservableState () ; // Fonction accessible aux agents permettant de consulter
        la valeur courante
}
public interface ActuatorArtifact extends FinalArtifact {
    boolean genEvent() ; // Fonction par dÃ©faut de demande d'action
    boolean genEvent( int ID, int new_value);
    boolean genEvent( int ID, float new_value);
    boolean genEvent( int ID, String new_value);
        [...]
}
```

FIGURE 6.4 – Interfaces SensorArtifact/ActuatorsArtifact

Au sein du programme, les artefacts sont créés au travers l'instanciation d'interface héritant de FinalArtifact : ActuatorArtifact, SensorArtifact (Figure 6.4), ServiceArtifact.

Nous pouvons donner comme exemple les classes Light (ActuatorArtifact) et VoiceRecognition (ServiceArtifact) représentant respectivement les catégories d'artefacts liés à la production de lumière et à la reconnaissance vocale. Si un composant est disponible et que les autorisations le permettent, le système autorise alors à la création des artefacts en tant qu'instance de la classe associée. On peut voir un exemple de création d'artefacts encapsulant un capteur au travers du code de $ArtifactSensorImpl$ (Figure 6.5).

Il revient alors à l'utilisateur de choisir, via l'interface graphique, quels artefacts mettre à disposition et quels agents activer au sein du workspace. L'utilisateur et les agents peuvent aussi voir et utiliser les artefacts proposés par d'autres appareils, mais ils ne seront pas en mesure de modifier leurs caractéristiques ou leurs visibilités dans le workspace.

6.2. MISE EN CONTEXTE

Pour illustrer les principaux concepts proposés, nous allons présenter un exemple d'utilisation. Les appareils utilisés lors de cette phase de test sont : un smartphone de type Samsung galaxy s8, une tablette de type Samsung tab A (2016) et un ordinateur portable de type Acer TravelMate P257 (Figure 6.6).

Au terme de la phase d'analyse des dispositions des appareils sous Android le galaxy S8 propose vingt-six Sensors et le Tab A en propose cinq. Avec deux appareils Android, si l'on ajoute les Services (2) et les Actuators (6) disponibles sur chaque dispositif, nous en sommes donc déjà à quarante-neuf artefacts pouvant être mis à disposition de l'utilisateur et des agents.

```
public class ArtifactSensorImpl extends FinalDeviceImpl implements SensorEventListener,
    SensorArtifact {
  int sensor_id;
  byte[] value;
  // Constructeur prenant les paramÃ©tres permettant la crÃ©ation de l' artefact
  public ArtifactSensorImpl(String vname, String SENSOR_SERVICE, Integer SENSOR_TYPE,
      int SENSOR_DELAY, String Stype) {
    super(vname, SENSOR_TYPE, true);
    this.SENSOR_SERVICE=SENSOR_SERVICE;
    this.SENSOR_TYPE=SENSOR_TYPE;
    this.SENSOR_DELAY=SENSOR_DELAY;
    this.type=Stype;
  }
  // Fonction de mise a jour des informations observables de l' artefact
  @Override
  public void onSensorChanged(SensorEvent event) {
    if(event.sensor.getType()== this.SENSOR_TYPE){
      setObservableState(state);
    } } }
}
```

FIGURE 6.5 – Implémentation d'un SensorArtifact

Modèle	Samsung galaxy s8	Galaxy Tab A (2016)	Acer TravelMate P257
Cadence Processeur	2,3 GHz	1.6GHz	2,4 GHz
Système	Android 9.0	Android 7.0	Ubuntu 16.04
RAM	4 Go	2 Go	6Go
Stockage	64 Go	16 Go	500Go
Connectivité (indicatif)	4G, NFC, Bluetooth 5.0, GPS, Wi-Fi, reconnaissance d'iris/faciale/digitale, USB Type C	ANT+, USB 2.0, GPS, prise jack 3.5mm Stereo,Bluetooth v4.2, Wi-Fi 802.11	Wi-Fi AC, Bluetooth 4.0, Webcam, Jack 3.5mm, RJ45, USB 2.0, USB 3.0, HDMI Femelle, VGA
Dimensions	148.9×68.1×8 mm pour 152 g	254.2×155.3×8.2 pour 525 g	256 × 381.6 mm × 29.2 mm pour 2.4 kg

FIGURE 6.6 – Descriptif des appareils utilisés lors de la phase de test

Afin d'illustrer le concept, nous avons effectué une utilisation de l'accéléromètre, servant habituellement pour déterminer l'orientation de l'écran ou stabiliser la prise de photographie, en tant que détecteur de chocs (Figure 6.7). Ainsi, l'entité intelligente, dans cette implémentation un agent SKUAD, devient à même, en ajustant les paramètres de sensibilité, de discriminer différents types de chocs (chute, hauteurs, sens, présence d'un choc final). En cas de choc, l'agent cherchera à attirer l'attention de l'utilisateur via le flash ou le vibreur des appareils Android ou encore en faisant clignoter l'écran de l'ordinateur. Si l'agent n'a pas de réponse, il peut alors utiliser la synthèse vocale

ainsi que la reconnaissance vocale afin de dialoguer avec un utilisateur. Pour ce cas, l'agent pose une suite de questions fermées donnant le choix à la personne interrogée de répondre parmi un ensemble de réponses prédéfinies. Chaque réponse prononcée par l'utilisateur sera transformée en texte puis analysée par l'agent afin de faire un constat de la situation.

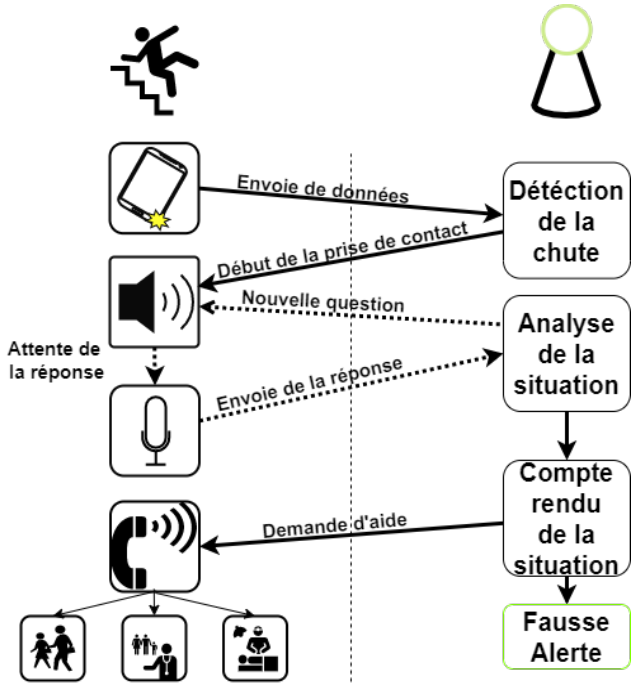

FIGURE 6.7 – Digramme illustrant le cas de la gestion de chute

6.3. DES RÉSULTATS ENCOURAGEANTS

À la vue des résultats obtenus, nous avons constaté qu'un simple smartphone ou une tablette suffit à animer bon nombre de services. Nous pouvons donner comme exemple le cas, pourtant simple, présenté précédemment où nous nous trouvons virtuellement et schématiquement au sein d'un système ayant 3 processeurs, 3 batteries, 12 Go de RAM ainsi de 580 Go de stockage (Figure 6.2). Cette puissance est bien supérieure à celle requise dans un usage courant et peut représenter un socle solide en termes de capacité de traitement des informations récupérées par les entités de captation. Cela permettrait notamment de diminuer le nombre de batteries, d'écrans et de composants liés au calcul au sein de ces appareils disséminés (RAM, processeur, stockage). Il est difficile de jauger exactement le gain en termes de matière première, qui va dépendre des solutions commercialisées, mais nous avons pu montrer ici que l'ajout de fonctionnalité peut se faire sans ajout d'équipement (centre de commande, bracelet, patch, pendentif muni

d'un détecteur de mouvement, effecteur, etc.). Notre élan technologique déraisonné pourrait alors être freiné et la dynamique actuelle cherchant à rendre l'environnement intelligent passerait alors, non pas par une intégration systématique de puissance de calcul au sein de chaque objet, mais plutôt par un usage plus raisonné et mutualisé des possibilités de nos appareils. De plus, cette solution offre une alternative au cloud computing, car les traitements peuvent être faits en local sur les périphériques qui en ont la capacité. Cela permet de diminuer l'empreinte énergétique, puisque qu'on ne sollicite ni le réseau internet et ni les data center, tout en limitant les fuites de données personnelles. Enfin, il paraît plus facile pour les concepteurs des services de n'avoir qu'à considérer les dispositions à un plus haut niveau d'abstraction, sans avoir à se préoccuper autant du niveau matériel.

7. Conclusion et Perspectives

Dans cet article, nous avons proposé une nouvelle vision de l'interaction avec notre environnement de plus en plus informatisé. À l'heure actuelle où la tendance converge vers une communautarisation des technologies, cette approche a pour but de limiter l'impact environnemental des objets connectés en suggérant leurs mutualisations. Grâce à un modèle d'architecture inspiré du concept d'artefact, nous proposons de mettre en avant virtuellement les dispositions de nos appareils connectés afin de diminuer les redondances au sein d'un environnement et ainsi voir dans nos objets connectés bien plus que leurs fonctionnalités initiales imposées par leurs concepteurs Nous avons effectué une preuve du concept au travers de l'implémentation du modèle sous Android. Cette étape nous a permis de mettre en avant le fait qu'une solution serait une utilisation plus intelligente de la multitude d'opportunités non exploitées résidant au sein de nos environnements. Nous pensons désormais étendre la phase de test du modèle à d'autres types d'objets tels que les télévisions, les montres, ainsi qu'à d'autre type d'usage. Car, par cette mutualisation, nous sommes à même de briser cet aspect « boite noire » pour dissocier l'objet, et ces dispositions, de l'usage traditionnel que l'on en fait. On peut alors y voir un moyen de monitorer quelqu'un en téléassistance (Télécommunications), d'améliorer la résilience d'un service en proposant des alternatives ou encore de désengorger le réseau en permettant d'exécuter des services en périphérie (Edge computing [42]). D'autre part, nous pensons que les fonctions offertes par les artefacts seraient enrichies par l'ajout d'une sémantique. En effet, de nombreuses recherches ont démontré l'intérêt des ontologies dans le cadre de l'exploitation des objets par un logiciel [15, 34, 38]. L'utilisation d'une ontologie serait donc tout indiquée dans le cas d'une mutualisation pour permettre une description précise et cohérente des possibilités offertes par les artefacts. Enfin, nous pensons que notre modèle permettrait d'enrichir les approches visant à mettre à disposition les fonctionnalités des objets connectés, comme c'est le cas pour le WoT. En mutualisant les composants, il devient possible de fournir plus de fonctionnalités, plus de redondance (et donc de robustesse), tout en évitant la consommation des matières premières rares qui sont fournies de manière limitée par notre planète.

Bibliographie

[1] Z. Afoutni, « Un modèle multi-agents pour la représentation de l'action située basé sur l'affordance et la stigmergie », Thèse, Université de la Réunion, 2015.

[2] M. Baldauf, S. Dustdar & F. Rosenberg, « A survey on context-aware systems », *International Journal of Ad Hoc and Ubiquitous Computing* **2** (2007), n° 4, p. 263-277.

[3] M. Böhlen & H. Frei, « Ambient Intelligence in the City overview and new perspectives », in *Handbook of ambient intelligence and smart environments*, Springer, Boston, MA, 2010, p. 911-938.

[4] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci & A. Santi, « Multi-agent oriented programming with JaCaMo », *Science of Computer Programming* **78** (2013), n° 6, p. 747-761.

[5] F. Bordage, « Study – The environmental footprint of the digital world », GreenIT, 2019, 39 pages.

[6] N. Bouhaï & I. Saleh, *Internet of things : evolutions and innovations*, John wiley & sons, 2017.

[7] L. Cailloce, « Numérique : le grand gâchis énergétique », CNRS Le journal, 2018.

[8] K. Cetnarowicz, « From Algorithm to Agent », in *Computational Science – ICCS 2009* (Berlin, Heidelberg) (G. Allen, J. Nabrzyski, E. Seidel, G. D. van Albada, J. Dongarra & P. M. A. Sloot, éds.), Springer Berlin Heidelberg, 2009, p. 825-834.

[9] S. A. DeLoach, « Multiagent systems engineering : A methodology and language for designing agent systems », Tech. report, Department of Electrical and Computer Engineering of Air Force Institute of Technology, 1999.

[10] P. Denis, « SKUAD, Software Kit for Ubiquitous Agent Development », 2018, `http://skuad.onover.top/`.

[11] T. S. Dillon, H. Zhuge, C. Wu, J. Singh & E. Chang, « Web-of-things framework for cyber–physical systems », *Concurrency and Computation : Practice and Experience* **23** (2011), n° 9, p. 905-923.

[12] European Commission, « EU Commission, Critical raw materials, Internal Market, Industry, Entrepreneurship and SMEs », 2020, `https://ec.europa.eu/growth/sectors/raw-materials/specific-interest/critical_en`.

[13] J. Ferber, *Les systèmes multi-agents : vers une intelligence collective*, I.I.A. Informatique intelligence artificielle, InterEditions, France, 1995.

[14] R. Fontaine, N. Aky, R. Courdier & D. Payet, « Vers une utilisation éco responsable des objets connectés par la mutualisation de leurs composants physiques : Une approche basée sur le concept d'artefact », in *Actes de la conférence nationale en intelligence artificielle CNIA - PFIA 2020, Angers, 2020*, vol. 23, CNIA, 2020, p. 38.

[15] A. Freitas, A. R. Panisson, L. Hilgert, F. Meneguzzi, R. Vieira & R. H. Bordini, « Integrating ontologies with multi-agent systems through CArtAgO artifacts », in *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, vol. 2, IEEE, 2015, p. 143-150.

[16] B. Gateau, Y. Naudet & J. Rykowski, « Ontology-based smart IoT engine for personal comfort management », in *2016 11th International Workshop on Semantic and Social Media Adaptation and Personalization (SMAP)* (Thessaloniki, Greece), IEEE, IEEE, 2016, p. 35-40.

[17] J. J. Gibson, « The theory of affordances », in *Perceiving, acting, and knowing : toward an ecological psychology* (R. E. Shaw & J. Bransford, éds.), Lawrence Erlbaum Associates, Hillsdale, N.J., 1977, p. 67-82.

[18] IEEE, *Middleware and application adaptation requirements and their support in pervasive computing*, Los Alamitos, CA, USA, IEEE Computer Society, 2003.

[19] N. R. Jennings, « On agent-based software engineering », *Artificial intelligence* **117** (2000), n° 2, p. 277-296.

[20] P. Kaiser, E. E. Aksoy, M. Grotz & T. Asfour, « Towards a hierarchy of loco-manipulation affordances », in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2016, p. 2839-2846.

[21] P. Kaiser & T. Asfour, « Autonomous detection and experimental validation of affordances », *IEEE Robotics and Automation Letters* **3** (2018), n° 3, p. 1949-1956.

[22] P. Kaiser, N. Vahrenkamp, F. Schültje, J. Borràs & T. Asfour, « Extraction of whole-body affordances for loco-manipulation tasks », *International Journal of Humanoid Robotics* **12** (2015), n° 03, article no. 1550031.

[23] A. Kameas & I. Calemis, « Pervasive Systems in Health Care », in *Handbook of Ambient Intelligence and Smart Environments* (H. Nakashima, H. Aghajan & J. C. Augusto, éds.), Springer US, Boston, MA, 2010, p. 315-346.

[24] R. Khan, S. U. Khan, R. Zaheer & S. Khan, « Future internet : the internet of things architecture, possible applications and key challenges », in *2012 10th international conference on frontiers of information technology*, IEEE, 2012, p. 257-260.

[25] T. Kindberg, J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris et al., « People, places, things : Web presence for the real world », *Mobile Networks and Applications* **7** (2002), nº 5, p. 365-376.

[26] K. E. Kjær, « A Survey of Context-Aware Middleware », in *Proceedings of the 25th Conference on IASTED International Multi-Conference : Software Engineering* (USA), SE'07, ACTA Press, 2007, p. 148-155.

[27] F. Klügl, « Using the affordance concept for model design in agent-based simulation », *Annals of Mathematics and Artificial Intelligence* **78** (2016), nº 1, p. 21-44.

[28] J. Kwan, Y. Gangat, D. Payet & R. Courdier, « A agentified use of the Internet of Things », in *Full Paper in 9th IEEE International Conference on Internet of Things(iThings 2016)*, IEEE CS, 2016.

[29] G. B. Laleci, A. Dogac, M. Olduz, I. Tasyurt, M. Yuksel & A. Okcan, « SAPHIRE : A Multi-Agent System for Remote Healthcare Monitoring through Computerized Clinical Guidelines », in *Agent Technology and e-Health* (Basel) (R. Annicchiarico, U. Cortés & C. Urdiales, éds.), Birkhäuser Basel, 2008, p. 25-44.

[30] Z. Maamar, N. Faci, K. Boukadi, E. Ugljanin, M. Sellami, T. Baker & R. Angarita, « How to agentify the Internet-of-Things ? », in *2018 12th International Conference on Research Challenges in Information Science (RCIS)*, IEEE, 2018, p. 1-6.

[31] Z. Maamar, N. Faci, S. Kallel, M. Sellami & E. Ugljanin, « Software agents meet internet of things », *Internet Technology Letters* **1** (2018), nº 3, article no. e17.

[32] N. Mohamed, J. Al-Jaroodi & I. Jawhar, « Middleware for Robotics : A Survey », in *RAM* (Chengdu, China), IEEE, 2008, p. 736-742.

[33] A. Omicini, A. Ricci & M. Viroli, « Artifacts in the A&A meta-model for multi-agent systems », *Autonomous agents and multi-agent systems* **17** (2008), nº 3, p. 432-456.

[34] A. R. Panisson, A. Freitas, D. Schmidt, L. Hilgert, F. Meneguzzi, R. Vieira & R. H. Bordini, « Arguing about task reallocation using ontological information in multi-agent systems », in *12th International Workshop on Argumentation in Multiagent Systems*, vol. 108, 2015.

[35] C. E. Pantoja, H. D. Soares, J. Viterbo & A. El Fallah-Seghrouchni, « An Architecture for the Development of Ambient Intelligence Systems Managed by Embedded Agents », in *SEKE*, 2018, p. 215-214.

[36] C. Ramos, G. Marreiros, R. Santos & C. F. Freitas, « Smart Offices and Intelligent Decision Rooms », in *Handbook of Ambient Intelligence and Smart Environments* (H. Nakashima, H. Aghajan & J. C. Augusto, éds.), Springer US, Boston, MA, 2010, p. 851-880.

[37] A. Ricci, M. Piunti, M. Viroli & A. Omicini, « Environment programming in CArtAgO », in *Multi-agent programming*, Springer, 2009, p. 259-288.

[38] A. Ricci, M. Viroli & A. Omicini, « CArtA gO : A Framework for Prototyping Artifact-Based Environments in MAS », in *Environments for Multi-Agent Systems III* (D. Weyns, H. V. D. Parunak & F. Michel, éds.), vol. 4389, Springer Berlin Heidelberg, 2007, p. 67-86.

[39] M. L. Roloff, M. R. Stemmer, J. F. Hübner, R. Schmitt, T. Pfeifer & G. Hüttemann, « A multi-agent system for the production control of printed circuit boards using JaCaMo and Prometheus AEOlus », in *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, IEEE, 2014, p. 236-241.

[40] S. J. Russell, « Rationality and intelligence », *Artificial Intelligence* **1** (1997), nº 94, p. 57-77.

[41] L. Seinturier, P. Merle, R. Rouvoy, D. Romero, V. Schiavoni & J.-B. Stefani, « A Component-Based Middleware Platform for Reconfigurable Service-Oriented Architectures », *Software : Practice and Experience* **42** (2012), nº 5, p. 559-583.

[42] W. Shi & S. Dustdar, « The promise of edge computing », *Computer* **49** (2016), nº 5, p. 78-81.

[43] M. V. Trifa, « Building blocks for a participatory web of things : devices, infrastructures, and programming frameworks », Thèse, ETH Zurich, 2011.

[44] M. Weiser, « Some computer science issues in ubiquitous computing », *Communications of the ACM* **36** (1993), n° 7, p. 75-84.
[45] M. Wooldridge, « Agent-based software engineering », *IEE Proceedings-software* **144** (1997), n° 1, p. 26-37.

Abstract. — On a global scale, the environmental footprint of digital technology represents a continent 2 to 3 times the size of France and 5 times the weight of the French car fleet. The major cause of this ecological disaster would be the overabundance of connected objects, linked in particular to an irrational use amplified by a logic of overconsumption. In order to limit the negative impact of this overabundance and to optimize the computing potential emerging from our environment, we propose in this article an architecture model, based on the concept of artifact, in order to promote the mutualization of the components present within our connected devices. To show its feasibility and advantages related to its use, we propose an implementation of this model for the Android platform.

Keywords. — Green T, Ubiquitous computing, mutualization, Agents & Artifacts metamodel, Real-time multi-agents systems.

Manuscrit reçu le 1er mars 2021, révisé le 5 août 2021, accepté le 4 décembre 2021.