Software Engineering for Internet of Things

Résumé

C’est en 1968 au cours de la conférence NATO que Douglas McIlroy introduit pour la première fois la notion de composant logiciel. Ce paradigme de programmation promeut la réutilisation massive et la fabrication de logiciel par assemblage de composants. Mais aujourd’hui, le domaine des composants logiciel ne tient pas encore ses promesses en terme de réutilisabilité et de portabilité des composants. En effet, il est souvent nécessaire de créer un code de glue pour permettre l’assemblage de composants existants. Par conséquent, il est souvent moins coûteux de réimplanter un composant plutôt que de réutiliser des composants existants. En outre la construction de composants par compositions d’autres composants n’est pas de mise dans tous les modèles.

Avec la démocratisation des technologies dans l’habitat (domotique), de nouveaux besoins en terme de génie logiciel se font à la foi présents et pressants, pour faire face aux problématiques d’hétérogénéité des technologies utilisées, d’évolutivité du logiciel, d’adaptation à son environnement au cours de son exécution, et bien d’autres. Afin de répondre au plus juste aux besoin de chacun, et aux spécificités de chaque installation pour faire face à des problématiques allant du handicap aux économies d’énergies, les systèmes logiciels de gestion et de contrôle de ces installations doivent être fabriqués rapidement, être fiables et personnalisables par l’utilisateur lui-même.

La thèse défendue dans ce document est que l’utilisation conjointe d’un modèle de composant logiciel contraint inspiré par l’électronique, et d’un environnement d’exécution ouvert et non contraint, permet de résoudre des problèmes d’interopérabilité, d’évolutivité, d’adaptation et de fiabilité des systèmes de contrôle/gestion domotique.

Pour supporter cette thèse, un cadre de développement d’applications domotiques appelé EnTiMid a été réalisé. Son environnement d’exécution Kevoree s’appuie sur les paradigmes de programmation orientée services offrant une grande souplesse. A plus haut niveau, l’espace de conception réutilise des principes des modèles à composants et modèles à l’exécution (Models@Runtime) à des fins de gestion et de validation de l’application. Fortement emprunt du domaine de l’électronique, EnTiMid permet nativement l’interopérabilité, la réutilisabilité et offre une intégration aisée de produits domotiques et de services disponibles à travers Internet.

C’est à travers cette facilité d’intégration de l’Internet des Objets (IoT) et de l’Internet des Services (IoS) qu’EnTiMid ouvre le champ des possibles dans le domaine des technologies de l’habitat et de leur utilisation pour l’assistance à domicile de personnes dépendantes.

L’adéquation d’EnTiMid avec les problématiques de terrain a été validée par son intégration dans un projet de la métropole Rennaise autour des problématiques d’aide à domicile pour les personnes âgées ou dépendantes. Cette intégration permet ainsi de valider la thèse proposée.

Summary

Table des matères

I) Context of the work, identification of requirements 4

A) Context of the work 4

i) Home Automation 4

ii) The AAL Context 4

B) Relevant problems 4

C) Existing Approaches 5

i) Model Driven Engineering 5

ii) Component models 5

iii) Service Oriented Architectures 5

iv) Agent Based 5

v) Internet Of \* and the Cloud 5

D) Generalization and synthesis 6

II) Thesis and realization 6

A) La these, explication du concept 6

B) Naïve 6

C) Using M@RT 6

D) EnTiMid component model 6

III) Architecture Synthesis 7

A) Static: FD, CT, CI 7

B) Dynamic; MAD Editor, BEPL 7

C) Synthétiseur, Compilateur 7

IV) Validation 7

A) A travers différents points de vue 7

B) IDA 7

C) ITI Project 7

D) Industrial contacts 7

E) Public Demonstrations 7

i) FDLS 7

ii) Inauguration ESIR 7

iii) AERES 7

F) Spin Off 7

V) Conclusion 7

A) Synthesis board 7

B) IoT-IoS 8

VI) Perspectives 8

A) Open Control/Command operating system ^^ 8

B) EUP 8

C) FuzyLogic 8

D) Reste à faire 8

VII) Bibliography 8

VIII) Glossary 8

IX) Definitions 9

# Context of the work, identification of requirements

Les travaux présentés dans ce document utilisent les technologies et techniques de développement de pointe pour répondre à une problématique sociétale ; celle de l’assistance et du maintien à domicile (MAD) de personnes dépendantes. Cette section brosse dans un premier temps un panorama des préoccupations dans ce domaine. Ce panorama mènera dans une seconde partie à une spécification des exigences en terme de génie logiciel.

Enfin, face à ces exigences, un sondage des solutions apportées par différentes approches nous permettra de préciser les lacunes à combler pour que les techniques de développement logiciel puissent offrir une réponse efficace à l’ensemble des préoccupations du MAD.

## Context of the work

### Home Automation

Présentation rapide des technologies de la domotique, des différents domaines d’application. Bref exposé des orientations de développement de la domotique dans plusieurs pays. Terminer par la France où les deux axes majeures sont les économies d’énergie et le MAD.

#### Pistes de développement dans le monde

Corée => Jeux videos  
Japon => Salle de bain (faire couler un bain depuis son telephone  
Etats-unis => Sécurité des biens

#### Pistes de développement en Europe

Italie => Design intérieur  
Allemagne  
Royaume Uni

#### Cas de la france

Economies d’énergies  
AAL

### The AAL Context

Présentation du maintien à domicile. Les enjeux, les besoins.

En France: \cite{BOULMIER:2009}\cite{ASSAD:2010}\cite{BERRA:2010}\cite{RIALLE:2007}

## Relevant problems

Des besoins précédent, extraction de problèmes de GL.

Interoperability  
Opening  
Adaptation  
Evolution  
Variability Management  
Remote Control  
Distribution  
Safety & Security  
Acceptability

## Existing Approaches

Confrontation de différentes approches / solutions techniques aux problématiques levées

### Model Driven Engineering

La thèse s’est déroulée au sein de l’Equipe TRISKELL, équipe dont le cœur de métier est l’ingénierie des modèles. De fait, l’ensemble de cette thèse repose sur des fondements issus de l’ingénierie des modèles. Cette section fait le point sur ces fondations.

#### Models@Runtime

### Component models

Au dessus de ces fondations, la notion de composant logiciel est apparue. Il est essentiel d’en saisir les concepts, car la thèse défendue dans ce document apporte une évolution dans ce domaine.

#### SCA Fractal Connector generation

### Service Oriented Architectures

Le domaine du génie logiciel est un domaine actif. Faisant suite aux modèles de composants, le paradigme d’architectures orientées services apparaît comme un axe intéressant pour le développement d’applications autour de l’Internet en général.

#### Service Based

##### WS REST OSGi

### Agent Based

#### UBIWARE

### Internet Of \* and the Cloud

#### Internet Of Things

L’Internet des objets désigne une approche dans laquelle les objets de tous les jours atteignent un niveau de maturité suffisant pour leur permettre d’interagir les uns avec les autres et rendre des services à valeur ajoutée. Etant encore dans ses prémices, cette approche se cherche encore des outils logiciels de développement pour décrire les interactions, et les services rendus. Il est fort probable que la solution réside en un amalgame subtil entre les architectures à composants et celles à services.

#### Internet Of Services

A l’image de son compère, l’Internet des Services promeut l’interaction de services disponibles à travers Internet. L’orchestration d’un ensemble de services unitaires, rend possible la création de services plus complexes, à grande valeur ajoutée, et ce à moindre coût.

#### The Cloud

Le concept de cloud détache les services et les objets d’une localisation precise. Autrement dit, les services et objets actifs sont diffus dans l’environnement d’exécution, et utilisables sans avoir à connaitre exactement leur localisation.

## Generalization and synthesis

Résumé dans un tableau des réponses des différentes approches / problématiques  
Identification du trou de souris dans lequel cette these se foure.

# Thesis and realization

Les bases conceptuelles posées, les exigences remontées par le terrain et l’étude crituque des réponses apportées par les approches existantes permettent d’expliciter la these défendue dans ce document.

Après une presentation conceptuelle de l’approche, cette section décrit et detaille les options d’implementation envisages, puis rapporte la realisation finale.

## La these, explication du concept

Se rapprocher du modèle électronique

Découpage clair Types/instances

Elévation des contraintes au niveau modèle

Vérifications de contraintes

## Naïve

Faire des objets conformes, avec API included, modèle par introspection

Bien que permissive, ce mocèle d’architecture ne permet pas de faire de verifications et pas de gestion par l’extérieur

## Using M@RT

Faire des objets conformes avec API included et les gérer avec un modèle décrit à coté (plus d’introspection)

Permissif, et accompagné d’un modèle, on peut gérer les composants, commencer à faire des verifications, mais on ne peut pas garantir la syncronisation forte modèle/code

## EnTiMid component model

Faire des objets conformes API excluded, managed by model containing APIs

Le code des composants contient peu voire pas d’information sémantique.

Le modèle du composant lui est ajoint par un set d’adjoint par un set d’annotation Java dans notre implementation, mais on pourrait très bien imaginer rentrer tout ca dans un fichier XML à coté.

On peut donc connecter ce qu’on veut au rutime. On peut faire des check au niveau du modèle, et on peut garantir la forte synchronisation du modèle et du code.

# Architecture Synthesis

## Static: FD, CT, CI

## Dynamic; MAD Editor, BEPL

BPEL distribué. Pas encore adressé mais résolu par nous….

## Synthétiseur, Compilateur

# Validation

Tout au long de la realisation de ces travaux, des validations tant au niveau technique qu’humain ont été réalisées. Il est essentiel, dans le contexte du MAD, que les solutions avancées et proposes par cette these répondent aux besoins du domaine, et soient acceptées par les multiples utilisateurs autour de ce système. Cette section fait echo des divers retours d’expériences et validations menées sur la solution proposée.

## A travers différents points de vue

* Dévelopeur
* Technicien/Installateur
* Utilisateur final

## IDA

\cite{ASSAD:2010}

## ITI Project

\cite{RIALLE:2007} \cite{COLAS:2009}

## Industrial contacts

## Public Demonstrations

### FDLS

### Inauguration ESIR

### AERES

## Spin Off

# Conclusion

A partir d’une etude précise des besoins dans le contexte societal fort du maintien à domicile de personne dependant, et d’un socle de connaissances en domotique et IDM, la these présentée dans ce document offre une solution tirant partie des bonnes pratiques d’approches existantes, et combles les manques, afin de répondre au plus juste aux problématiques du domaine.

## Synthesis board

Reprise du tableau, 1 ligne de plus.

## IoT-IoS

Seamless integration of IoT and IoS

# Perspectives

Au dela du travail realise, il reste aujourd’hui de nombreauses pistes à explorer autour de ces travaux. Cette section donne un apercu des possibles suites à donner dans ce domaine.

## Open Control/Command operating system ^^

## EUP

Make it possible for everybody to understand and modify software behaviour

## FuzyLogic

Base behaviour reflexions on non-fixed value such as “high” or “low”. This could incite people to use the system and appropriate the configuration.

## Reste à faire

# Bibliography

* + Using MDE to build a schisophrenic middleware

# Glossary

DPWS : Device Profile for Web Services

EPC : Electronic Product Code™

EUP : End User Programming

ICT : Information and Communication Technology

IOS : Internet Of Services

IOT : Internet Of Things

MDE : Model Driven Engineering

RFID : Radio Frequency IDentification

UPnP : Universal Plug & Play

MAD

HAD

# Definitions

Plateforme : Environnement logiciel et materiel pleinement spécifié dans lequel s’exécute une application.

Plateforme d’exécution : Evironnement materiel pure prévu pour accueillir des applications.

Environnement d’exécution : Support d’exécution logiciel pour une apllication finale.