Étude / Analyse du système RUNES au travers d'un cas concret d'utilisation

Introduction

- Description générale du projet RUNES :
    - Buts et Objectifs
    - Architecture générale

1. Exemples de cas concrets d'utilisations (scénarios)

1.1. Healthcare

1.1.1. Expliquer l'importance de ce domaine

1.1.2. Exemple sur le problème cardiaque

1.1.2.1. Symptômes

1.1.2.2. Quels capteurs et infrastructure

1.2. Automative safety and security

1.2.1. Expliquer l'hétérogénéité des voitures et le besoin de factoriser (grâce à RUNES)

1.2.2. Conduite assistée

1.2.2.1. Capteurs

1.2.2.2. Centre d'analyse dans la voiture

1.2.2.3. Communication entre voitures pour relais d'information si accidents, dégradation soudaine de la route

1.2.2.4. Exemple avec le consortium CAR 2 CAR

1.2.3. Régulation de trafic

1.2.3.1. Détection des voitures

1.3. Factory automation

1.3.1. L'importance de produire plus à moindre coût

1.3.2. RUNES permettrait de gérer des capteurs pour réguler et analyser la chaîne logistique de la production à la distribution pour améliorer les processus moins performants

1.3.3. Exemple : la production de vin

1.3.3.1. Marché énorme de 150 milliards d'€

1.3.3.2. RUNES permettrait de surveiller la chaîne de production pour garantir un vin de bonne qualité

1.3.3.3. Prise en charge du réseau de distribution

1.3.3.4. Comparaison avec Eurojenet Wine qui ne s'occupe que de la production

1.4. In-home safety and security

1.4.1. Marché important de la sécurité contre voleurs et sécurité pour personnes âgées

1.4.2. Comparaison avec le projet GERHOME du CSTB

Phrase de transition vers l'exemple du tunnel, un scénario principal pour RUNES réunissant certains cas concrets mentionnés

2. Principes de fonctionnement (basé sur un des exemples précédents)

(Études des possibilités et restrictions d'un tel système au travers d'un cas concret d'utilisation)

2.1. Adaptation à la topologie du / des réseaux

2.1.1. Architecture du réseau

2.1.1.1. Expliquer l'abstraction mise en place pour l'accès aux capteurs

2.1.1.1.1. Réseau en 3 couches, rôle, utilité

2.1.1.1.1.1. Rouge, Routeur pur

2.1.1.1.1.2. Jaune, Sous-routeur pour capteur, capteur

2.1.1.1.1.3. Capteur pur

2.1.1.1.2. Type de communication privilégiée

2.1.1.1.2.1. Filaire le plus possible

2.1.1.1.2.2. RF le cas échéant

2.1.1.1.2.3. Réseau ad-hoc

2.1.2. Réaction du réseau à un mouvement

2.1.2.1. Déplacement d'un sous système

2.1.2.1.1. Réaction du réseau

2.1.2.1.1.1. Transmettre les infos du déplacement

2.1.2.1.2. Réaction du sous système

2.1.2.1.2.1. Reconfigurer l'accès aux données voulues

2.1.2.1.3. Réaction du point de vue applicatif

2.1.2.1.3.1. Presque transparent, pas de configuration manuelle

2.1.2.2. Cassure sur le réseau

2.1.2.2.1. Réaction du réseau

2.1.2.2.1.1. Rétablissement de la connexion tel 'un mouvement

2.1.2.2.1.2. Communication de la cassure

2.1.2.2.2. Réaction des applications

2.1.2.2.2.1. Signalisation de la cassure

2.1.2.2.2.2. Ne doit pas empêcher l'utilisation du système

2.1.3. Exemple d'organisation

2.1.3.1. Dessin de l'exemple du tunnel

2.1.3.2. Accès aux données et filtrages

2.1.3.2.1. Quelles données transmettre vers où ?

2.1.3.2.2. Que doit on filtrer, pourquoi ?

2.1.4. Problèmes posés sur le réseau

2.1.4.1. Sécurité et accès aux données

2.1.4.1.1. Chemin d'accès à un capteur donné

2.1.4.1.2. Politique d'accès aux donnés transitant sur un réseau

2.1.4.1.2.1. Toutes les données sont actuellement libres

2.1.4.1.2.2. Pose des problèmes sur certaines données sensibles, peuvent être falsifiés

2.1.4.2. Localisation des sous systèmes

2.1.4.2.1. Systèmes fixes

2.1.4.2.1.1. Résolution par stockage sur le capteur

2.1.4.2.1.2. Pose le problème de configuration manuelle

2.1.4.2.2. Systèmes mobiles

2.1.4.2.2.1. Résolution par repérage par rapport aux systèmes fixes

2.1.4.3. Saturation du réseau

2.1.4.3.1. Politique de filtrage des informations importantes

2.1.4.3.1.1. Trop d'info peut tuer l'info

2.1.4.3.1.2. Trop d'info retarde l'info

2.1.4.3.1.3. Priorité des infos

2.1.4.3.2. Prévention de l'accès au réseau

2.1.4.3.2.1. Exemple de la voiture qui va changer de réseau

2.1.4.3.2.2. Limitation de la reconfiguration, en amont (protocoles lourd)

Phrase de transition vers l'étude de l'architecture du middleware permettant de remplir les fonctionnalités précédemment mentionnées.

2.2. Modèle de composants

2.2.1. Présentation du modèle

2.2.1.1. Description des caractéristiques physique d'un composant (+ 1 schéma)

2.2.1.1.1. Interface

2.2.1.1.2. Réceptacle

2.2.1.1.3. Capacité de composition

2.2.2. Connexion entre composant. (+ 1 schéma).

2.2.2.1. Composant Connector

2.2.2.2. Capacités d'introspection

2.2.2.2.1. Ajout d'intercepteur

2.2.3. Capacité d'introspection et de réflexivité

2.2.3.1. Composant Capsule

2.2.3.2. Capacités d'introspection

2.2.3.2.1. Cycle de vie d'un composant

2.2.4. Comparaison avec un modèle de composant existant

2.2.4.1. Présentation du modèle de composant Fractal (+ 1 schéma)

2.2.4.1.1. Interface cliente et serveur

2.2.4.1.2. Membrane, contrôleurs et intercepteurs

2.2.4.2. Comparaison avec le modèle de composant Runes

2.2.4.2.1. Modélisation de l'introspection

2.2.4.2.2. Modélisation de la Réflexivité

2.2.4.2.3. Critique des avantages de conceptions des deux modèles

Phrase de transitions vers les services orientés composants permettant la réalisation des fonctionnalités requise dans l'exemple

2.3. Services associés à la couche middleware (middleware services) mis en œuvre dans ce cas d'utilisation

2.3.1. Liens entres les services et la couche inférieure

2.3.1.1. Développés en assemblant et regroupant des composants

2.3.1.2. Notion de "component framework" (si pas explicitement présent dans partie précédente)

2.3.1.3. Chargement durant l'exécution

2.3.2. But de ces services

2.3.2.1. Poser les bases nécessaires au fonctionnement ainsi qu'au développement d'application distribuées et/ou embarquées

2.3.2.2. Chargement des services en fonction des besoins pour modifier / étendre les comportements du système

2.3.3. Détails des principaux services développés par l'équipe du projet RUNES et mis en œuvre dans cet exemple

2.3.3.1. Interaction service

2.3.3.1.1. Transmission aux applications de toutes les communications provenant d'autres CFs ou services dans le but de permettre leur interaction

2.3.3.1.2. Possibilité d'ajout de plugin (PIP) pour étendre les possibilités d'interaction

2.3.3.2. Overlay Service

2.3.3.2.1. Assurant la couverture du réseau par le middleware

2.3.3.2.2. Permet de s'abstraire de tout type de support physique via les "plugin overlay networks"

2.3.3.3. Advertising and Discovery Service

2.3.3.3.1. Permet à tout CF venant d'être chargé d'informer l'ensemble du système de son arrivée (advertising)

2.3.3.3.2. Permet à tout CF préalablement chargé de chercher un autre CF disponible depuis peu (discovery)

2.3.3.4. Logical Mobility Service

2.3.3.4.1. Permet la modification dynamique de la configuration du système en fonction des données collectées

2.3.3.4.2. Envoi de Logical Mobility Unit (LMU) aux entités

2.3.4. Remarques sur le développement et l'utilisation de ces services

2.3.4.1. Relative facilité de développement de nouveaux services grâce aux API et plugin proposés

2.3.4.2. Forte adaptabilité / hétérogénéité du système

2.3.4.3. Complexité croissante avec la taille du système

2.3.4.4. Fonctionnement dynamique et transparent la plupart du temps

Conclusion

- Bilan des possibilités et restrictions de ce système dans un tel cas d'utilisation