ONTOLOGY FOR SOFTWARE CONFIGURATION MANAGEMENT
A Knowledge Management Framework for Software Configuration Management

Nikiforos Ploskas
TELETEL SA, 124, Kifissias Avenue, 115 26 Athens, Greece

Michael Berger, Jiang Zhang, Lars Dittmann
COM·DTU, Technical University of Denmark, Oersteds Plads Bldg 343, DK-2800, Lyngby, Denmark

Gert-Joachim Wintterle
Alcatel-Lucent AG, Lorenzstr. 10, 70435 Stuttgart, Germany

Keywords: Knowledge Management, SCM, Ontology, OWL, Reasoner.

Abstract: This paper describes a Knowledge Management Framework for Software Configuration Management, which will enable efficient engineering, deployment, and run-time management of reconfigurable ambient intelligent services. Software Configuration Management (SCM) procedures are commonly initiated by device agents located in the users gateways. The Knowledge Management Framework makes use of Ontologies to represent knowledge required to perform SCM and to perform knowledge inference based on Description Logic reasoning. The work has been carried out within the European project COMANCHE that will utilize ontology models to support SCM. The COMANCHE ontology has been developed to provide a standard data model for the information that relates to SCM, and determine (infer) which SW Services need to be installed on the devices of users.

1 INTRODUCTION

The main objective of COMANCHE (COMANCHE, 2008) is to develop and validate a generic framework for Software Configuration Management (SCM), which will pave the way to the realization of technically and commercially viable private spaces incorporating ambient intelligence features. For this purpose the project will specify and develop the COMANCHE modular and scalable architecture targeting the provision of consistent, secure, low-cost (low-effort) SCM services across today's heterogeneous, and multi-vendor environments. The realization of the COMANCHE SCM services framework will be built on an adequate software engineering and knowledge management infrastructure that the project will deliver. The main components of this infrastructure will be the following:

i) The COMANCHE Knowledge Management Framework, which will provide the means for effectively conceptualising, organizing, discovering, and exploiting the tremendous amounts of (currently unorganised and scattered) attribute information, pertaining to software configuration management.

ii) A modular component-based software architecture and an adequate design methodology and tool, which will effectively address the engineering and run-time management of reconfigurable software for ambient intelligent networked services environments.

iii) A formal modelling methodology and a consistency validation framework for capturing and analysing the structure and run-time behaviour of distributed software systems. This approach will aim to preserve the integrity of the target networked services environments in terms of allowing software-configuration error detection and recovery across present complex, and multi-vendor private spaces.

This paper will mainly focus on i) The COMANCHE Knowledge Management Framework and furthermore it will present an innovative, concrete methodology for performing SCM by means of logical inference in the developed COMANCHE ontology. The idea of using ontologies for SCM is not new (Arantes, 2007), (Asikainen, 2004), (Shahri, 2007), but this paper proposes a novel procedure for SW configuration using available of-the-shelf ontology inference engines (reasoners).

The organization of the paper is as follows: In the following section 2, a small example is provided to illustrate the principles of knowledge inference for software configuration management. The simple examples provided in this section illustrate some of the principles of the COMANCHE ontology presented later. Following this, section 3 presents the SCM provider Infrastructure that is responsible for the coordination of SCM procedures and derivation of SCM decisions. Section 4 describes the Knowledge management framework which is responsible for maintaining the COMANCE ontology described in section 5. Finally, section 6 is the conclusion.

2 KNOWLEDGE INFERENCE

A small example is provided here to illustrate the principles of knowledge inference for software configuration management. It is assumed that a VCR device is going to be installed in a home environment already equipped with a TV set and Set Top Box (STB). The objective of the SCM procedure is then to determine an appropriate software configuration for the new VCR device. The first step is to determine candidate SW services that fulfil the needs of the new device. Available SW services are described by a service Ontology. Figure 1 shows an example Ontology for a SW service denoted VCR_Service_1.

The behaviour of the service is described by the VCR_Service1_profile class. This class basically defines the properties of the service in terms of functionality, e.g. inputs and results. Furthermore, the ontology specifies dependencies by the *dependsOn* property and the *supportsDevices* property indicates the hardware platforms compatible with the service. The *typeOfService* property specifies that VCR_Service_1 is a VCRService and the property *isDeliveredBy* gives the vendor of the SW service.

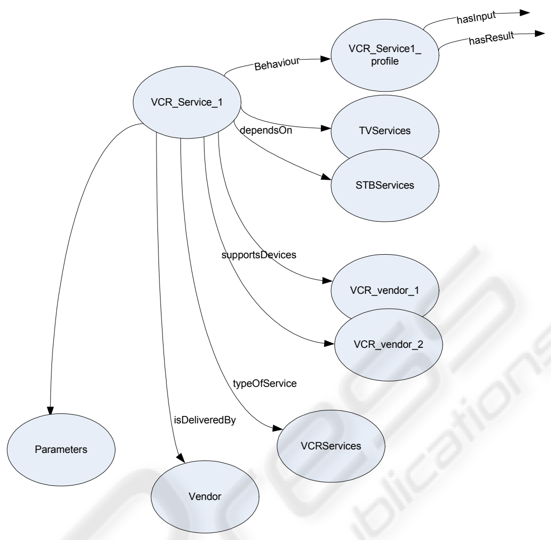

Figure 1: Ontology for a specific VCR service: VCR_service_1.

The behaviour of the service is described by the VCR_Service1_profile class. This class basically defines the properties of the service in terms of functionality, e.g. inputs and results. Furthermore, the ontology specifies dependencies by the *dependsOn* property and the *supportsDevices* property indicates the hardware platforms compatible with the service. The *typeOfService* property specifies that VCR_Service_1 is a VCRService and the property *isDeliveredBy* gives the vendor of the SW service.

In this example the Protégé OWL [5] Ontology editor is used to manage the ontologies. It is assumed that a new VCR SW service is going to be installed. The VCR is delivered by Company_X and it is a requirement that the service supports pause of live TV transmissions. To obtain a list of available candidate services, a new class VCRAvailableServices is defined with an equivalent class in Protégé syntax as shown in Figure 2.

```
Service
and supportsDevices some VCR_COMPANY_X
and behaviour some (ServiceProfile
              and hasResults some Pause_live_TV)
and typeOfService some VCRServices
```

Figure 2: Equivalent class definition for VCRAvaliableServices.

The goal is to determine available candidate services that fulfil the equivalent class definition above. This is obtained by inference performed by a Reasoner, in this case FaCT++ [6]. The reasoner derives an inferred class hierarchy for the Ontology, and in this case, the reasoner infers that VCR_Service_1 and VCR_Service_2 are subclasses of VCRAvaliable services, see Figure 3.

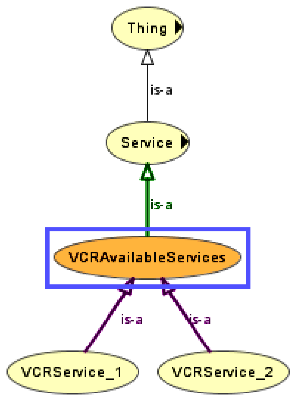

Figure 3: Inferred class hierarchy.

Based on the inferred class hierarchy it is concluded that the two SW services VCR_Service_1 and VCR_Service_2 fulfils the requirements to be installed on the actual VCR device.

Assuming that VCR_service_1 is selected, the next step is to determine any services that need to be installed on dependent devices i.e. the TV and STB. In this example it is assumed that the STB is provided by Company_Y and the TV is provided by Company_Z. In order to determine dependent services for the STB and TV, a new class is defined in the Ontology. The class name is VCR_Service_1_dependencies and the equivalent class definition is shown in Figure 4. Note that an invDependsOn property has been defined as the inverse of the DependsOn property.

```
Service and
(supportsDevices some STB_COMPANY_Y) or
(supportsDevices some TV_COMPANY_Z)
and typeOfService some (ServiceType
and invDependsOn some VCRService_1)
```

Figure 4: Equivalent class definition for VCR_Service_1_dependencies.

Again, the reasoner is activated to infer classes representing services that must be installed on the dependent devices. The inferred class hierarchy that the reasoner calculates is shown in Figure 5. Based on the inferred class hierarchy it is concluded that the SW services STBService_1, STBService_3, TVService_1 and TVService_2 should be installed on STB_COMPANY_Y and TV_COMPANY_Z respectively. These inferred services may have been installed already or they may have further dependencies that have to be resolved by similar procedures. The following section will provide details on the framework in which the inference takes place.

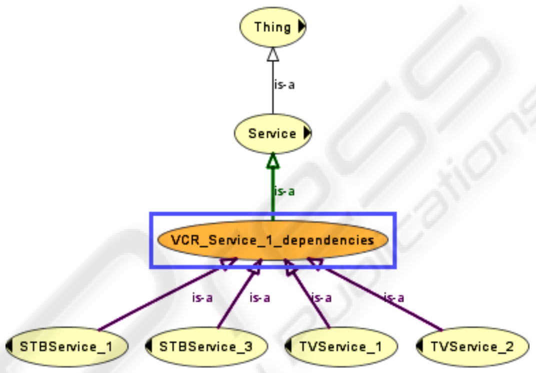

Figure 5: Inferred Class hierarchy.

3 SCM PROVIDER INFRASTRUCTURE

The SCM Engine is the heart of the COMANCHE architecture. The SCM Engine is responsible for the coordination of SCM procedures and the derivation of SCM decisions.

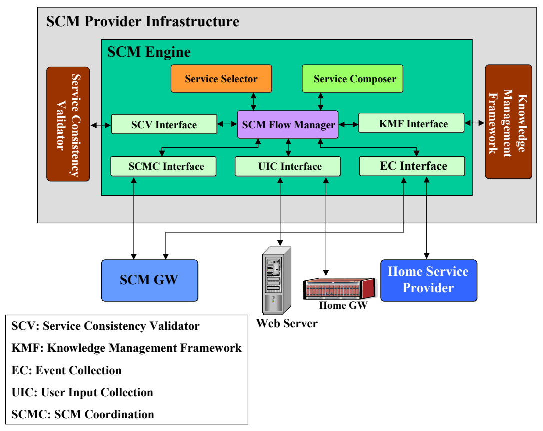

Figure 6: Functional architecture and interfaces of SCM Engine.

An SCM procedure is treated as what is commonly known as a service composition procedure. The intelligence of the SCM Engine will largely depend on the use of the COMANCHE ontology. In this context, the SCM Engine will select service components (i.e. SW Services and Internet Services) and will synthesise Composite Service Descriptions (CSDs). A Composite Service Description (CSD) is the specification of the software configuration of a Home Environment. It specifies which SW Services (SW Components) will need to be installed on which devices of the Home Environment, and how these services will need to be configured on each device. SCM procedures are typically triggered by device agents on the SCM GW. Configuration is done through the Device Agent entity that realises a web service based interface towards the SCM engine.

The main functional purpose of the **Service Selector** is the matching of SCM needs to Service Profiles. SCM requirements will be represented by the instantiations of the Home Environment and Context Ontology (integrated in the COMANCHE ontology). Service Profiles will be represented by the instantiations of the Service Profile Ontology (also integrated in the COMANCHE ontology).

The main functional purpose of the **Service Composer** is the synthesis of Composite Service Descriptions (CSDs). A successful, executable composition correctly combines and configures a set of compatible components to achieve the composition's overall goal. Full automation of composition is still the object of ongoing, highly speculative research with little short-term hope (Kim, 2004). However, partial automation of composition, with a human controller as the most significant decision mechanism, seems a feasible goal. The Service Composer will adopt a **user interactive approach** for service composition and configuration. Relatively simple, fully automated SCM cases will only be considered for a limited set of remote diagnostics and repair applications. The Service Composer functionality will be primarily based on the use of the Service Process Model. User input as well as instantiations of the Home Environment and Context Ontology and User and Business Domain Ontology will be queried and exploited for this purpose.

The **SCV (Service Consistency Validator) Interface** will be used for interacting with the Service Consistency Validator of the SCM Provider Infrastructure. The SCV Interface will be internal to the SCM Provider infrastructure. The SCM Engine will make use of this interface in order to request the execution of off-line consistency validation tests with regard to a formulated CSD. The output of these tests will be communicated by the Consistency Validator to the SCM Engine.

The **UIC (User Input Collection) Interface** will serve interactions with users. The UIC Interface will be external to the SCM Provider infrastructure. It will be based on the use of Web Services (WSDL, SOAP). The UIC Interface will allow the interaction of the SCM Engine with the back-end of a Web Server or a Home Gateway or any other network element exposing a user interface.

The **EC (Event Collection) Interface** will allow the collection of SCM-related events. These events will be collected from Home Service Providers (e.g. a device fault event reported by a remote diagnostics service provider, a request for a service upgrade, etc), home gateways (e.g. a malfunction detection event), and SCM Gateways (Service Execution Errors). Apart from the interface with the SCM Gateways allowing the notification of Service Execution Errors, all other interfaces with Home Service Providers, and Home Gateways will be considered as proprietary and out of the scope of the present specification.

The **KMF Interface** is the Knowledge Management Framework interface. All requests that need to be done to the KMF will be carried out by this interface. This includes fetching ontology instantiations and the update of the Knowledge Base according to events received by the SCM Engine's EC Interface.

The **SCM Flow Manager** is responsible for the coordination of the operations performed by the SCM Engine. All messages and flows are controlled by this component, so that the procedures the SCM Engine is responsible for can be carried out successfully.

4 KNOWLEDGE MANAGEMENT FRAMEWORK

The Knowledge Management Framework (KMF) is responsible for formulating and maintaining valid and consistent instantiations of the COMANCHE ontology. These instantiations are created through discovering and collecting/importing knowledge from Attribute Providers and Context Providers as depicted in Figure 7. The KMF is composed of a number of components that are described in detail bellow:

Attribute Providers. The Attribute Providers publish and supply persistent knowledge instantiations relating to services, user profiles, and business domain relations. This knowledge instantiations are structured (by the Attribute Providers) using the Service Ontology and User Profile and Business Domain Ontology. Attribute Providers can be SW Providers publishing knowledge about their SW, Service Providers publishing knowledge about their Services, Device Manufacturers publishing knowledge about their Devices, etc

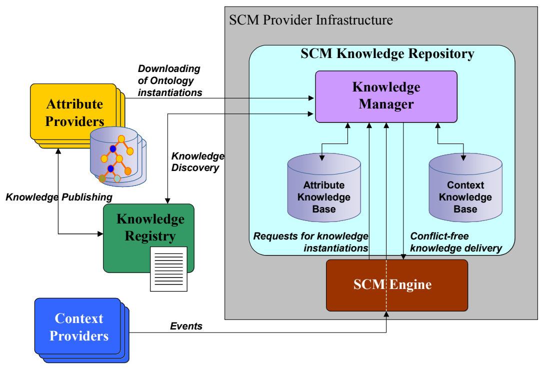

Figure 7: Functional architecture of the COMANCHE Knowledge Management Framework.

Context Providers. The Context Providers supply dynamic context information regarding Home Environments. This information is communicated in the form of appropriately structured event messages and mapped on the Home Environment and Context Ontology. Within the COMANCHE framework, Context Providers are the SCM Gateways, home gateways, and Service Providers (e.g. a Remote Diagnostics & Maintenance Provider may report the event of a device fault).

Knowledge Manager. The **Knowledge Manager** is a key component to the implementation of the SCM Knowledge Repository. It is responsible for the following main tasks:

- Receive (by the SCM Engine) and process requests for the delivery of conflict-free knowledge instantiations.
- Discovery of knowledge instantiations across different Attribute Providers based on the aforementioned requests. For this purpose, the Knowledge Manager will need to formulate appropriate queries to the Knowledge Registry.
- Mapping of the information contained in event messages received by Context Providers onto the Home Environment and Context Ontology.
- Resolving of potential conflicts identified in the knowledge instantiations discovered across different Attribute Providers and Resolving of potential conflicts identified in the information received by Context Providers. The trustworthiness of the different Attribute Providers will be the key criterion for achieving this. In addition, the Knowledge Manager will need to be capable of inference. An off-the-self inference engine (reasoner) will be used for this purpose (e.g. FaCT++ Reasoner).
- Importing of conflict-free knowledge instantiations in the Attribute and Context Knowledge Bases.
- Delivery of consistent knowledge instantiations to the SCM Engine in order to be exploited for SCM.

Attribute and Context Knowledge Base. The Attribute Knowledge Base consists of the Service Ontology, User Profile and Business Domain Ontology, the parts of the Home Environment and Context ontology that contain static information on software and devices, as well as their conflict-free instantiations.

The Context Knowledge Base consists of the dynamic part of the Home Environment and Context Ontology, as well as its conflict-free instantiations.

Knowledge Registry. This component enables the discovery of the different Attribute Providers by the SCM Services Providers. Attribute providers use the registry interface for publishing attribute knowledge. Semantically enhanced UDDI (Universal Description, Discovery, and Integration) services are employed for this purpose as it is detailed in the following section:

5 THE COMANCHE ONTOLOGY

The purpose of the ontology is to provide a standard data model for the information that relates to SCM, and determine (infer) which SW Services need to be installed on the devices of users. The inference is based on a number of selection principles as described in the following:

Selection of a SW Service depending on a Desired Device Function. The actual device in question is defined as an individual, e.g. the individual Bob_Washing_Machine belongs to the class Bob_devices. The desired device functions are described by the needsToSupport property as object properties of the device, e.g needsToSupport Washing_Of_Silk_Clothes and needsToSupport Washing_Of_Colored_Clothes, where Washing_Of_Silk_Clothes and

Washing_Of_Colored_Clothes are also defined as individuals in the ontology.

Selection of a SW Service depending on Compatibility with a Device SW Platform. The SW service configuration will depend on the device vendor, i.e. the vendor of Bob_Washing_Machine. The actual device model is described by the hasDeviceModel property as object properties of the device, e.g. hasDeviceModel CompanyX_Washing_Machine_0001. Changing the device model would lead to new SW services to be installed on the device.

Selection of a SW Service depending on the Subscriptions of the Device Owner. The owner Bob of Bob_Washing_Machine is defined as an individual in the Ontology. A property holdsUserSubscription is used to describe which SW services that the user Bob is subscribed to. SW services that the user does not subscribe to would not be installed on the device.

Selection of a SW Service depending on whether the Software Provider is Trusted by the Device Manufacturer. As in the example above it is assumed that the device model of Bob_washing_Machine is CompanyX_Washing_Machine_0001. This is also an individual that will connect to the vendor, in this case CompanyX, through a hasManufactorer property. Trusted SW providers for CompanyX is declared by a hasTrustedThirdParty property.

Selection of SW Services for Gateway Devices. The target device of a SW_Service is the device that will make use of its functionality. The previous examples referred to SW Services that are installed on the target device (i.e. Bob_Washing_Machine). It may be the case that a SW Service is not installed on the target device but on a different device (e.g. a Home Gateway or a Control Point) that connects to it. This type of devices that host SW Services to be used by other devices are termed as Gateways in the ontology.

Selection of SW Services in Accordance with known Dependencies between SW Services. Dependencies between software services are described by the dependsOn property, e.g. SW_service_7 dependsOn SW_service_5. Dependencies could be organised in a dependency tree but due to the transitiveness of the dependsOn property, the tree can be automatically flattened by the inference procedure.

5.1 Equivalent Class Definition

Based on the SW service selection principles above, the main achievement has been the equivalent class definition in the Ontology that allows the reasoner to infer SW service configuration, see Figure 8.

```
(toBeInstalledOnDeviceType some
(inverse_of_isOfDeviceType value
Bob_Washing_Machine))
or (inverse_of_dependsOn some
(toBeInstalledOnDeviceType some
(inverse_of_isOfDeviceType value
Bob_Washing_Machine)))
and (inverse_of_dependsOn some
(inverse_of_containsService some
(inverse_of_holdsUserSubscription some (owns value
Bob_Washing_Machine))))
    or (inverse_of_containsService some
(inverse_of_holdsUserSubscription some (owns value
Bob_Washing_Machine)))
and (inverse_of_dependsOn some (compatibleWith
some (inverse_of_hasSWPlatform some
(inverse_of_hasDeviceModel value
Bob_Washing_Machine))))
    or (compatibleWith some
(inverse_of_hasSWPlatform some
(inverse_of_hasDeviceModel value
Bob_Washing_Machine)))
and (hasTargetDeviceModel some
(inverse_of_hasDeviceModel value
Bob_Washing_Machine))
    or (inverse_of_dependsOn some
(hasTargetDeviceModel some
(inverse_of_hasDeviceModel value
Bob_Washing_Machine)))
and (hasServiceProvider some
(inverse_of_hasTrustedThirdParty some
(inverse_of_hasManufacturer some
(inverse_of_hasDeviceModel value
Bob_Washing_Machine))))
    or (inverse_of_dependsOn some
(hasServiceProvider some
(inverse_of_hasTrustedThirdParty some
(inverse_of_hasManufacturer some
(inverse_of_hasDeviceModel value
Bob_Washing_Machine)))))
and (inverse_of_dependsOn some
(supportsDeviceFunction some
(inverse_of_needsToSupport value
Bob_Washing_Machine)))
    or (supportsDeviceFunction some
(inverse_of_needsToSupport value
Bob_Washing_Machine)
```

Figure 8: Equivalent class definition for Bob_washing_machine.

The equivalent class definition might look rather complicated, but it also has to include all the above selection principles. On the other hand the definition is rather generic and provided that the devices are of same type, they only differ in the name of the specific device. By invoking the reasoner (e.g. FaCT++), the required SW configuration will be obtained, i.e. a list of all SW services to be installed on the device.

5.2 Demo Architecture

A shown in the following Figure 9, the functional entities under development will be hosted on three PCs. These PCs will be connected to a LAN.
The functional entities to be developed are the following:

- Knowledge Manager (KM) that will quire and update the SCM Knowledge Base.
- Identity Provider (IDP) that will quire the Policies Repository.
- Knowledge Registry (KR).
- Attribute Provider (AP) that will quire the Attribute Base.

The SCM Knowledge Base, Policy Repository, and Attribute Base will be different versions (instances) of the COMANCHE ontology.

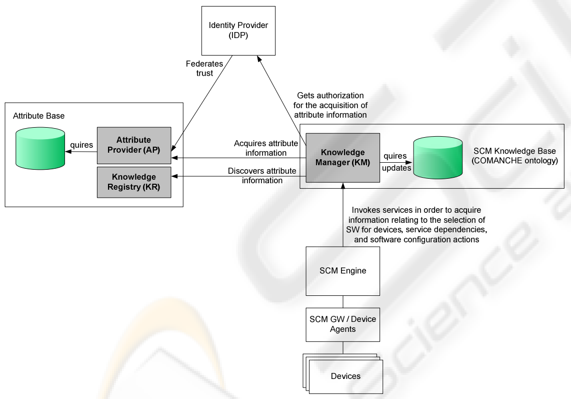
Identity Provider (IDP)
Federates trust
Gets authorization for the acquisition of attribute information
Attribute Base
quires
Attribute Provider (AP)
Knowledge Registry (KR)
Acquires attribute information
Discovers attribute information
Knowledge Manager (KM)
quires updates
SCM Knowledge Base (COMANCHE ontology)
Invokes services in order to acquire information relating to the selection of SW for devices, service dependencies, and software configuration actions
SCM Engine
SCM GW / Device Agents
Devices

Figure 9: KMF demonstration architecture and placement in the COMANCHE framework - KMF entities are represented by the shaded boxes.

5.3 Functional Roles of the Architecture Entities

Knowledge Manager. The Knowledge Manager performs the following main tasks:

- Responds to the queries of the SCM Engine and provides information relating to the selection of SW for devices, service dependencies, and software configuration actions. For this purpose the KM accesses and process data (attribute information) that resides in the SCM Knowledge Base. The data contained in the SCM Knowledge Base is modelled through the use of the COMANCHE Ontology.
- In case that the data (attribute information) contained in the SCM Knowledge Base is not sufficient (i.e. some data fields in the SCM Knowledge Base are empty), and due to this fact the KM is not able to respond to a query from the SCM Engine, then it will discover the URI of the missing data by querying the Knowledge Registry, and then acquire the data by querying the Attribute Provider. The Identity Provider (IDP) entity appropriately authorises these actions towards ensuring that user privacy preferences and policies are respected.

Knowledge Registry. The Knowledge Registry (KR) provides to the Knowledge Manager (KM) the URIs of data elements (attribute information) that need to be acquired by the latter. These data elements will reside at Attribute Providers (AP). So the KR in fact provides each time to the KM the URI of the AP that holds the data in question.

Identity Provider. The Identity Provider (IDP) will authorize access to attribute information. For this purpose, the IDP will issue authorization tokens to the KM. The Policy Repository at the IDP will contain policy information that will serve as input for determining whether access to data should be granted or not.

Attribute Provider. The Attribute Provider will supply to the KM the requested attribute information, provided that the KM has been authorized for this by the IDP. The attribute information in question will be contained in the Attribute Base at the AP.

5.4 Demonstration Scenario

A simple indicative scenario is presented in the sequel.

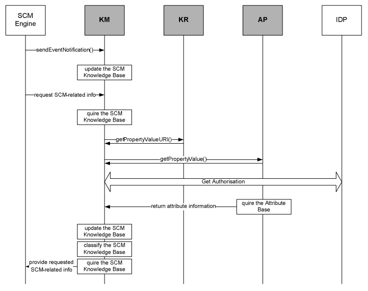
SCM Engine
KM
KR
AP
IDP
sendEventNotification()
update the SCM Knowledge Base
request SCM-related info
quire the SCM Knowledge Base
getPropertyValueURI()
getPropertyValue()
Get Authorisation
return attribute information
quire the Attribute Base
update the SCM Knowledge Base
classify the SCM Knowledge Base
provide requested SCM-related info
quire the SCM Knowledge Base

Figure 10: KMF representative scenario.

The scenario involves the following steps:

1. An event occurs in the home network: e.g. a new device is connected or a fault occurs on a device. A message is passed from the device to the SCM GW and then to the SCM Engine. The event is communicated by the SCM Engine to the Knowledge Manager using the sendEventNotification() Web service. The ontology is updated accordingly: e.g. the newly connected device is included in the ontology or the error is logged.

2. The previous event had as a consequence the triggering of an SCM procedure. The SCM Engine starts to issue requests for SCM-related info to the Knowledge Manager. The requested information may concern service dependencies, SW selection for devices, configuration actions, etc.

3. The Knowledge Manager quires the SCM Knowledge Base and finds out that a previous request by the SCM Engine cannot be served due to information missing.

4. The Knowledge Manager uses the getPropertyValueURI() Web service to request from the Knowledge Registry the URI of the missing attribute information. The Knowledge Registry replies by providing the URI (or URL of the Attribute Provider).

5. The Knowledge Manager uses the getPropertyValue() Web service to request from the Attribute Provider the missing information. The Attribute Provider replies that authorisation from the Identity Provider is required.

6. Authentication and authorisation procedures using the services of the Identity Provider take place.

7. Once authorised, the Attribute Provider quires the Attribute Base and supplies the requested information to the Knowledge Manager.

8. The Knowledge Manager updates the SCM Knowledge Base with the above attribute information.

9. The logic (property restrictions and rules) contained in the ontology are put in force . The ontology is classified and the inferred types of the concept instances are computed.

10. The Knowledge Manager quires the inferred knowledge base and supplies to the SCM Engine the requested information.

6 CONCLUSIONS

Based on the COMANCHE specification of a Knowledge Management framework for Software Configuration Management, the COMANCHE ontology has been designed to facilitate organisation of attribute information related to SCM, and to facilitate automated configuration specifications based on a novel procedure for knowledge inference using available of-the-shelf ontology inference engines (reasoners). The Knowledge Management Framework (KMF) provides the means for managing the Ontology through external attribute providers and derives knowledge to the SCM engine for the calculation of Composite Service Descriptions.

ACKNOWLEDGEMENTS

This work has been performed in the framework of the IST-034909 project COMANCHE, which is partly funded by the European Commission. The authors would like to acknowledge the contribution of their colleagues from Gorenje, Intracom, Indesit Company, Alcatel-SEL, Solinet, APEX, CERFRIEL, Technical University of Denmark – COM, Teletel and RWTH Aachen University.

REFERENCES

The COMANCHE Project, www.ist-comanche.eu, 2008

L. Arantes, R. Falbo and G. Guizzardi. Evolving a Software Configuration Management Ontology. *WOMSDE 2007*, Brazil.

T. Asikainen, T. Männistö, and T. Soininen. Representing Feature Models of Software Product Families Using a Configuration Ontology. *Presented at the ECAI 2004 Configuration Workshop,* Valencia, Spain

H. Shahri, J. Hendler, A. Porter. Software Configuration Management Using Ontologies. *SWESE 2007*, Austria.

http://protege.stanford.edu/plugins/owl/, 2008

FaCT++, owl.man.ac.uk/factplusplus/, 2008.

J. Kim, Y. Gil, and M. Spraragen. A Knowledge-Based Approach to Interactive Workflow Composition. *Proc. Int'l Conf. Automated Planning and Scheduling, Workshop Planning and Scheduling for Web and Grid Services*, AAAI Press, 2004.