

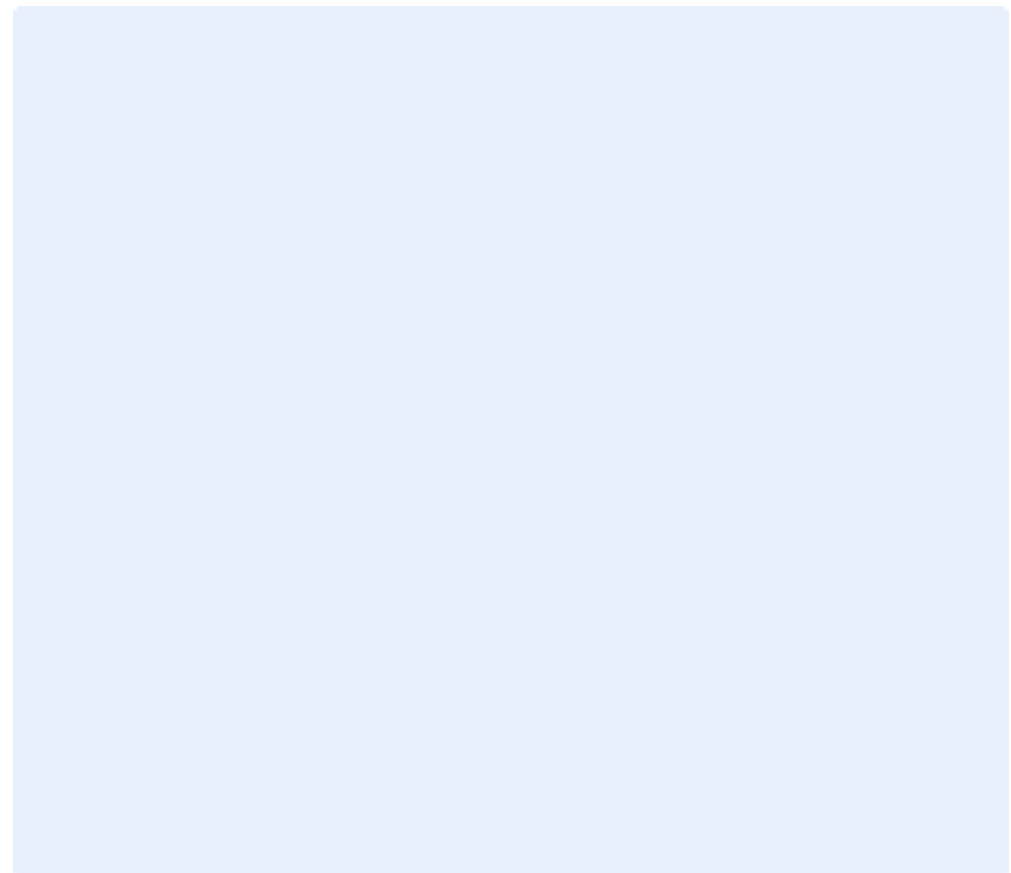
Report

FI – M2M D1.1 Platform architecture

Author(s)

Svein Olav Hallsteinsen

Babak Farshchian, Anna Burla, Shanshan Jiang, Bjørn Magnus Mathisen



SINTEF IKT
SINTEF ICT

Address:
Postboks 4760 Sluppen
NO-7465 Trondheim
NORWAY

Telephone: +47 73593000
Telefax: +47 73592977

postmottak.ikt@sintef.no
www.sintef.no
Enterprise /VAT No:
NO 948 007 029 MVA

Report

FI – M2M D1.1 Platform architecture

KEYWORDS:
Future Internet
Internet of Things
M2M
Self Adaptation
Auto Installation
Auto Evolution

VERSION
1.0

DATE
2011-12-09

AUTHOR(S)
Svein Olav Hallsteinsen,

Babak Farshchian, Anna Burla, Shanshan Jiang, Bjørn Magnus Mathisen

CLIENT(S)
Telenor

CLIENT'S REF.
FI-M2M D1.1

PROJECT NO.
90C329

NUMBER OF PAGES/APPENDICES:
52 + Appendices

ABSTRACT

FI-M2M Platform architecture

This document specifies the architecture of the FI – M2M development platform

for **PREPARED BY**
Svein Hallsteinsen

SIGNATURE



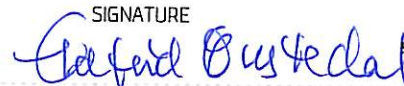
CHECKED BY
Babak Farschian

SIGNATURE



APPROVED BY
Eldfrid Øvstedal

SIGNATURE



REPORT NO.
SINTEF A21905

ISBN
978-82-14-05276-3

CLASSIFICATION
Unrestricted

CLASSIFICATION THIS PAGE
Unrestricted

Document history

VERSION	DATE	VERSION DESCRIPTION
V0.1	2011-05-11	Skeleton based on the Arcade template
V0.2	2011-05-30	Filled in initial content in selected sections
V0.5	2011-09-19	Comments from Babak and some changes to the scenario
V0.6	2011-12-05	More cleaning up of language and content.
V1.0	2011-12-09	Final version, changed author list to represent work.

Table of contents

1	Introduction	5
1.1	Motivation and background.....	5
1.2	Summary	6
1.3	Structure of this document	6
2	Context Viewpoint	7
2.1	AAL Domain.....	7
2.2	Scenarios	7
2.2.1	Adaptation.....	8
2.2.2	Another adaptation scenario:.....	9
2.2.3	Development	10
2.2.4	Evolution	10
2.3	Environment.....	11
2.4	Work processes	11
2.5	Business to system Mapping.....	11
2.5.1	Stakeholders	11
2.5.2	Reference use cases currently supported	14
3	Requirements Viewpoint	16
3.1	High level requirements	16
3.2	Target System Mapping Model	17
3.2.1	System categories	18
3.2.2	The behavioural aspects (service collaboration patterns)	19
3.2.3	Runtime support provider	20
3.2.4	Community support provider	27
3.2.5	Developer support provider	32
4	Component viewpoint.....	34
4.1	System information model.....	34
4.2	System Decomposition Model.....	36
4.3	Component and Interface Specification Model.....	37
4.4	System Collaboration Model.....	40
4.4.1	M2M platform collaboration model.....	40
5	Realisation viewpoint.....	43
5.1	System Deployment Model.....	43
5.2	Technology Mapping Model.....	46
5.2.1	Community support.....	46
5.2.2	Development support.....	46
5.2.3	Network simulator	47
5.2.4	Context and adaptation middleware.....	48
5.2.5	Service discovery.....	49
5.2.6	Communication layer.....	49
6	Conclusion	51

APPENDICES

Appendix A MUSIC "Description" (paper submitted to the Journal of Software and Services, special issue on the state of the art of adaptive systems.

Appendix B MUSIC Services (Book Chapter submitted to planned Springer book on Software Engineering for Self-Adaptive Systems)

Appendix C Network simulator (paper accepted by NIK 2011)

1 Introduction

1.1 Motivation and background

With more than a billion users, today's Internet forms an unparalleled information and communication network. The EU 7th Framework Programme is addressing issues concerning the Future Internet, where the fusion of The Internet of Things, Services, and Content and Media are central themes. The FI-M2M project mainly focuses on the first of these themes, The Internet of Things, and in particular the flexible integration of things into applications and services serving human needs. In this context, M2M (Machine-to Machine) systems and sensor networks play central roles. A key objective in most M2M research initiatives is to connect everyday objects (things) to form a ubiquitous network, so as to allow people to control them and use the data they provide to assist everyday tasks related both to work and recreation.

The vision is that systems in the Future Internet will be characterized by that they:

- consists of many parts which are things, applications, or services.
- leverage services developed and deployed by third parties.
- form semi-automatically, driven by the availability of parts and the situation, and guidance by humans defining high-level goals and making decisions.
- evolve dynamically over their lifetime.
- support mobility of both things and people.

The Future Internet and M2M service enablement project targets three main research objectives (RO). These can be summarized as follows:

- RO1: Provide architectures, middleware, tools, and methodologies that can act as an open common basis for M2M developers. Providing solutions that attract developers (professionals and non-professionals) is central for promoting growth in the M2M communication sector. To this end we will seek to produce architectures, tools, and methodologies that can contribute to simplify design, development, and composition of M2M oriented applications for mobile and ubiquitous environments. In particular, we will focus on providing better support for managing QoS in dynamically composed systems by means of dynamic adaptation. The project will explore the RESTful services paradigm and semantic description and discovery of services provided by embedded devices, and will seek to integrate relevant existing results from earlier projects in both Telenor and SINTEF.
- RO2: Identify easy and cost-effective means for deployment and evolution of M2M systems. The deployment of M2M solutions is complicated and expensive. The same is the case with evolution after initial deployment. This hampers the uptake of such solutions and there is the need to simplify and automate these tasks. This calls for technology, which support self-configuration at deployment time, and the ability to reconfigure automatically when parts are added or removed. Our approach will be to investigate the potential of smart wholesale application stores that exploits the capabilities of the platform provided by RO1 to simplify and partially automate deployment and evolution.
- RO3: Assess relevance and impact of M2M concepts in selected verticals and derive requirements for a common platform. In order to broaden the application space for M2M concepts, and promote acceptance and adoption of relevant tools and services, it is necessary to develop an understanding of how to accommodate such specific application areas. A central part of this project will be to tailor M2M concepts for selected verticals, and to conduct empirical evaluations to generate a better understanding of the added value and benefits.

The project is planned in two major steps. In the initial step we focus on the exploitation of parts with self-adaptation capabilities to ease the deployment and evolution of systems. In the next step we will focus on

special tools leveraging on the self-adaptation capabilities of software parts to further automate the deployment and evolution process.

This document is a preliminary architecture document meant to serve in the first step. It will be extended and matured in the 2nd step, to also reflect more completely the requirements related to more user driven and automated deployment and evolution and to become a final platform architecture document.

1.2 Summary

This document specifies the requirements and the initial architecture of the target tools and middleware platform. The document also elects a technology baseline for the realisation of the platform, building on existing technologies in Telenor and SINTEF, and will serve as a blueprint for the implementation. The design of the architecture has been elaborated following the architecture design method and description framework *Arcade*, which again is based on ANSI/IEEE 1471-2000, *Recommended Practice for Architecture Description of Software-Intensive Systems*. *Arcade* is open, while 1471-2000 is not (though you can find a description of it here: http://en.wikipedia.org/wiki/IEEE_1471). Information about *Arcade* can be found here: <http://www.arcade-framework.org/>.

In accordance with *Arcade* the document includes an analysis of stakeholders and scenarios from the AAL domain as a basis for the requirements. The design of the architecture has been shaped by these requirements, the constraints imposed by the baseline technologies and the wish to comply with the reference architecture for the AAL domain proposed by the UNIVERSAAL¹ project. UNIVERSAAL is a European FP7 integrated project addressing specifically software engineering of AAL systems running in parallel with the FI-M2M project.

1.3 Structure of this document

The structure of this document follows the guidelines of *Arcade* for the structure of architecture documents, and describes the system from different viewpoints in accordance with ANSI/IEEE 1471 2000.

Section 2, Context viewpoint, presents the AAL domain and identifies stakeholders and typical usage scenarios for AAL systems. Section 3, Requirements viewpoint, defines requirements. Section, Component viewpoint, defines the information needed and manipulated by the system and identifies the main software components of the platform and the relationships between them. Section 5, Distribution viewpoint, describes the distribution aspect of the platform. Finally, section 6, Realisation viewpoint, specifies and justifies the choice of baseline technologies for the realisation.

There are a few deviations from the *Arcade* guideline, primarily omission of some recommended sections, because we felt that including them would only lead to repetition.

¹ Project description can be found at:
http://cordis.europa.eu/fetch?CALLER=PROJ_ICT&ACTION=D&CAT=PROJ&RCN=93776

2 Context Viewpoint

2.1 AAL Domain

Although the project address the Internet of things and M2M and their integration into the future internet in general, the AAL domain has been chosen as an example domain to drive the project. We believe that this domain provides good examples of the kind of scenarios and requirements related to the integration of sensors and embedded devices in general with services and applications hosted partly in the cloud and partly on personal and/or mobile terminals.

The following definition of AAL is given by the ambient assisted living joint programme[ref <http://www.aal-europe.eu/about-us>].

The concept of Ambient Assisted Living is understood as

- to extend the time people can live in their preferred environment by increasing their autonomy, self-confidence and mobility,
- to support maintaining health and functional capability of the elderly individuals,
- to promote a better and healthier lifestyle for individuals at risk,
- to enhance the security, to prevent social isolation and to support maintaining the multifunctional network around the individual,
- to support caregivers, families and care organizations,
- to increase the efficiency and productivity of used resources in the ageing societies.

2.2 Scenarios

To illustrate the intended use of the proposed architecture, we include some scenarios from the AAL domain, exemplifying the kind of functionality typically provided by applications in this domain, the kind of devices involved and their roles, and relevant adaptation and evolution behaviours.

The scenarios are about Mary, an elderly person living alone, and whose health condition worsens gradually. She wants to continue living alone in her house, so her family together with health care personnel incrementally build up a AAL system in the house to make that possible despite several severe health problems. An overview of the system is shown in Figure 1. The system and its evolution and use is explained in more detail below.

Mary suffers from Parkinson's disease, and because of the shivering she has problems using ordinary keys to lock and unlock the doors in her house. Therefore automatic door locks (L) are installed and Mary is equipped with an arm wrist device (G) using NFC to communicate with the locks, so when she operates the door lever, the lock opens automatically, and closes again when she close the door.

Mary also suffers from auricular fibrillation. At irregular intervals her heart will start beating very fast while hardly pumping blood for a short period. In some cases this may cause her to faint and be in acute need of medical treatment. To deal with this Mary is equipped with a heartbeat sensor (S) which reports abnormal heartbeat to a nearby hospital via the arm wrist device and a WLAN installed in the house, so the hospital can send a medic team to assist when an attack occurs.

To help filter out the harmless cases, input from the motion detectors (M) initially installed as part of the burglary alarm system are used to detect if Mary is moving. The WLAN is extended with more base stations so the arm-wrist device can locate her accurately. A PC in the house (C) collects this information and triggers the alarm if Mary has an attack and is not moving. It also reports all irregular heart activity to a logging service hosted in "the cloud", where it can be inspected by authorised health personnel (K). When

Mary leaves the house, the role of the PC is taken over by her Smartphone (P), which reports to the Shepherd server via GSM and uses its built-in GPS for localisation.

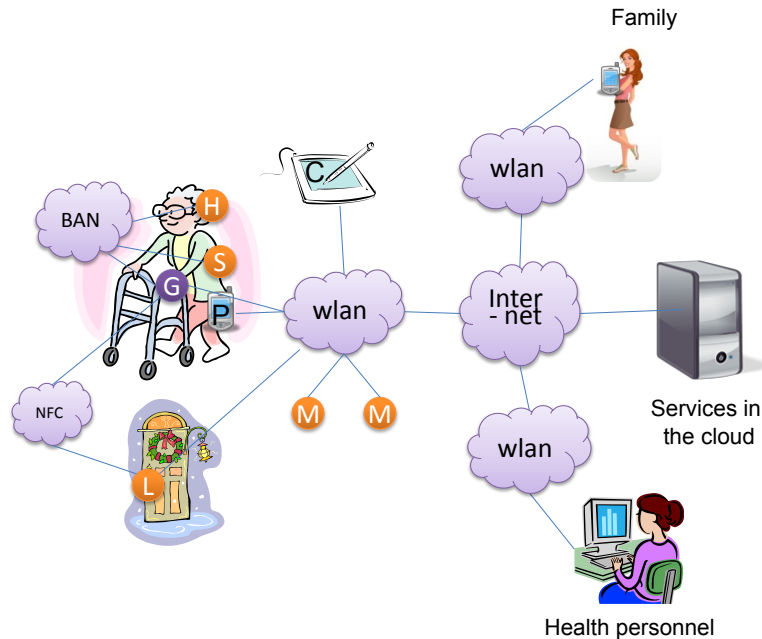


Figure 1 Example deployment scenario

The smartphone also serves as a backup for the PC and can also use the WLAN for more precise localisation inside and around the house. Services in the cloud also monitors the PC and smartphone and alarms the hospital or family about failures.

To make sure that Mary always brings her smartphone when she leaves the house, the arm wrist device and the smartphone keeps track of the distance between them, and if Mary attempts to leave the house (detected by the door lock application) without carrying the phone, the arm wrist device will remind her using her hearing aid (H).

Finally Mary has started to develop dementia, meaning that she easily forgets appointments and she may get lost in neighbourhoods where she normally knows her way. To cope with this problem Mary uses a calendar to remind her about appointments and a navigation system installed on her smartphone, advising her in situations when she gets lost using the hearing aid and the smartphone display. If this is not sufficient the navigator will notify a family member or caring service about the situation so she can be picked up.

We assume that the arm wrist device serves as a gateway for a body area network integrating the hearing aid and the heartbeat sensor (and possibly other biometrical sensors) into the system. The door locks, in addition to communicating with the arm wrist unit via NFC, also communicate with the burglary alarm system via WLAN. These devices do not run the FI-M2M runtime platform.

2.2.1 Adaptation

The first scenario is an example of how the system is used to tackle a common task, namely a visit to the health centre where Mary have appointments with the doctor more or less regularly to follow up on her health status. Additional actors in this scenario are:

- the Bus which Mary uses to get from her house to the health centre.

- the public transportation company operating the bus line.

Mary has difficulties remembering appointments and sometimes she does not recognise where she is, even in well-known environments. To help her overcome these difficulties she makes use of the calendar on her smartphone, which keep track of and alert about upcoming appointments and a travel assistant application to plan an itinerary on public transport to go to appointments when necessary and guide her on her way.

The city where Mary lives has an electronic ticketing system for its public transportation system, and the travel assistant application also takes care of buying and validating tickets.

The following flow of events could be imagined.

- At the end of the last visit to the health centre the secretary entered a new appointment in Mary's calendar.
- The TA notices the appointment, plans an itinerary and sets the advance notice time so that Mary has ample time to get ready and walk to the bus stop. Several itinerary services are available, one with a fee that guarantees that information about the itinerary is not spread to 3rd parties, and one that is free to use, but without such a guarantee. In accordance with her profile, the TA prefers the service with a fee that does not spread information about her movements.
- The TA alerts Mary that it is time to get ready for the appointment.
- Mary leaves the house and starts walking to the bus stop.
- The TA tracks the route so it can advice Mary if she takes the wrong way, which because of her dementia symptoms she sometimes does.
- The TA buys the ticket on the way to the bus stop, using the ticket service of the bus line. The bus line offers payment by credit card and Paypal. Mary has both a credit card and a Paypal account, so the TA may use both. Mary normally prefers payment by credit card. However this day a security issue has been detected with the credit card service, so Paypal is used instead.
- Mary needs to be reminded to leave the bus at her destination stop, otherwise she may easily forget. Some bus companies offer a service that allows the TA to ask the driver to make sure that Mary gets off the bus at her destination stop. Alternatively the TA may remind her itself through the hearing aid. According to Mary's profile, she prefers to be reminded by the driver, but only if privacy is respected, i.e. trip data is deleted as soon as she leaves the bus. The local bus company where Mary lives guarantees this, so the driver assistance is selected.

2.2.2 Another adaptation scenario:

Mary is visiting her daughter's family in another city. The daughters house is not equipped in the same way as Mary's flat, so the applications on Marys arm wrist device and phone has to adapt to the new environment:

- The house does not have motion detectors in the rooms. However, it is a big house and the in-house WLAN has several base stations and therefore can be used for fairly accurate indoor positioning. The heartbeat monitoring application adapts accordingly.
- The house does not have electronic door locks, so the heartbeat monitoring application on the arm wrist device uses WLAN location instead to detect if Mary is about to leave the house without her phone, and therefore must be reminded.
- The local bus company in the city where the daughter lives does provide the driver assistance service, but does not provide the privacy guarantee. Therefore the TA will not use this service, but rather do the reminding itself.

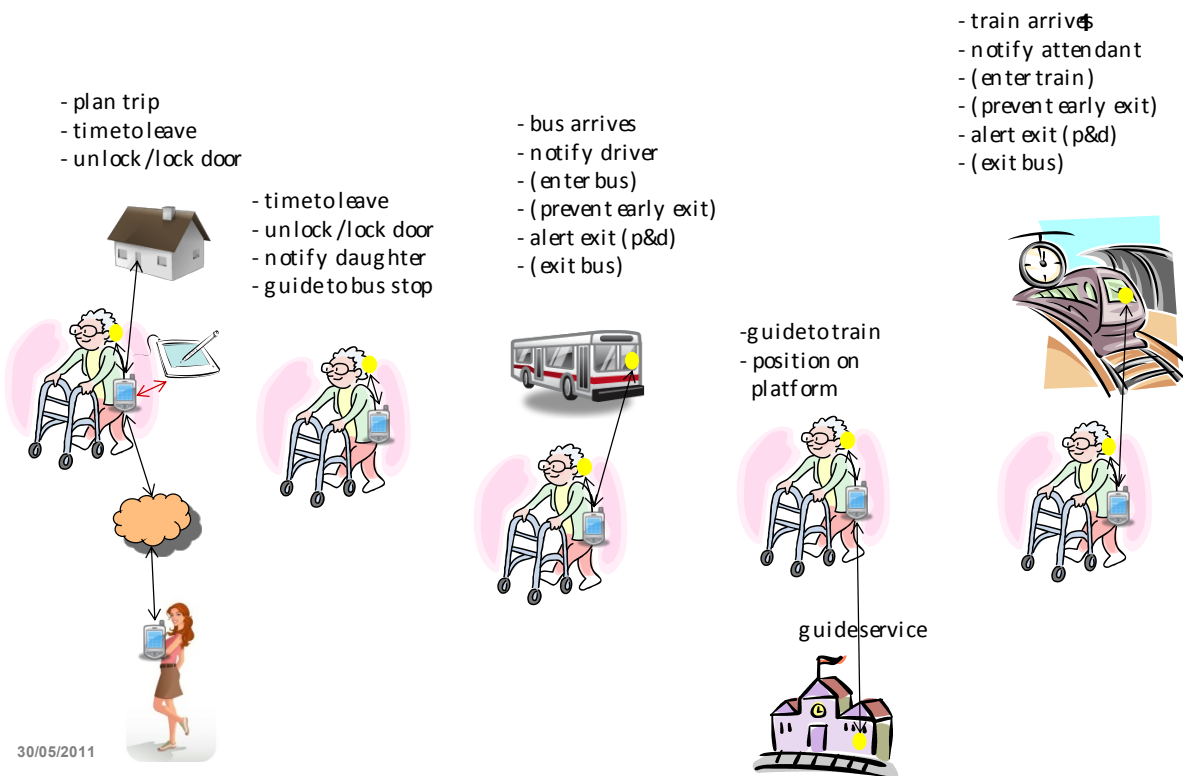


Figure 2 Visiting Family scenario

2.2.3 Development

To enable the initial deployment, usage and evolution scenarios described above require the development of a set of flexible/extendable software applications and services and publishing them in the app store hosted in “the cloud”. Several independent development groups may contribute and the software may be developed, released and published incrementally. These processes are supported by several of the use cases mentioned later in this document, but will be further elaborated upon in step two of this project.

2.2.4 Evolution

This AAL system has been installed in steps following the development of Mary’s need for assistance. For each step in the evolution of the AAL system, family members and/or health personnel use the smart deployment and evolution assistant to find and select the required functionality, to select (and possibly purchase) the necessary hardware, and to deploy the required software parts on the relevant devices.

- First Mary develops Parkinson’s disease, and electronic locks were installed and Mary got the arm wrist device functioning as an electronic key. A key administration service is available in “the cloud” and both Mary (using a client application on her home pc) and the home care centre may hand out and destroy keys.
- After a while, when Mary develops auricular fibrillation and the system is extended with the heartbeat sensor, the motion detection sensors already present in the house are integrated, and indoor positioning is improved by installing more WLAN base stations. Furthermore, the door lock

application is extended to report when Mary enters or leaves the house to the heartbeat monitoring application.

- Finally Mary also starts to develop dementia and the personal assistant application is installed on her smart-phone.

2.3 Environment

As illustrated by the scenarios above, the environment where AAL systems will be deployed and used can be characterised as follows:

- Many heterogeneous sensors and other small devices, ranging from tiny severely resource constrained ones like the heartbeat sensor and the hearing aid, via more powerful but still battery powered devices like the arm wrist unit and the motion sensors, to powerful handheld terminals like the smartphone.
- Heterogeneous communications infrastructure (BAN, WLAN, GSM, ...)
- Heterogeneous service publication and discovery technologies, including SENSEI - RD / SensiNode NanoServices
- Professional Infrastructure providers, offering a managed “cloud” environment where common shared services, applications and application specific services may be hosted, for example Telenor Objects with their Shepherd platform.

2.4 Work processes

The following work processes should be supported by the proposed platform. The work processes are related to the concerns documented through the scenarios above and the stakeholder analysis in section 2.5. In addition to these the scenarios outlined in subsection 2.2 has some extended requirements related to the work processes.

- Develop an application, component or service
- Extend an existing application with new functionality and/or context awareness and adaptation capabilities (Requirement from scenario)
- Release an application to the SW repository. (Ability of easy interaction, Business concerns, Compliance to standards)
- Release an application extension to the SW repository (Ability of easy interaction)
- Configure and install an initial system for a new user (Ability of easy interaction)
- Configure and install an extension for a new assistance function for an already installed system

2.5 Business to system Mapping

2.5.1 Stakeholders

This subsection contains a stakeholder analysis for the AAL domain. This work is done in the context of the universAAL project, which has a wider scope than this project. Therefore the stakeholder analysis also covers aspects, which in the context of this project belongs to step two.

Figure 3 shows a high-level view of a value network that has been used as a basis for creating the UNIVERSAAL Reference Architecture (RA)².

² UniversAAL D1.3-PartIII: <http://universaal.org/images/stories/deliverables/D1.3-B.pdf>

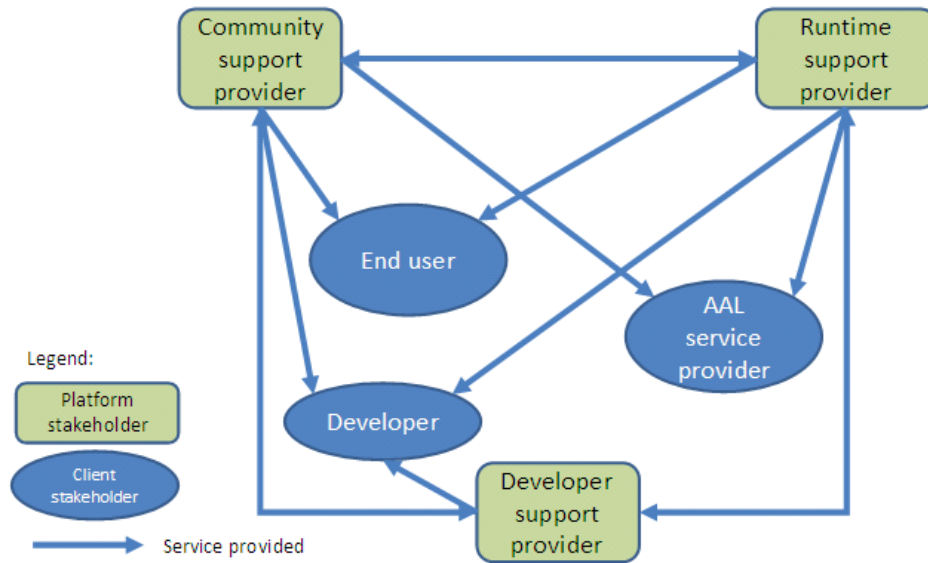


Figure 3: The AAL value network supported by universAAL

We are concerned with platforms for AAL. AAL stakeholders are grouped into two groups:

- **Platform stakeholders** (rounded rectangles in Figure 3): These are stakeholders who provide and operate platform components in an AAL environment. We envision a value network where these platform stakeholders will provide services of value to each other and to client stakeholders.
 - Runtime Support Provider – provides support for proper functioning of AAL runtime environments. The AAL runtime environment is where users (the elderly, but also their caregivers) live and interact with the AAL technologies. Typical examples are smart homes.
 - Developer Support Provider – provides software engineering support for AAL solutions developers. AAL services contain a large amount of software, and support for proper development, testing and maintenance of this software is crucial.
 - Community Support Provider – provides infrastructure for end users, service providers and developers to build community. The community can be used to trade services, collect user requirements, enable collaboration among service providers etc.
- **Client stakeholders** (ovals in): These are stakeholders who use the AAL platform functionality provided by the above three groups of platform stakeholders. Client stakeholders considered so far in our RA are:
 - End users - non-technical end users such as assisted persons and their caregivers who consume AAL services.
 - Developers - developers of AAL solutions and technologies.
 - AAL service providers - providers of AAL services to the end users

The above classification can be easily mapped onto the stakeholder classification provided by AALIANCE³:

³ G. Van Den Broek, F. Cavallo, and C. Wehrmann, eds. 2010. Ambient Assisted Living Roadmap. IOS Press. <http://www.aaliance.eu/public/documents/aaliance-roadmap/ambient-assisted-living-roadmap>.

- *Primary stakeholders*: These correspond to our end users.
- *Secondary stakeholders*: They correspond to the three platform stakeholders above (if we consider platform in general as a service provided to platform users). Also AAL service providers are secondary stakeholders.
- *Tertiary stakeholders*: This category can be mapped to our developer category.

Another class of stakeholders, i.e. **authorities**, is not yet addressed in the RA but is planned for the next iteration. For now it suffices to say that requirements such as support for audits and privacy protection are being considered in designing the RA.

The list of stakeholders above is highly generalized. In many specific cases involving specific solutions a more specific list of stakeholders will be required. Deliverable UNIVERSAAL D1.1 has a more elaborate list of stakeholders, with examples of specific types of each stakeholder group presented above. For the discussions regarding the reference architecture, the list above suffices.

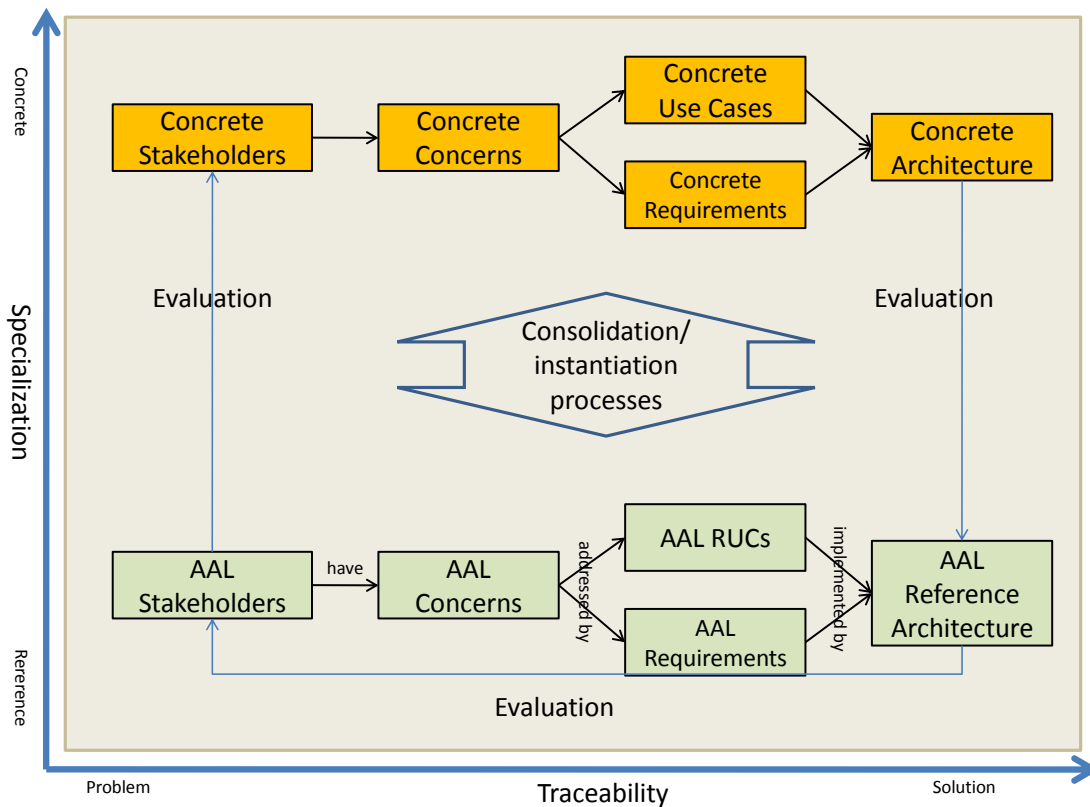


Figure 4: The two purposes of our Reference Architecture in form of the two dimensions of traceability and specialization

After identifying the stakeholders of relevance to the RA, we would like to look into their major **concerns**, i.e. overall expectations that each stakeholder has from the RA (see the traceability axis Figure 4). Deliverables D1.1 and D1.2 in universAAL document a thorough concerns and requirements analysis. What is important for us here is to point out the concerns that have been addressed in the current version of the RA. The RA addresses a number of concerns as briefly described below based on each client stakeholder:

- **End users:** Elderly, their families and caregivers are mainly concerned with service quality, which is related to how services are developed (supported by developer support provider, see), acquired (supported by community support provider) and operated (supported by runtime support provider). Good development environments will lead to better quality software to run the AAL service, and will attract talented developers. Efficient communication with service providers and among service providers will result in services that are better personalized for each end user. High quality runtime support will guarantee e.g. fault-tolerance and timely response, and protect privacy. The RA tries to address these concerns by supporting the platform stakeholders in the three central areas shown in .
- **Developers:** are mainly concerned with good development environments, a knowledgeable community of developers (addressed by developer support providers, see), and access to a market for their software (addressed by community support providers). Our RA supports developers as a major stakeholder and allows developers to participate in a community together with service providers and end users.
- **AAL service providers:** are concerned with efficient communication with their end users (supported by community support provider), fluent interaction with the runtime environment for their services (supported by runtime support provider), and access to talented developers for development and personalization of their services (supported by community support provider and developer support provider). The services provided by our RA allow service providers to deploy, monitor and otherwise manage their services as provided to their end users.

For platform stakeholders in general, the RA addresses a major concern, which is to promote the standardization of the platform ecosystem for AAL technologies.

It is these concerns (documented in more details in UNIVERSAAL D1.1⁴) that have guided the development of our RA. These concerns have resulted in a set of reference use cases (RUCs) that play a central role in laying out the requirements for AAL platforms.

2.5.2 Reference use cases currently supported

In order to make stakeholder concerns more tangible and understandable to multiple parties, and make them easily mapped onto a technical architecture, we have developed a set of Reference Use Cases (RUCs). RUCs demonstrate, in an easy-to-understand way, what specific concern of a stakeholder means and how it can be supported by AAL technologies. Current RUCs are documented in UNIVERSAAL D1.1, but we will here discuss shortly how the RA is addressing each RUC. Three categories of RUCs have been defined in UNIVERSAAL D1.1:

- Category 1: A platform for creating innovative and commercial grade AAL services. These are use cases that show how a platform can support higher QoS for AAL services at runtime.
- Category 2: A platform for creating a market place for AAL services. These are use cases that show how a platform can promote an online community for AAL stakeholders, and remove barriers for uptake of such services through promoting collaboration and business.
- Category 3: A platform for supporting developers and users in innovating the AAL market. These are use cases that show how the platform can help building a knowledgeable and capable community of developers who can produce high-quality AAL services in collaboration with users and service providers.

⁴ <http://universaal.org/images/stories/deliverables/D1.1-B.pdf>

There is an obvious mapping of the three categories to the three platform stakeholders as we believe platform stakeholders wish to specialize in their own area of expertise. However, the RA also considers communication among the three platform stakeholders as we will see later. Table 1 below shows an overview of which RUCs are supported in the current version of RA. The column called “How supported?” denotes an RA Service (RAS) that is identified as part of the services provided by our RA. The RAS definitions are provided later in Sections 3.2.2 through 3.2.5. For detailed description of each RUC please consult UNIVERSAAL D1.1. The column marked “ACT” describes which if any of the FI-M2M project activities are relevant for the RUC. Some RUCs will be marked “N/A” if they can’t be applied to our project directly and some will be marked as not supported at all (“N/S”)

Table 1: List of Reference Use Cases and their support in the current Reference Architecture

Cat.‘	RUC name and description	Currently supported?	How supported?	ACT
1	RUC#1: Supporting rich human computer interaction	Yes	RAS#1.7, 1.24	N/A
1	RUC#2: Supporting intelligent context management and hardware abstraction	Yes	RAS#1.28,	1
1	RUC#3: Enabling system driven interaction	Yes	RAS#2.17	1
1	RUC#4: Supporting continuity of care in different AAL Spaces	Yes	RAS#1.8	N/A
1	RUC#5: End user security and privacy management.	Partially	RAS#1.5, 1.8, 1.9, 1.38, 1.39, 1.47	N/S
1	RUC#6: Installation, configuration and management of Platform components	Yes	RAS#2.4,1.24, 1.21, 1.2, 1.20	2
1	RUC#7: Remote/local operation and provision of AAL Services	Yes	RAS, 1.20, 1.4, 1.2	1
1	RUC#8: Support for multi-user AAL Services in one AAL Space	Yes	RAS#1.4, 1.23, 1.8	N/A
1	RUC#9: Interfacing with existing information systems	No		1
2	RUC#10: Services providers offering AAL Services in uStore	Yes	RAS#2.13, RAS#2.18, RAS#2.19	2
2	RUC#11: uStore allows users to easily find and acquire AAL services	Yes	RAS#2.4, RAS#2.5, RAS#2.7, RAS#2.9, RAS#2.11	2
2	RUC#12: uStore supports exploitation of different business models	Partially	RAS#2.9, RAS#2.15,	2

Cat.‘	RUC name and description	Currently supported?	How supported?	ACT
			RAS#2.19, RAS#2.23, RAS#2.24	
2	RUC#13: Capturing and utilizing user feedback through the uStore	Yes	RAS#2.8, RAS#2.10, RAS#2.12, RAS#2.20, RAS#2.22, RAS#2.25, RAS#2.26	2
3	RUC#14: IDE for rapid development of AAL services, including templates, wizards, libraries and process support tools	Yes	RAS#3.1, RAS#3.2	1
3	RUC#15: Model-based development of AAL services through integrated model transformation tools	Yes	RAS#3.2	1
3	RUC#16: Support for online elicitation of requirements and collection of runtime feedback from users of AAL services	No		2
3	RUC#17: IDE support for advanced search, reuse and sharing of service components and developer resources through the Developer Depot	Partially	RAS#3.1, RAS#3.2, RAS#3.4	2
3	RUC#18: Generic tool support for utilizing personalization capabilities offered by universAAL runtime environment	No		2

3 Requirements Viewpoint

3.1 High level requirements

Based on the understanding of the needs of the domain presented in chapter 2, we have derived the following high-level requirements:

- Support the development of dynamically adaptive SoA style systems, i.e. systems consisting of many parts collaborating by providing services to and using services from each other and where system topology forms dynamically and adapts to changes in the environment:
 - Nodes and services appearing and disappearing.
 - Resources becoming exhausted (e.g. batteries running flat).
 - Communication links appearing and disappearing and varying in capacity and other characteristics.

- Support temporarily disconnected “islands”, i.e. subsystems which continue to work despite being disconnected from the “Internet”, possibly with reduced functionality.
- Enable extension of already deployed systems, both in terms of extended functionality and in terms of extended context awareness and adaptation capabilities.

In addition the following requirements are imposed by the project owners:

- Base as far as possible on technologies available in Telenor and SINTEF.
- Comply as far as possible with emerging standards in the Internet of things domain.

3.2 Target System Mapping Model

We have so far seen who our major stakeholders are, what major concerns each of them have, and which of these concerns are currently addressed in the RA. In this section we take the first step to bridge the so-far business-related context to what will come in the following sections, i.e. the technical ICT architecture. We will look at how each of the concerns is addressed in form of services exchanged among stakeholders.

A first step in this process is to map stakeholders to services they provide to each other. We have already seen the interactions among the stakeholders in Figure 3. The lines among the stakeholders in Figure 3 denote services of value exchanged among the stakeholders. Our RA is concerned with these services and how they can be implemented in a standardized way.

The value network illustrated in Figure 3 is mapped to a SOAML *Services Architecture* diagram in Figure 5 below. In this figure participants (rectangles) denote stakeholders and service contracts (ovals) denote services that are exchanged among stakeholders. Client stakeholders are shown in orange (grey in B/W print), and platform stakeholders are shown as rectangles with thick borders.

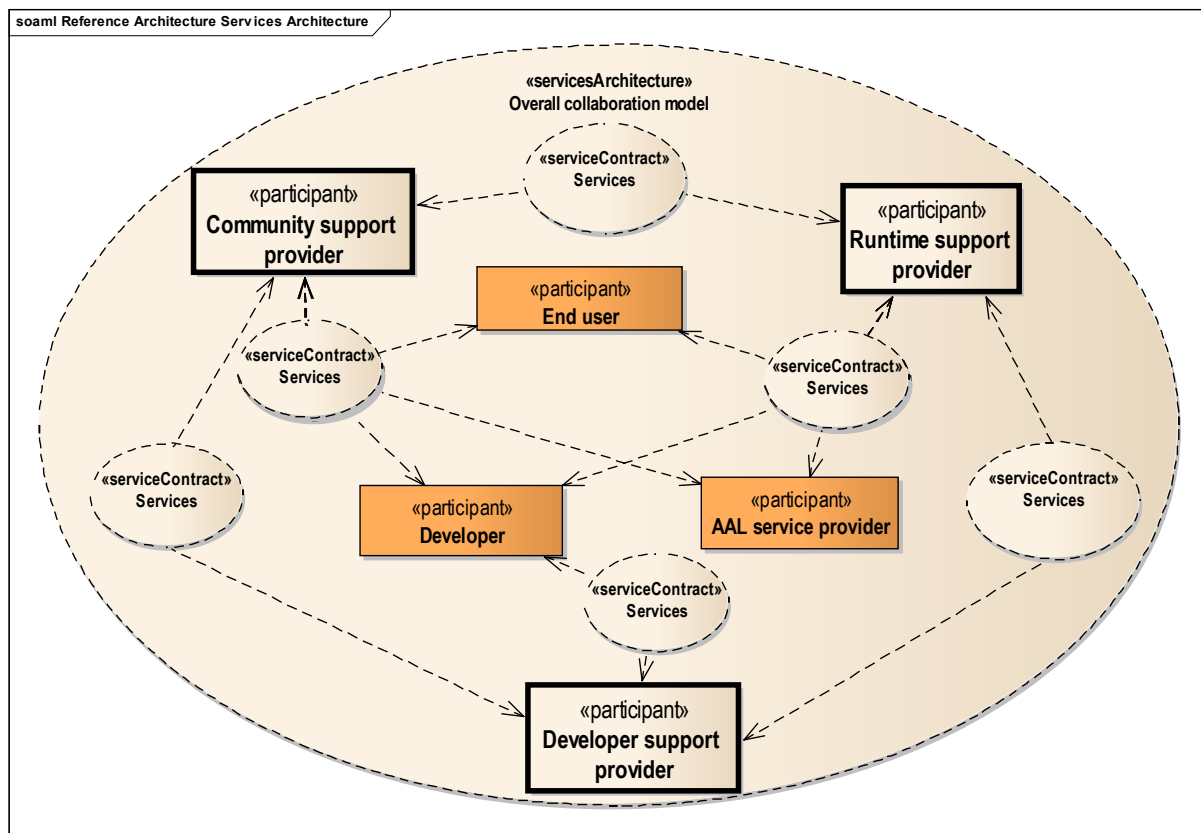


Figure 5: Services Architecture for AAL main stakeholders and their services capabilities

According to SOAML services that are exchanged among stakeholders are modelled using a specific type of collaboration called *service contract*. In the figure above each of the service contracts (ovals) is a placeholder for all the services that are exchanged between the two parts of the service contract. E.g. the service contract between Community Support Provider and Runtime Support Provider denotes all the services provided by each of the two to the other. We will see the details of these services in Sections 3.2.3 through 3.2.5.

In this section we will present the different collaboration models in the stakeholder analysis (2.5.1 and 2.5.2) and M2M platform (4.4.1). The models for the stakeholder analysis are included to give the reader a better context to interpret the stakeholder analysis. The reader should note that the interfaces and collaboration models does not directly represent the M2M platform, but rather a more exhaustive description of interfaces and interactions that covers the requirements provided in earlier sections. As such the subsections 2.5.1 and 2.5.2 should be regarded as examples from the AAL domain that the stakeholder analysis is based on. Only parts of these subsections will be generalised into the M2M architecture.

During the service analysis part of this section we have marked the relevant services mapped from the RUCs we support.

3.2.1 System categories

In order to assign a high-level implementation (by a stakeholder) to the service capabilities, we map each of the participants in Figure 5 onto a UML component as shown in Figure 6. In addition we have a component called AAL service, which is the incarnation of the AAL service itself in the platform. The information concepts in Figure 6 show how an AAL service is represented in the platform. Seen from our standpoint, an AAL service is a third party component. Nevertheless it is a central component to relate to since a major part

of the services provided by *Runtime support for AAL spaces* is provided to AAL services. In the RA, AAL service will only consume services provided by *Runtime support for AAL spaces*.

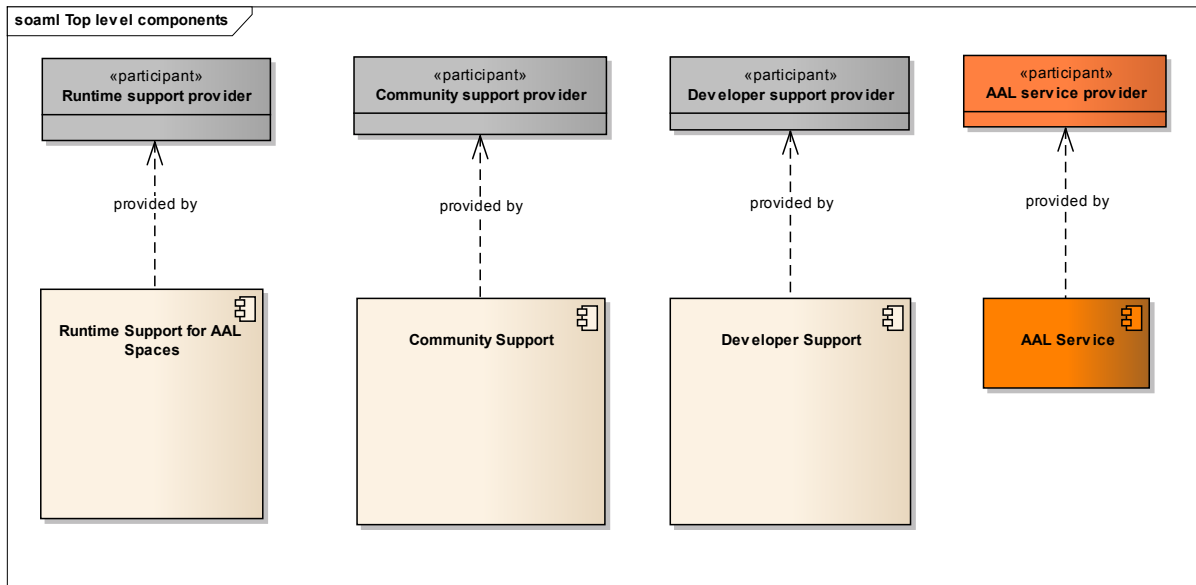


Figure 6: Top level platform components for AAL platform reference architecture

In addition to these top-level platform components (which represent service provided and consumed by platform stakeholders), we introduce three top-level tools components as shown in Figure 7. Each of these tool components represents interactions with client stakeholders. Seen from another perspective, these tools mainly represent the “presentation layer” in the architecture, while the platform components in Figure 6 represent the “business logic”.

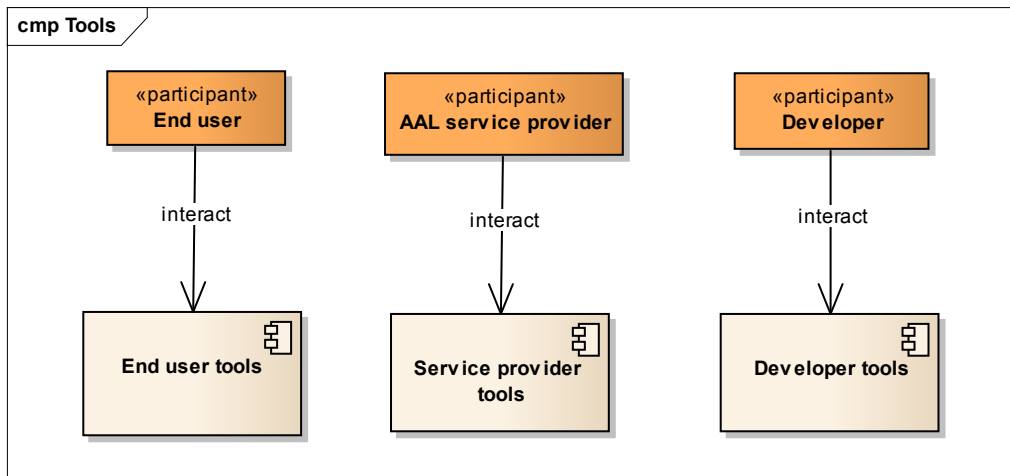


Figure 7: Each client stakeholder is presented a tool component for emphasizing user friendly interaction

3.2.2 The behavioural aspects (service collaboration patterns)

In the following sections we will look into the details of each of the three platform components and associated tool component. The goal is to identify the high-level services that are provided at four different directions as shown in Table 2. Each type of service will be described in its own section using a table and a diagram. Each service is labeled with an ID RAS#A.B where RAS means Reference Architecture Service, and A and B are numbers. These IDs are used as short-cut reference to the services.

Table 2: Types of services exchanged in the Reference Architecture.

Type of service	Meaning
Platform-to-Platform	Services that are provided by one platform stakeholder to other platform stakeholders of a different type.
Platform-to-Client	Services that are provided by a platform stakeholder to client stakeholders.
Peer	Services that are provided by one type of platform stakeholder to other platform stakeholders of the same type (e.g. services exchange among runtime support providers).
Platform-to-AAL Service	Services provided at runtime to operational AAL Services. This category of services is only relevant for AAL Runtime Support Providers.

The following three sub-sections describe the above types of services for each of the three platform stakeholders.

3.2.3 Runtime support provider

3.2.3.1 Platform-to-Platform services

In the following sections (4.2.3 to and including 4.2.5) the column marked “ACT” describes which if any of the FI-M2M project activities are relevant for the RUC. Rows without any value in this column is not applicable to our project. Thus some tables will not have this column, as none of the RASes listed is applicable, these tables are included for completeness.

Table 3: Platform-to-platform services provided by Runtime support providers

id	Service	Provided to	Description	ACT
RAS#1.1	Provide access	Community support provider	Provide usage data and profile information	
RAS#1.2	Remote maintenance	Community support provider	Remote Configuration, management and maintenance of software's	1,2
RAS#1.3	Get audit logs	Community support provider	Getting usage data and profile information	
RAS#1.4	Remote personalization	Community support provider	No description	1
RAS#1.5	Security Keys management	Community support provider	Manage security certificates e.g. after expiry or issue a new certificate or a revoke an already issued certificate	

id	Service	Provided to	Description	ACT
RAS#1.6	Testing applications	Developer support provider	Support for sandbox testing of applications? E.g. deploying applications in a test runtime?	

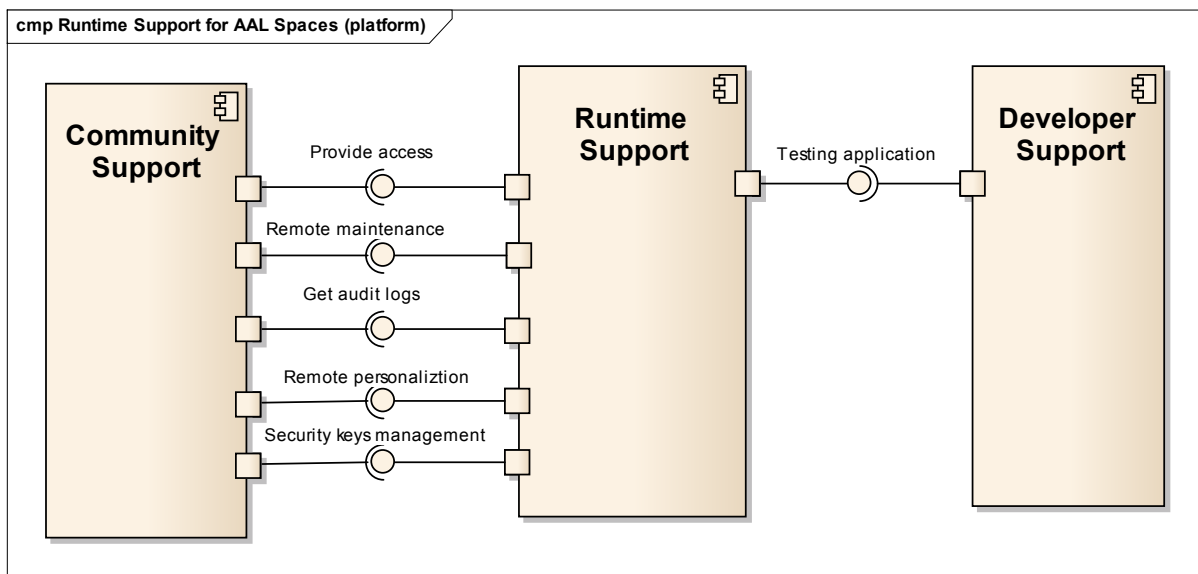


Figure 8: Platform-to-platform services provided by Runtime support providers

3.2.3.2 Platform-to-Client services

Table 4: Platform-to-client services provided by Runtime support provider

Id	Service	Provided to	Description	ACT
RAS#1.7	Manage user interaction	End User	Facilitates interaction of the End user with the system such as by providing consisting look and feel and adaptability etc.	
RAS#1.8	Encryption, signing, authentication etc.	End User	End user can use encrypt and sign their data. They can also authenticate with the remote service.	
RAS#1.9	Consent policy specification	End User	End user can specify his/her consent preferences which denotes who can access his /her data	
RAS#1.20	Remote configuration	AAL service	AAL service provider can remotely	1,2

Id	Service	Provided to	Description	ACT
		provider	configure the service	
RAS#1.21	Remote maintenance	AAL service provider	AAL service provider can do the maintenance remotely.	2
RAS#1.22	Communicate Audit trail	AAL service provider	Audit trail is communicated to the AAL service provider which may contain information such as usage etc.	
RAS#1.23	Runtime Orchestration of services	AAL service provider	Service orchestrator is needed for runtime orchestration of services which decides which sub-services can be offered from a set of services.	
RAS#1.24	Pluggable user interfaces	AAL service provider	Support the inclusion of new components/interfaces at any time without the need to restart the system.	2

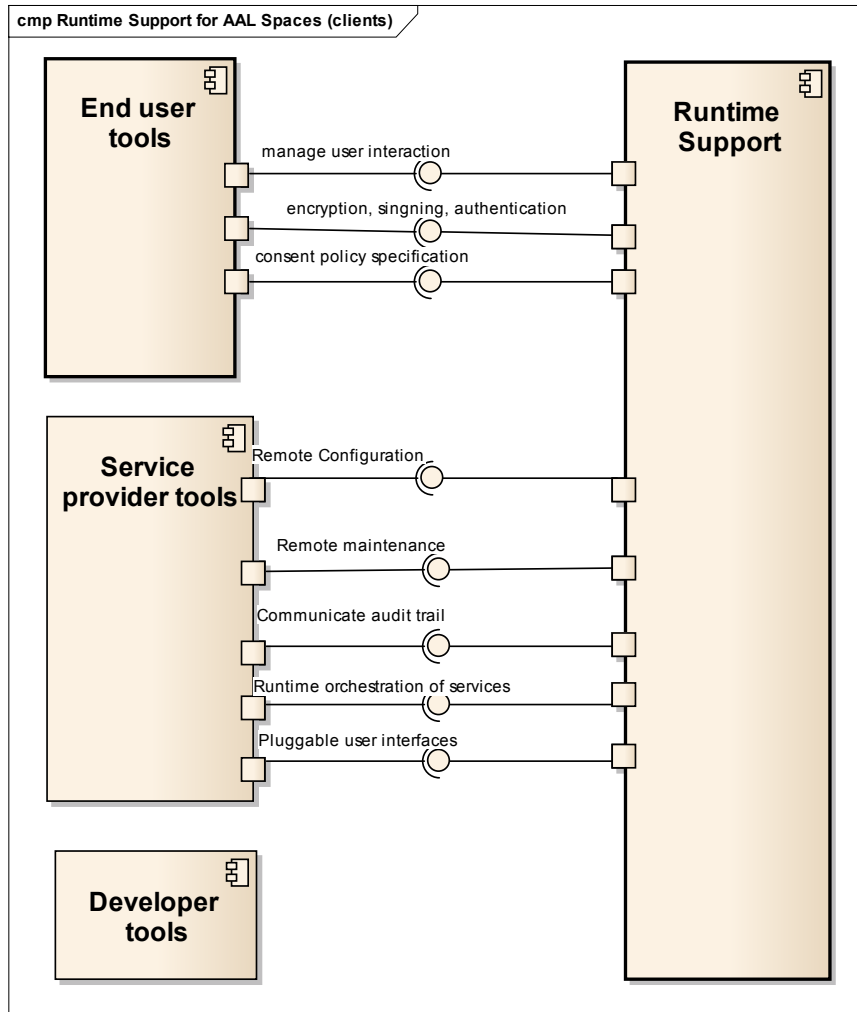


Figure 9: Platform-to-client services provided by Runtime support provider

3.2.3.3 Peer services

Table 5: Peer services provided by Runtime support provider

Id	Service	Provided to	Description	ACT
RAS#1.25	Communication	Other runtime support provider	An AAL aware node has the capability to communicate with another AAL aware node	
RAS#1.26	Discovery	Other runtime support provider	An AAL aware node has the capability to	

Id	Service	Provided to	Description	ACT
			discover another AAL aware node.	
RAS#1.27	Peering	Other runtime support provider	An AAL aware node has the capability to peer with another AAL aware node	
RAS#1.28	Share context information and history	Other runtime support provider	An AAL aware node has the capability to share context history with another AAL aware node	1
RAS#1.29	Share platform information	Other runtime support provider	An AAL aware node has the capability to share platform specific information with another AAL aware node	
RAS#1.30	Share profiling information	Other runtime support provider	An AAL aware node has the capability to share profile information of an end user with another AAL aware node.	

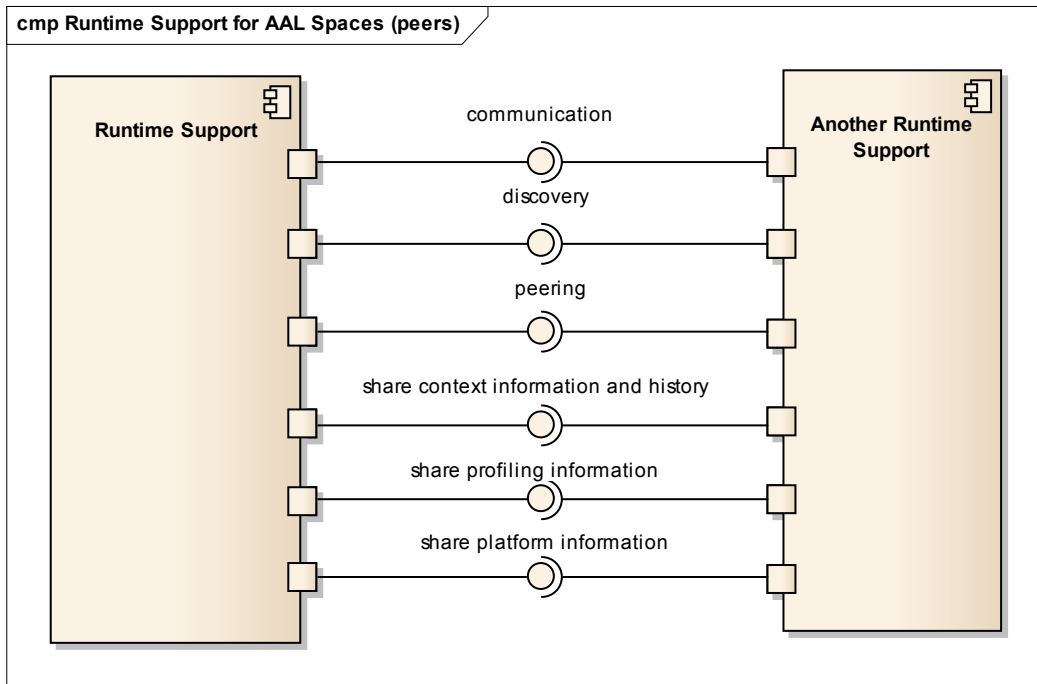


Figure 10: Peer services provided by Runtime support provider

3.2.3.4 Platform-to AAL service

Services provided to AAL service at runtime are a special case for the runtime support part of our platform. These services are described here.

Table 6: Services provided to AAL service at runtime

Id	Service	Provided to	Description
RAS#1.31	Manage multicast communication	AAL service	An AAL service has the capability to manage multicast communication through runtime support provider.
RAS#1.32	Manage stream communication	AAL service	An AAL service has the capability to do stream communication with the runtime support (or AAL aware node).
RAS#1.33	Manage peer communication	AAL service	
RAS#1.34	Join AAL Space	AAL service	AAL service has the capability to join an AAL space.
RAS#1.35	Get information about AAL Space	AAL service	An AAL service has the capability to get the information about an AAL space
RAS#1.36	Discover peers	AAL service	
RAS#1.37	Manage applications	AAL service	An AAL service has the capability to

Id	Service	Provided to	Description
			perform functions related lifecycle management of an application.
RAS#1.38	Manage security keys	AAL service	An AAL service has the capability to manager certificates on an AAL aware node.
RAS#1.39	Encryption, signing and authentication	AAL service	An AAL service has the capability to receive encrypted and signed information. It can also authenticate the user.
RAS#1.40	Run a test	AAL service	An AAL service can run a test.
RAS#1.41	Activate monitoring	AAL service	An AAL service has the capability to remotely activate monitoring
RAS#1.42	Manage service workflows	AAL service	
RAS#1.43	Get context history	AAL service	An AAL service has the capability to obtain context history from an AAL aware node
RAS#1.44	Register reasoning rules	AAL service	
RAS#1.45	Obtain platform information	AAL service	An AAL service has the capability to obtain the platform specific information from AAL aware node.
RAS#1.46	Obtain profiling information	AAL service	An AAL service has the capability to obtain the user profile information from the AAL aware node.
RAS#1.47	Obtain End user consent	AAL service	An AAL service can get the user consent through AAL aware node.

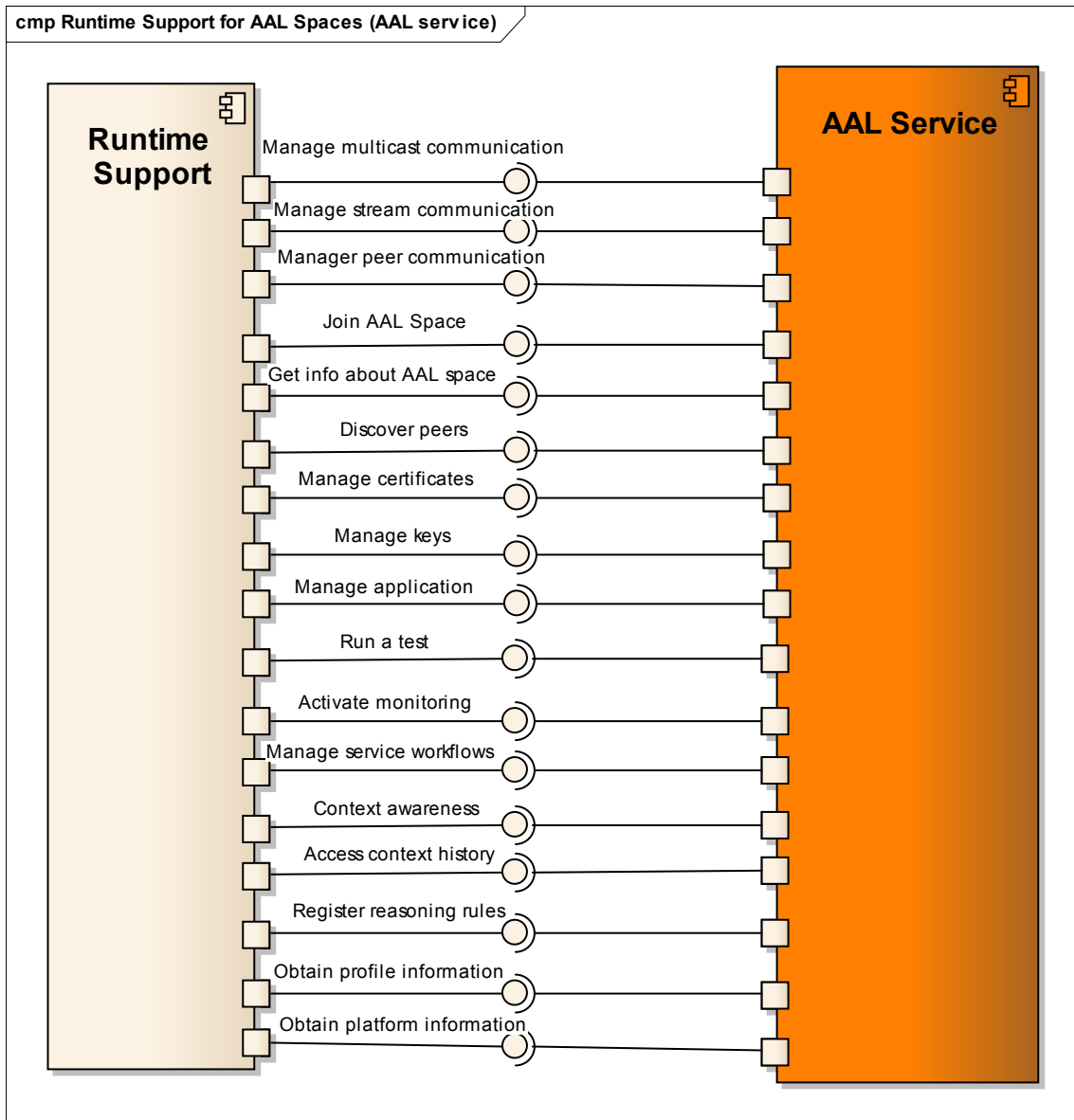


Figure 11: Services provided to AAL service at runtime

3.2.4 Community support provider

3.2.4.1 Platform-to-Platform services

Table 7: Platform-to-Platform services provided by Community support provider

Id	Service	Provided to	Description	ACT
RAS#2.1	Publish development tools	Developer support provider	Upload, publish and advertise development tools, training courses, related documentation, etc.	

Id	Service	Provided to	Description	ACT
RAS#2.2	Get feedback	Developer support provider	Get feedback on published tools	
RAS#2.3	Get ideas	Developer support provider	Get feedback about market needs	
RAS#2.4	Manage profiles	Runtime support provider	Storing and accessing/synchronizing user and AAL space profiles	2
RAS#2.5	Download software	Runtime support provider	Download software to runtime platform	2
RAS#2.6	Get feedback	Runtime support provider	Get feedback about runtime support provider	

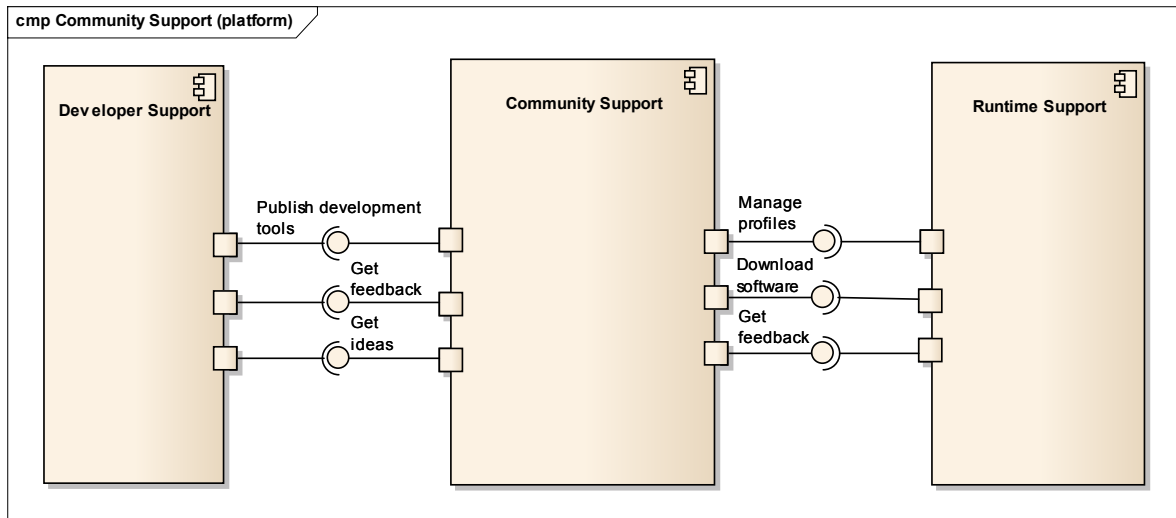


Figure 12: Platform-to-Platform services provided by Community support provider

3.2.4.2 Platform-to-Client services

Table 8: Platform-to-Client services provided by Community support provider

Id	Service	Provided to	Description	ACT
RAS#2.7	Look for AAL Services	End user	Look for AAL Services based on user needs, preferences and existing environment	2
RAS#2.8	Create a request	End user	Create a request describing new AAL Service or new functionality needed	2
RAS#2.9	Acquire AAL Service	End user	Purchase or freely acquire AAL Service according to their status using legal framework (SLA, licences	2

Id	Service	Provided to	Description	ACT
			etc.)	
RAS#2.10	Provide feedback	End user	Provide feedback (comments, ratings etc.) on existing AAL Service or service provider/developer	2
RAS#2.11	Request assistance	End user	Request assistance from AAL Service Providers to install software etc.	2
RAS#2.12	Communicate	End user	Communicate with other users inside AAL User community (exchange experience etc.)	
RAS#2.13	Publish AAL Services	AAL service provider	Publish AAL services using legal & business framework	2
RAS#2.14	Explore the market	AAL service provider	Explore existing AAL Services & AAL Applications	
RAS#2.15	Communicate with developers	AAL service provider	Find and manage business connections with developers from AAL Developer Community	2
RAS#2.16	Communicate	AAL service provider	Communicate inside AAL Service Provider community	
RAS#2.17	Service bundling	AAL service provider	Join services of different service providers in packages	1
RAS#2.18	Run conformance tests	AAL service provider	Run conformance testing on software part of AAL Services	2
RAS#2.19	Require certificate	AAL service provider	Require certificate for AAL Service	2
RAS#2.20	Receive feedback	AAL service provider	Receive feedback from AAL User Community	2
RAS#2.21	Create a request	AAL service provider	Create a request for new AAL Application from AAL Developer Community	2
RAS#2.22	Receive requests	AAL service provider	Receive requests for new services or updates of existing one from AAL User Community	2
RAS#2.23	Upload AAL	Developer	Upload software and software updates as AAL	2

Id	Service	Provided to	Description	ACT
	Applications		applications for free or fee usage	
RAS#2.24	Make business agreements with service providers	Developer	Make business agreements with service providers to provide uploaded AAL applications as part of AAL services	2
RAS#2.25	Receive feedback	Developer	Receive feedback from AAL User Community	2
RAS#2.26	Receive requests	Developer	Receive requests for new applications or updates of existing ones from AAL User Community and AAL Service Provider Community	2
RAS#2.27	Communicate inside AAL Developer Community	Developer	Communicate between developers in AAL Developer Community	
RAS#2.28	Run conformance tests	Developer	Run conformance testing on uploaded software	
RAS#2.29	Require certificate for AAL Application	Developer	Require certificate for AAL Application (based on conformance test results)	

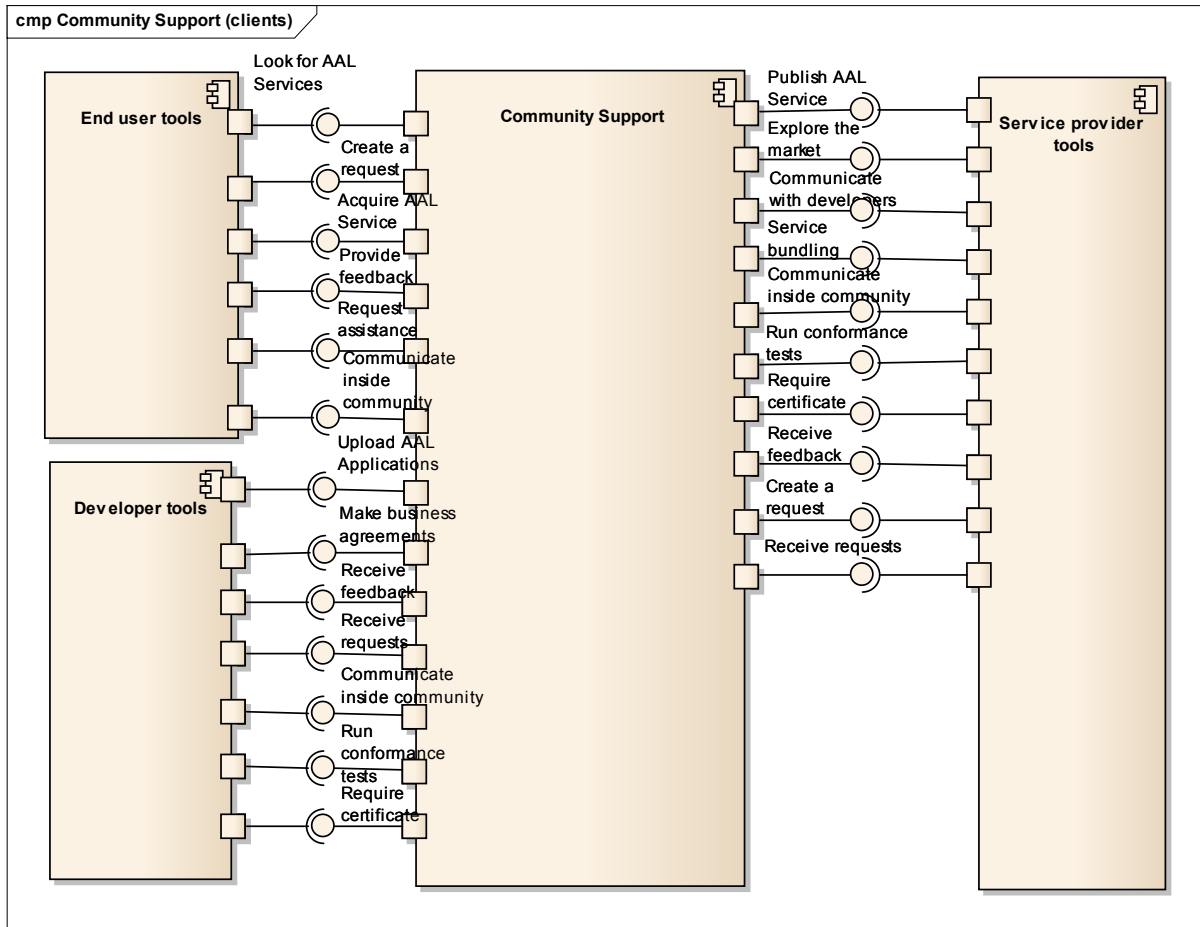


Figure 13: Platform-to-Client services provided by Community support provider

3.2.4.3 Peer services

Table 9: Peer services provided by Community support provider

Id	Service	Provided to	Description
RAS#2.30	Explore market	Community Support Provider	Explore proposed AAL Services and interchange information with users from different communities
RAS#2.31	Link to AAL Services	Community Support Provider	Allow purchasing or acquiring of AAL services from another community support tool

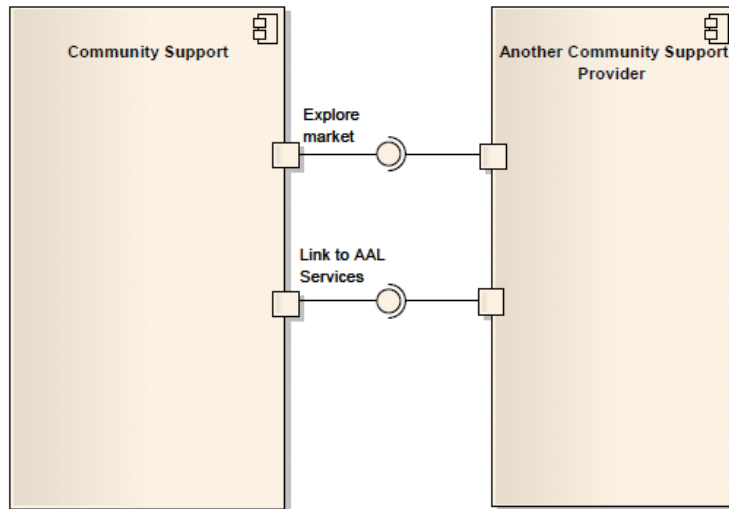


Figure 14: Peer services provided by Community support provider

3.2.5 Developer support provider

3.2.5.1 Platform-to-Platform services

N/A.

3.2.5.2 Platform-to-Client services

Table 10: Platform-to-Client services provider by Developer support provider

Id	Service	Provided to	Description	ACT
RAS#3.1	Use IDE	Developer	Provide IDE with plug-ins etc.	1
RAS#3.2	Use development tools	Developer	Allow using wizards, modelling tools, conformance tools, build tools etc during development process	1
RAS#3.3	Develop in community	Developer	Develop inside AAL Developer community (Exchange ideas, discuss technical issues etc)	
RAS#3.4	Contribute code	Developer	Contribute code to platform	1
RAS#3.5	Read and provide documentation	Developer	Read and provide documentation, guides, tutorials etc	

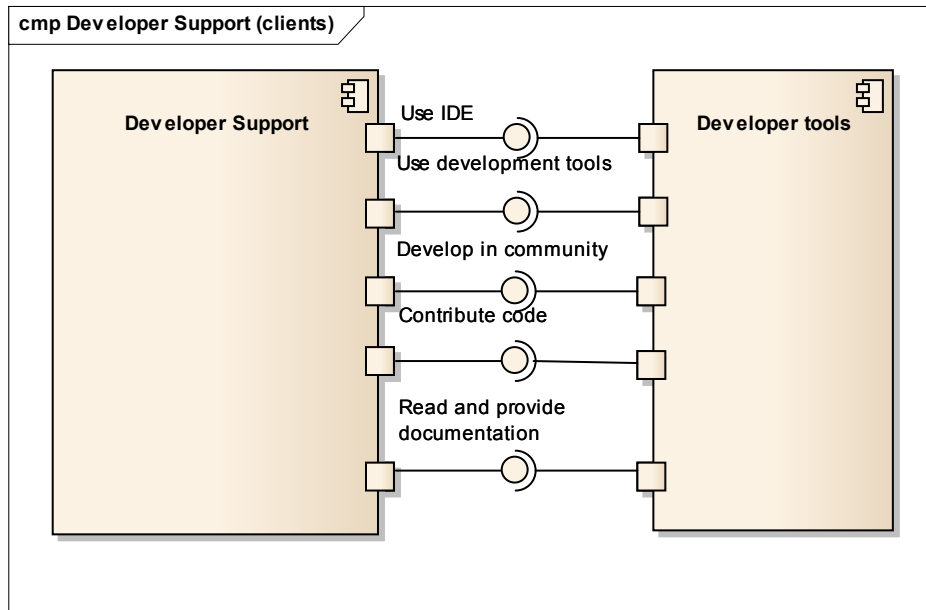


Figure 15: Platform-to-Client services provider by Developer support provider

3.2.5.3 Peer services

Table 11: Peer services provided by Developer support provider

	Service	Provided to	Description
RAS#3.6	Link artefacts	Developer Support Provider	Allow linking of artefacts between developer support tools
RAS#3.7	Exchange	Developer Support Provider	Exchange code, documentation (about concrete architecture etc.)
RAS#3.8	Provide external interoperability	Developer Support Provider	Provide any needed information, documentation and tools that developers would need in order to interact with another platform

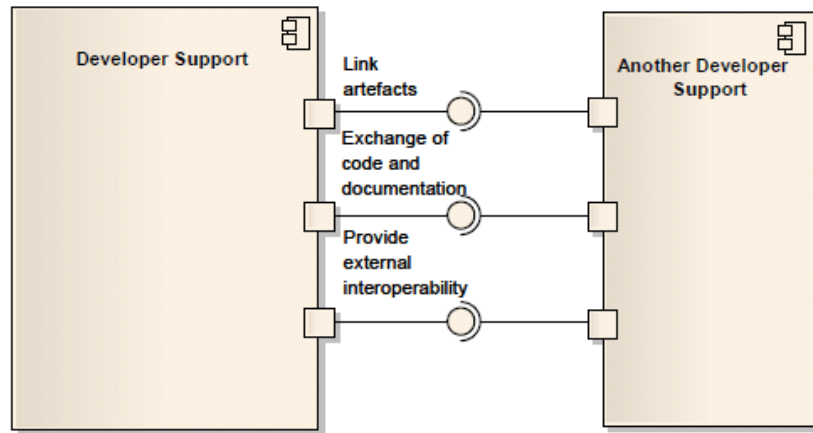


Figure 16: Peer services provided by Developer support provider

4 Component viewpoint

In this section we identify the main components of the platform, their responsibilities and the collaboration between them, and we outline some of the major information concepts that are used and manipulated by the platform.

4.1 System information model

As part of specifying any information system one needs to be specific about the information that will be processed by that system. This is the purpose of the System information model. An overview of the Information Model is depicted in the diagram in Figure 17.

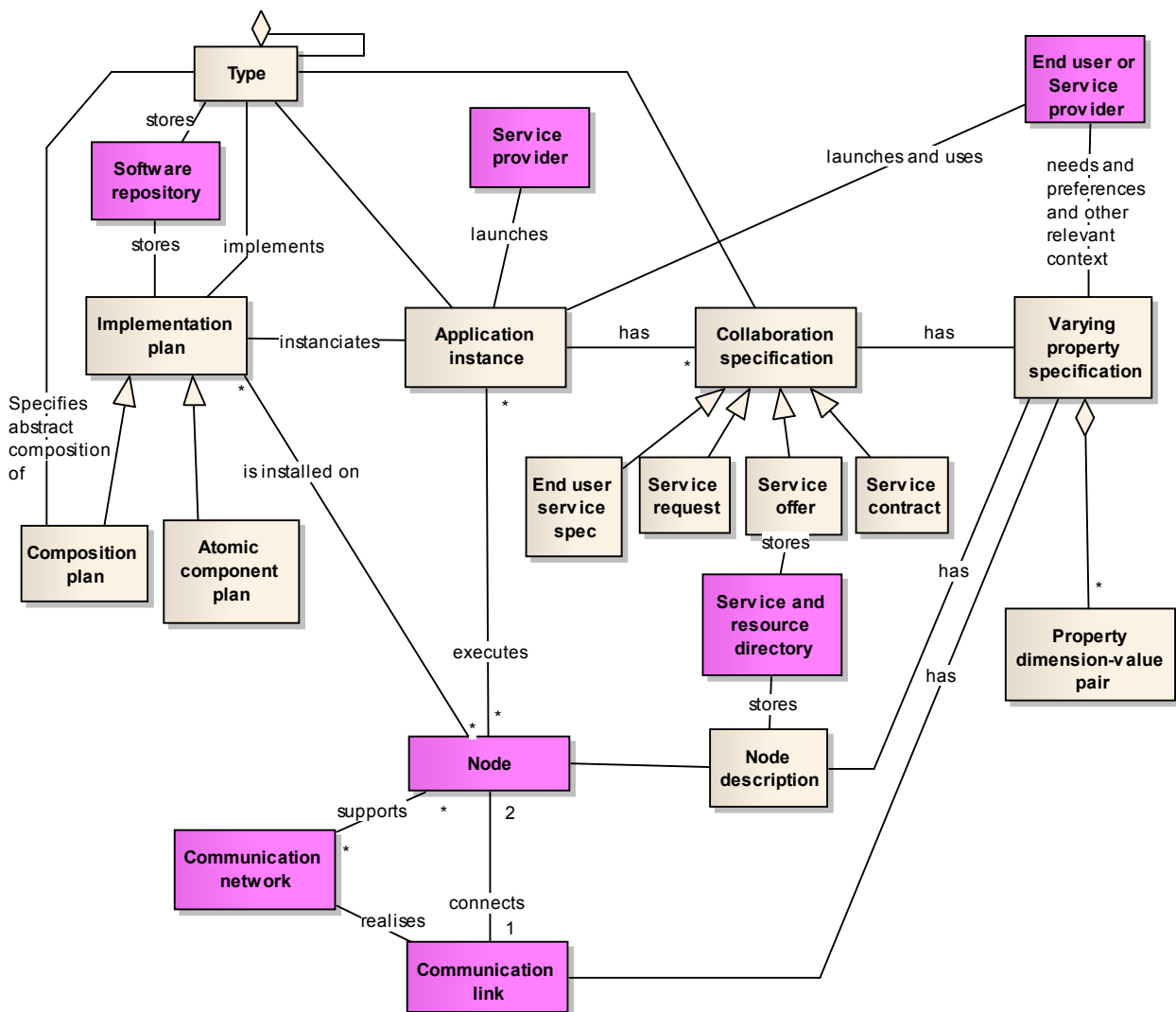


Figure 17: Major platform information concepts related to AAL service management

In addition to the information objects themselves we also included some real world object, which typically creates, uses or manipulates the information, in order to ease the understanding of the model. In the following we briefly explain the main concepts of the model.

A *type* specifies a piece of software in terms of its interfaces to its surroundings, i.e users and/or other pieces of software. A type may be defined as an aggregation of simpler types. An *implementation plan* describe the implementation of a piece of software, either in terms of program code (*atomic component plan*) or in terms of an abstract composition of types of smaller pieces (*composition plan*). Types and plans are stored in the Software repository.

An *application instance* is the instantiations of an implementation plan on a *node* (computer). Application instances are created by end users or service providers by launching a type. The platform then finds a suitable implementation plan matching the type installed on the node, and configures and instantiates an application instance fitting as far as possible the current context of user needs and preferences, the resources of the node on which it is executing, and services offered by other running application instances, either on the same node or on other nodes connected to the current node through a communication network it supports.

The collaboration specifications associated with an application instance describes its current and potential collaborations with the surroundings, in the form of either i) a user interface allowing a dialogue with an end user, or a service contract representing an ongoing collaboration with another application instance, or a service offer representing the capability and willingness to provide a service to another instance.

Varying property specifications specify properties of objects which vary between different instances of a type and which may vary over time for a given instance. They are used for several purposes: They describe QoS properties of offered services, or capabilities and capacities of nodes and links (*node descriptions*), or end user needs and preferences, or other relevant context information. They have the form of a set of property dimension – value pairs.

The capabilities and capacities of nodes and link, and service offers are stored in and retrieved from the *Service and resource directory*

The platform will monitor varying properties and adapt the running application instances to ensure best possible compliance with policies embedded in the implementation plans.

4.2 System Decomposition Model

Figure 18 shows an architectural sketch of the FI-M2M platform. The architecture is represented as a matrix, where the columns represent different dimensions of support provided by the platform, and the rows represent architectural layers.

There are 3 columns representing dimensions of support reflecting the stakeholder model in section 2.5.1. To the right there is *Runtime Support* in the form of middleware. To the left we have *Development support* in the form of modelling languages and tools supporting the development of context aware and adaptive applications and services. In the middle we have *Community support* which serves as a bridge between developers, supporting cooperative development between independent developer teams, and between developers and users, supporting the need.

There are 4 architectural layers. At the bottom there is the *Device Discovery and Communications* layer responsible for enabling devices to discover each other and establish communication links between them. Next there is the *Services and Resources* layer responsible for publishing services provided by a node and discovering services published by other nodes. Then we have the *Context Awareness and Adaptation* layer responsible for supporting context sensing and adaptation to context changes. At the top there is the *Functionality* layer responsible for implementing the functional behaviour of applications and services.

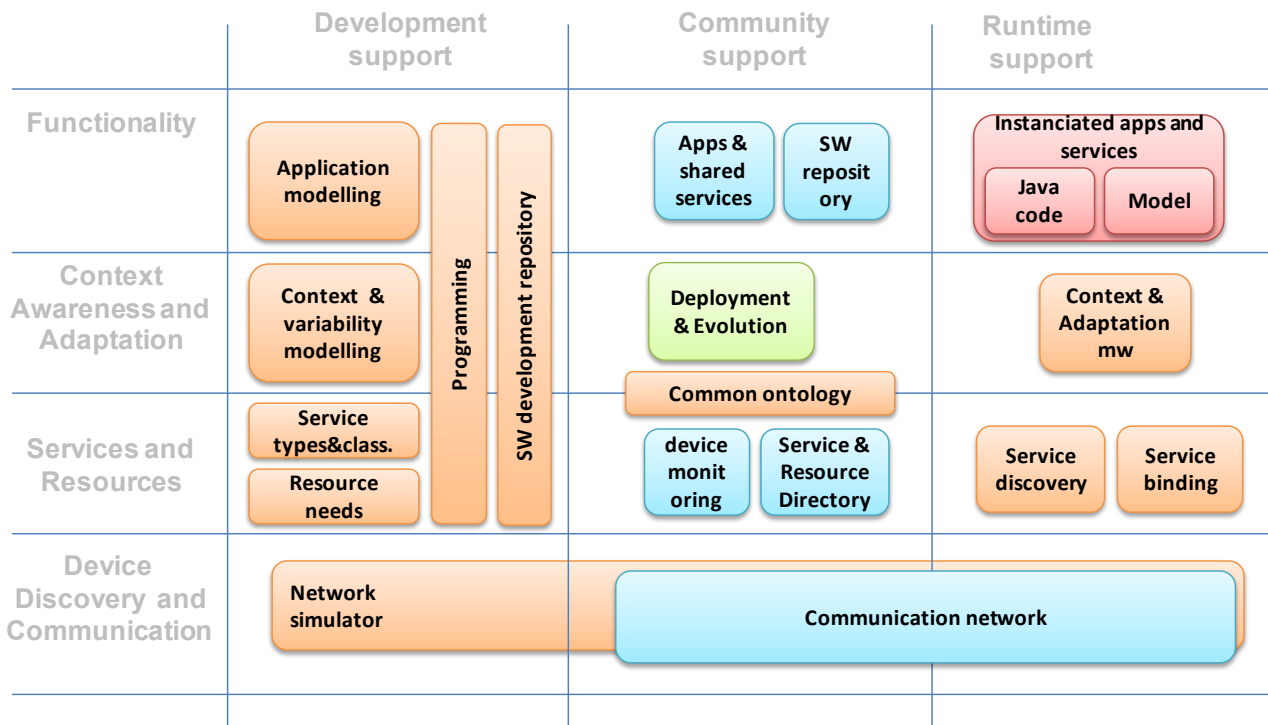


Figure 18 Architectural sketch of the FI-M2M tools and middleware platform

Note that this architecture is an initial sketch and may be subject to changes in the future as we progress further into the investigation of the demands on the technologies when they are integrated.

4.3 Component and Interface Specification Model

This section specifies the functionality and interfaces for the components identified in Figure 18.

The Context & Adaption mw component (Figure 19) is responsible for context monitoring and reasoning, adaptation reasoning and reconfiguration. It provides an interface *IConfigurable* that every reconfigurable component needs to implement. The *IConfigurable* interface provides methods to get and change the component state.

Once the application has been compiled and deployed into the middleware (mw) platform, the runtime support environment builds a runtime model of the application, by instantiating all the components needed for the execution and compose them according to the context and application models defined at design time. The Context & Adaptation mw takes care of monitoring the context-related data required or provided by the application. Upon the detection of context changes the Context & Adaptation mw, based on specific utility functions provided by the developer at modelling time, evaluates all the possible configurations and combinations for the application components and selects the composition providing the best utility (cf. Figure 27).

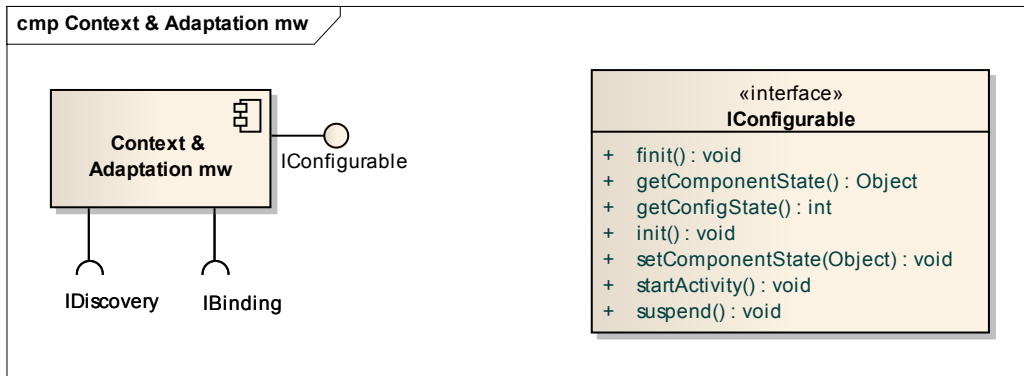


Figure 19 Interfaces for Conext & Adaptation mw component

The Service discovery component (Figure 20) is responsible for service publication and discovery. It can discover services via the *IDirectory* interface from the Service & Resource Directory component. It can discover services based on events from the Communication network (cf. Figure 26). This component provides the *IDiscovery* interface.

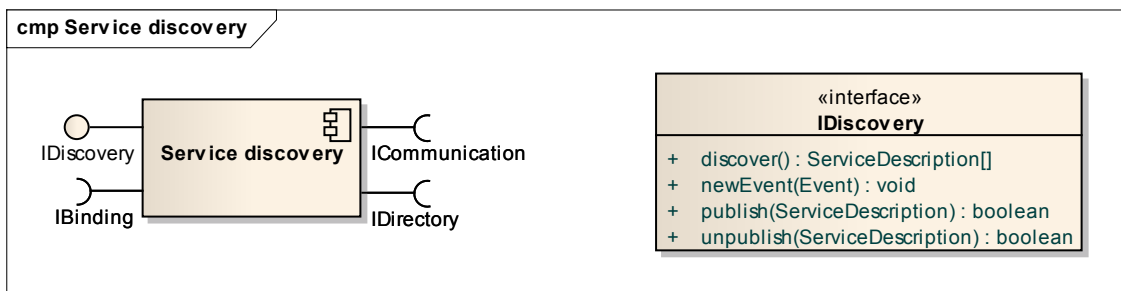


Figure 20 Interfaces for Service discovery component

The Service binding component (Figure 21) is responsible for communication with the remote service providers or consumers located in other nodes. It has two main functions: 1) It creates *proxies* (stubs and skeletons) for remote communication. When a service is discovered and imported, it generates a stub for invoking the remote service. When a service is published and exported, it generates a skeleton to accept method calls from remote clients. 2) The Service binding component generates *service plan variants* for discovered services. Such service plan variant contains information about service properties obtained from the service description as well as information about the link properties obtained via the *ICommunication* interface. Such information is used by the Context & Adaptation mw for adaptation planning to select remote services. In addition, the proxies are also associated with link properties, so that the adaptation mw can choose which link to use for the remote communication via proxies. The component provides *IBinding* interface for remote communication, and uses *ICommunication* interface to get link properties from the communication layer.

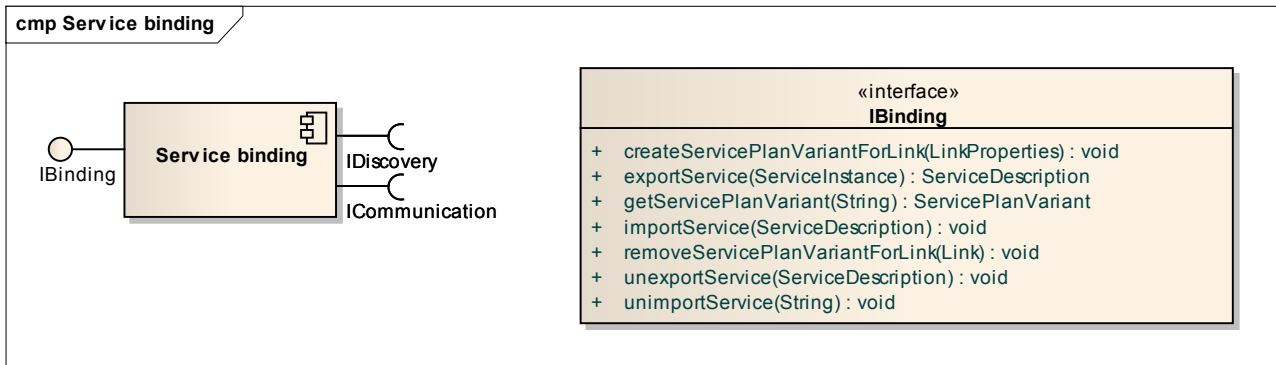


Figure 21 Interfaces for Service binding component

The Communication network module (Figure 22) provides information about the communication links and sends messages between nodes. It provides the *ICommunication* interface. Events about the Communication network are sent to the Service discovery component for proper handling. e.g., New Link event may trigger service discovery via the link if no links exist before with the remote node (cf. Figure 26). To realize cross-layer adaptation, the Communication network provides link properties to the application layer so that the link properties are incorporated in adaptation planning. Examples of link properties are delay, bitrates, transmission power, energy consumption and forwarding cost. The *forwarding cost* of a link is based on its resource usage (such as CPU, memory and battery level). For example, in an application, we may simply represent the forwarding cost of a multi-hop link as the average value of the battery levels for all the relay nodes.

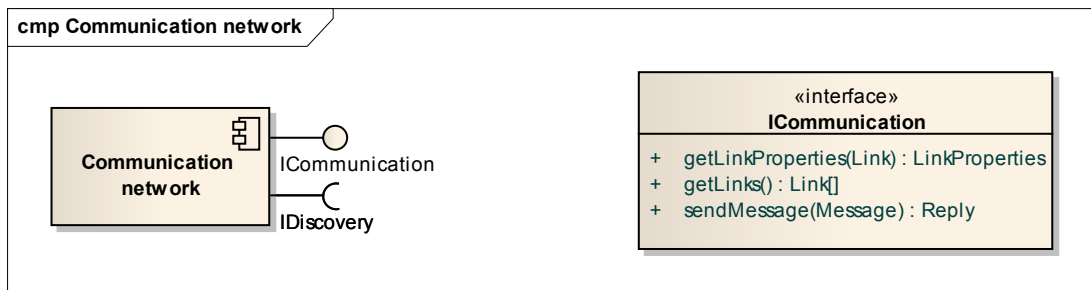


Figure 22 Interfaces for Communication network component

The Service & Resource Directory component (Figure 23) is responsible for resource and service registration and lookup. It provides *IDirectory* interface. This component and interface should be compatible with the IETF emerging standards, e.g., use CoRE for resource description, use publish/subscribe mechanism to disseminate events, access via HTTP REST and GUI interfaces.

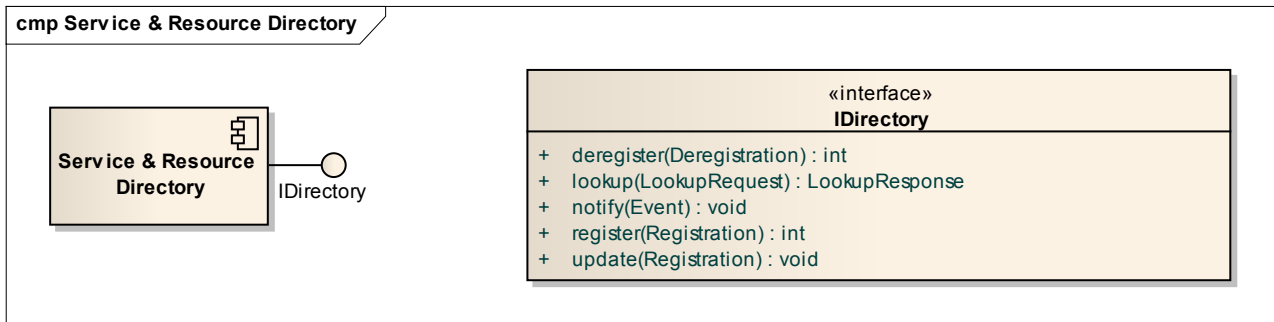


Figure 23 Interface for Service & Resource Directory component

The Device monitoring component (Figure 24) is responsible for monitoring of devices. It provides *IDevice* interface.

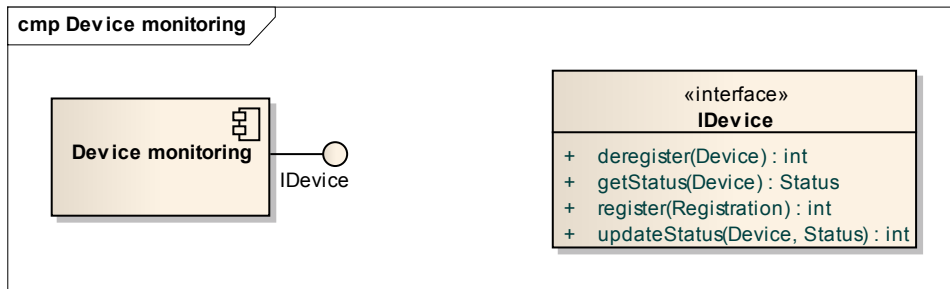


Figure 24 Interface for Device monitoring component

4.4 System Collaboration Model

4.4.1 M2M platform collaboration model

This section provides a high level view of interactions between different components.

Figure 25 represents the exportation and publication of a local service. The exportation and publication of a local service makes this service available in the network and other nodes might import it. The *ServiceInstance* reference includes some meta-information about the service to be exported as well as a reference to the service instance. The Service binding component uses this meta-information to create the skeleton (server-side service proxy). By default, an exported service is also published so that it is possible to discover this service from other nodes. The Service discovery component then sends the service description to all available nodes via the Communication network.

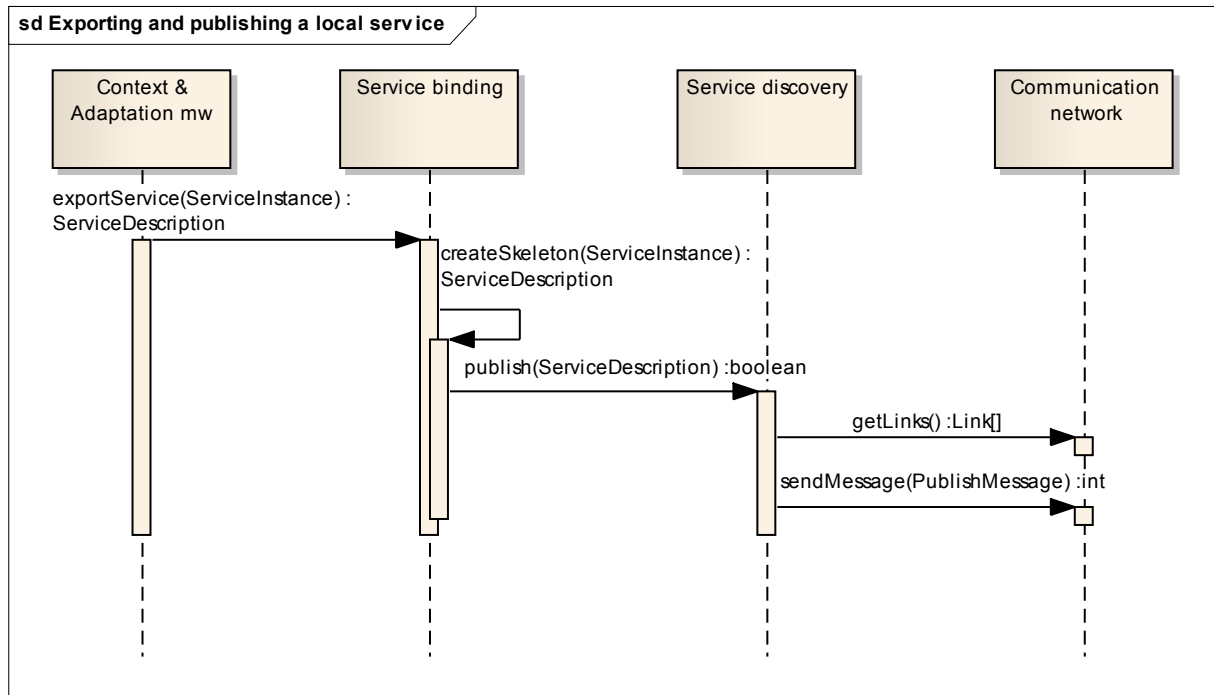


Figure 25 Sequence diagram for exporting and publishing a local service

Service discovery can be implemented by different approaches. Figure 26 shows the sequence diagram for an event driven service discovery. This discovery protocol takes account for the network topology changes based on the events from the Communication network. When a new link appears, middleware asks for the available services through this new link. If this is the first link between two nodes, a new service discovery request is sent. The discovered services are then imported, i.e., stubs (proxy at the client side) and service plan variants with the link properties are created. Otherwise, the middleware just creates a new service plan variant with link properties for each of the discovered services provided by the provider node. When a link disappears, all the service plan variants associated with this link will be deleted (i.e., unavailable from the link, not shown in the figure). In addition, when a provider updates service properties of a published service or retires a published service, it re-publishes or un-publishes the service (cf. Figure 25).

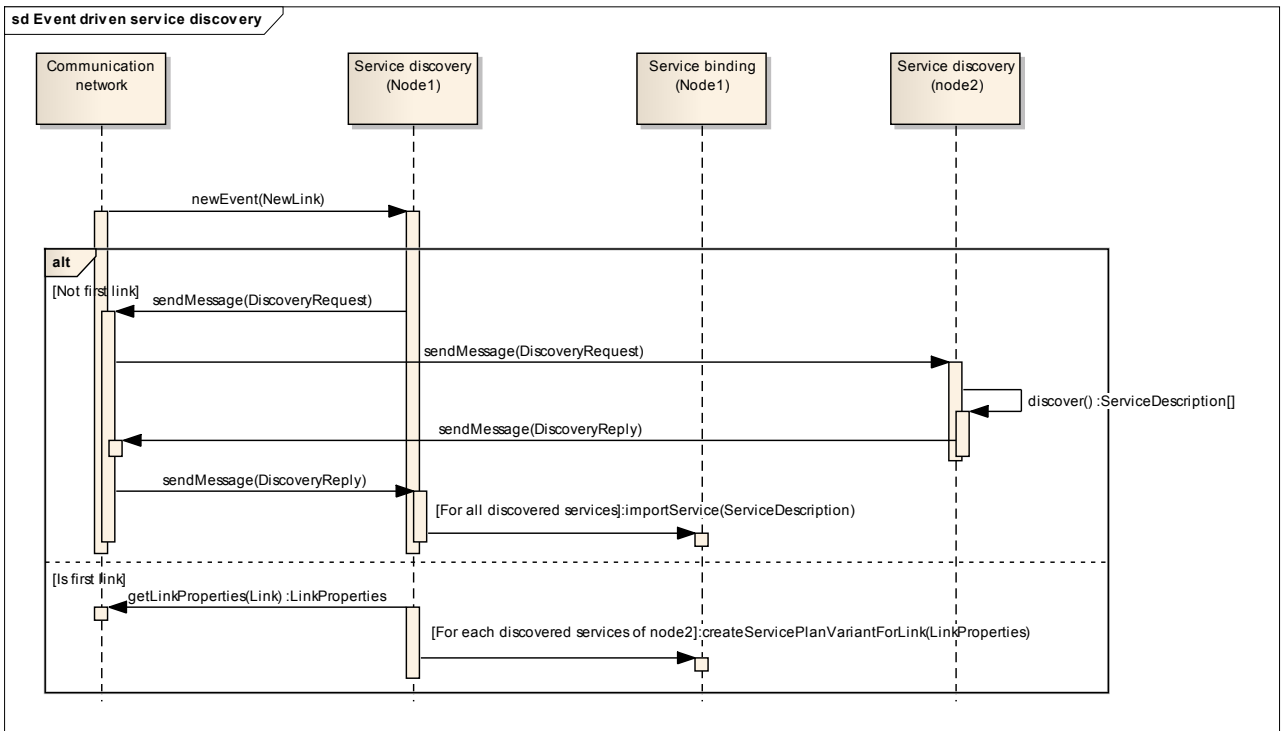


Figure 26 Sequence diagram for event driven service discovery

Figure 27 shows the adaptation process. The adaptation is triggered by context change events like new services or service disappearing. The Context & Adaption mw calculates the utility of a composition based on the service properties and link properties contained in the service plan variants, and selects the configuration that can optimize the utility.

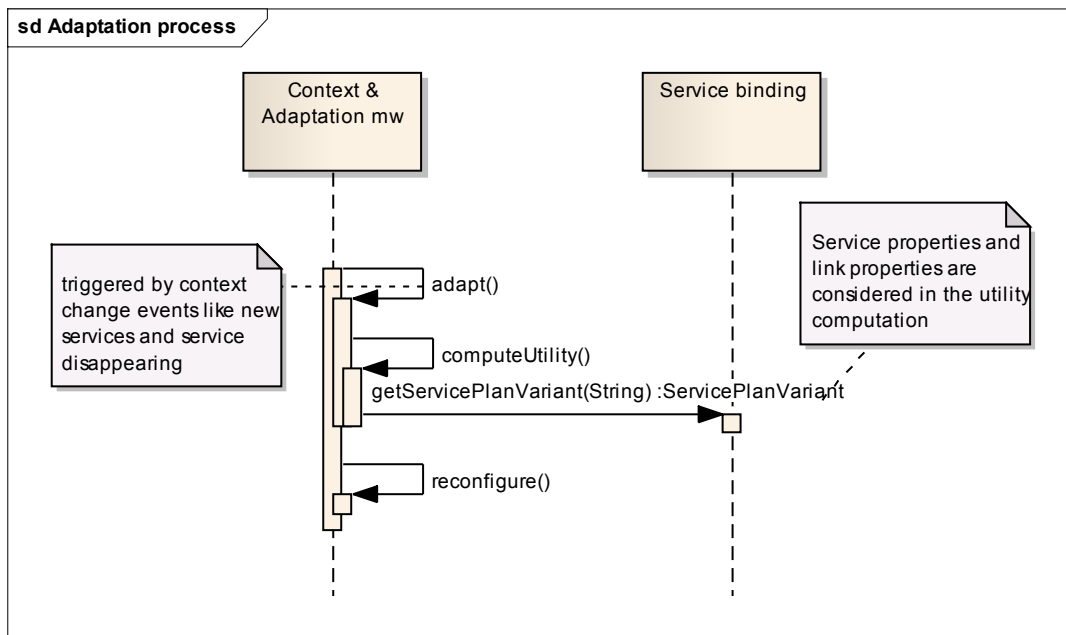


Figure 27 Sequence diagram for adaptation process

5 Realisation viewpoint

In this section we outline how the target platform can be realised by integrating and extending existing technologies, the kind of computers typically involved and how the main component will be deployed..

5.1 System Deployment Model

An overview of the kind of computers on which the platform and the systems running on it will typically be deployed, and where the different parts will typically be deployed is depicted in Figure 28.

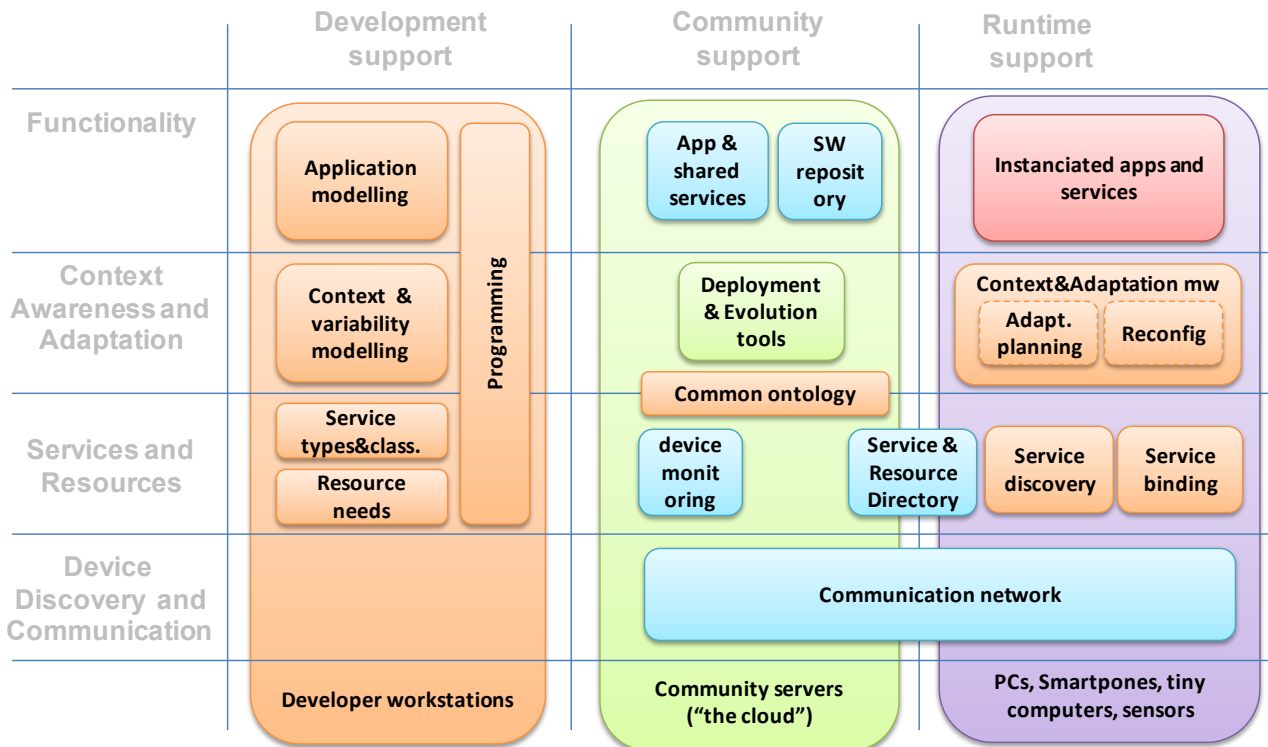


Figure 28 Deployment of the platform

Not all the runtime support components need to be present on all nodes. Some nodes may be too small for the full adaptation mw. An overview of the possible deployments is given in Table 12.

Table 12 Alternative deployments of the runtime support on different kinds of devices

	Registry and provision of services	Discovery and use of services	Re-configuration	Context monitoring	Adaptation planning
Master	x	x	x	x	x
Slave	x	x	x	(x)	
Static slave	x	x			
Tinyslave	x				

Figure 29 Example deployment of the AAL system described in the scenarios

To illustrate the deployment model we include an example deployment of a system built on the platform. The example is the AAL system described in the scenario section (section 2.2).

The arm wrist device has the full configuration of the runtime support and acts as a gateway and master node for the hearing aid and the heart sensor. It communicates with them through a Body Area Network, and with the outside world through WLAN.

The heartbeat sensor (labelled S in Figure 30) is implanted in Mary's body and is a very resource poor device where battery lifetime is a major concern, since when the battery is depleted, surgery is required to replace it. Therefore it uses a special protocol and is integrated through an edge component. The edge component will register it in the resource directory and bind it to the heartbeat monitoring service offered by heartbeat application in the arm wrist device.

The hearing aid is also a resource poor device, but it has the service discovery and binding components installed so when it is switched on, it will discover the resource directory and register itself and the services it offers. It also has the reconfiguration mw installed so its functionality can be dynamically extended to support the transmission of audio messages.

The door lock uses the BAN to detect if a valid key is close enough. This means about 1 meter.

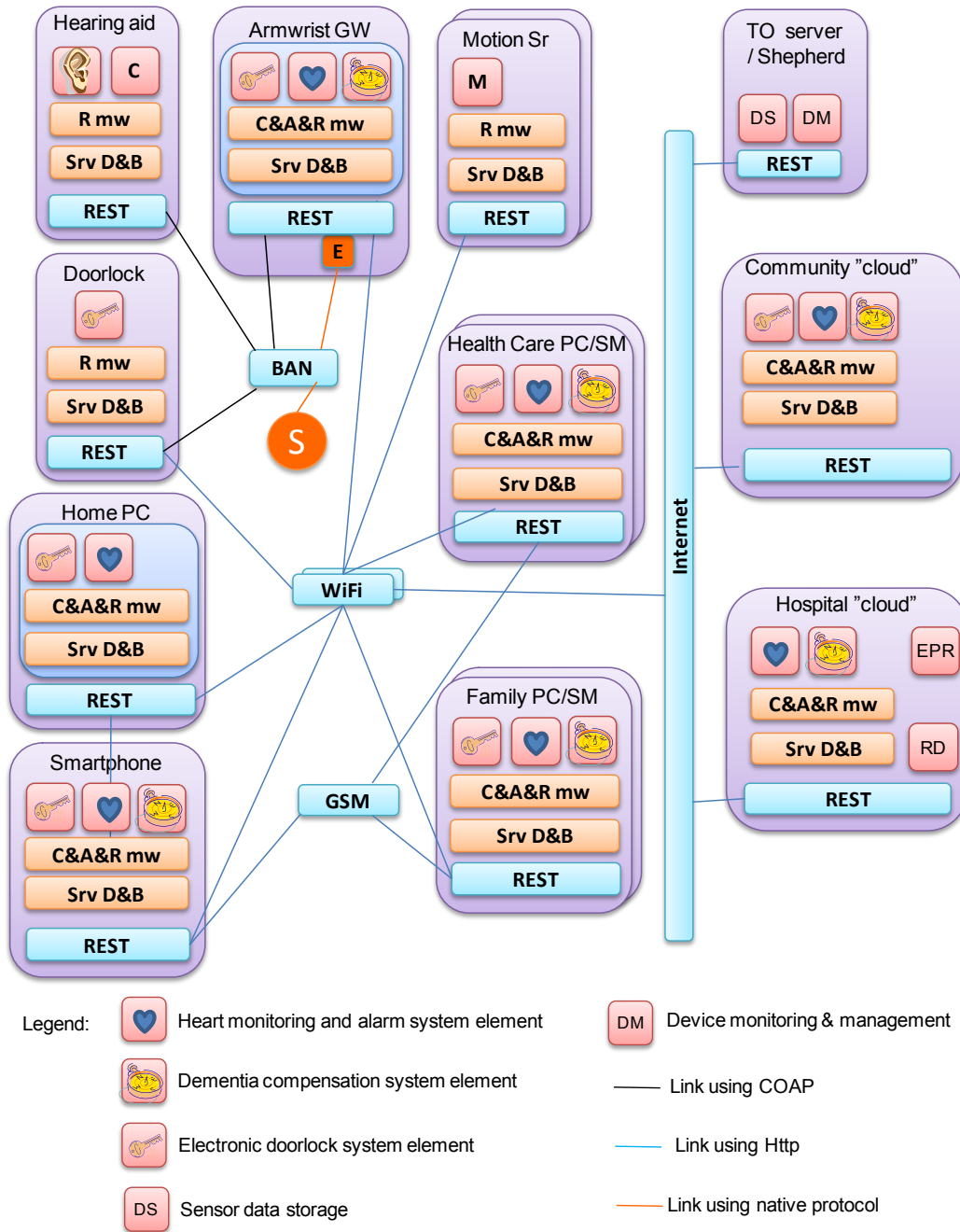


Figure 30 Example deployment

5.2 Technology Mapping Model

In this section we describe how we intend to realise the platform reusing and/or adapting available technologies. An overview of the technology mapping is presented in Figure 31. Explanation and justification is presented in the following subsection.

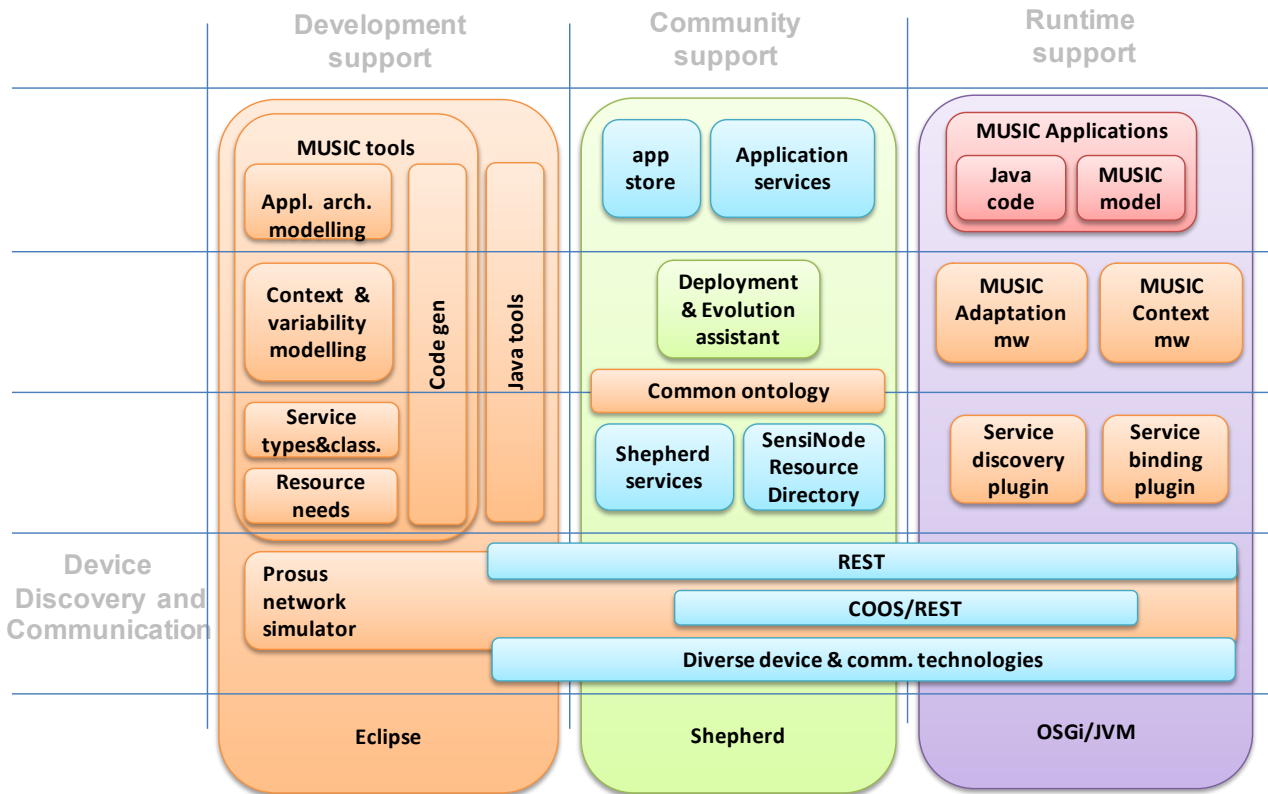


Figure 31 Technology mapping

5.2.1 Community support

5.2.1.1 Shepherd

Shepherd is a platform for hosting and provisioning managed services for M2M systems operated by Telenor Objects. This fits nicely in our architecture as a host for community services. The services provided by the Shepherd platform include sensor data storage and access, and device registration and management. Shepherd services are accessible through REST interfaces. Alternatively, both community services and application specific services may be hosted in "the cloud".

5.2.2 Development support

5.2.2.1 MUSIC modelling notation and tools

MUSIC was an IP project funded by the EU 7th framework programme. MUSIC has developed a comprehensive *software development framework*, which includes models, methods and tools that allow automation of the adaptation of the software to the varying user needs and operating conditions at runtime. An overview of the MUSIC framework is given in Figure 32. The development is alleviated by extensive support from the framework, including

- a modelling language which supports separation of concerns; and where self-adaptation and business logic are addressed separately to avoid the explosion in complexity;
- generic, reusable middleware components which automate context monitoring and management, and adaptation;
- tools which support the development of design models annotated with context and adaptation concerns and transform them into run-time knowledge available to the middleware.

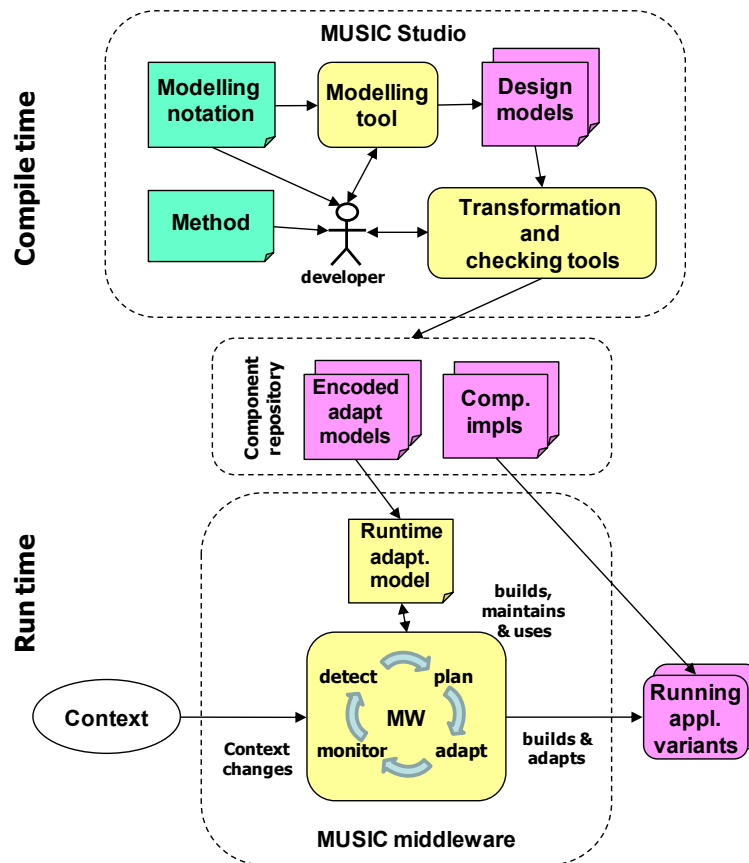


Figure 32: Overview of the MUSIC development framework

The MUSIC tools provide appropriate tool development support required to develop services and applications planned in the project. These services will need adaptability and context awareness. This is the main focus of, MUSIC, and therefore MUSIC was selected as a providing technology. The MUSIC platform was developed in a project where SINTEF was highly involved, thus SINTEF has the necessary competence with regards to integrating and tailoring the technology to the needs of this project.

The MUSIC modelling notation and tools covers application architecture modelling, context and variability modelling, service specification, and include code generation tools translating these models into a runtime representation (Java code) needed by the MUSIC middleware. The tools are embedded in Eclipse and together with standard Java development tool form a complete development environment.

5.2.3 Network simulator

The Prosus network simulator was originally designed to investigate the suitability of the MUSIC platform to support subsea sensor networks based on acoustic communication, but basically it is suited to simulate

heterogeneous and dynamic networks in general, and thus may serve to allow testing of complex heterogeneous distributed systems in simulated environments, to verify both their functional and adaptive behaviour before deployment on real networks and devices. It will be adopted as a network simulator component. The network simulator is described in Appendix C.

5.2.4 Context and adaptation middleware

5.2.4.1 MUSIC Middleware

The MUSIC middleware implements a central control loop which monitors the relevant context. When significant changes are detected, it triggers a planning process to decide if adaptation is necessary. When it this is the case the planning process finds a new configuration that fits the current context better than the one that is currently running, and triggers the adaptation of the running application. To do this the middleware relies on an annotated QoS aware architecture model of the application available at runtime, which specifies its adaptation capabilities and its dependencies on context information. The planning process evaluates the utility of alternative configurations, selects the most suitable one (i.e. the one with the highest utility for the current context which does not violate any resource constraints) for the current context and adapts the application accordingly. A number of different adaptation mechanisms are supported, including parameter setting, component replacement and redeployment (Geihs, et al. 2009) (Floch 2006), and service rebinding (Rouvoy 2008).

The MUSIC middleware is available as an open source project under LGPL license [21]. The MUSIC middleware can be applied for the development of adaptive applications for any type of target devices such as PCs, Laptops or mobile phones, which support at least Java 1.4 and must have OSGi available at run-time. The MUSIC middleware platform utilizes the dynamic deployment and bundle management functionalities provided by OSGi as a basis for dynamic reconfiguration, and supports incremental evolution of running applications by deployment and retirement of bundles containing MUSIC components and compositions plans.

MUSIC has an extensible architecture based on plug-ins, which is ideal for integration with other technologies. For example, plug-ins can be implemented to support different service discovery protocols or realize different reasoning algorithms.

A remarkable feature of the MUSIC middleware is the configurability of the middleware itself. MUSIC has separate bundles for adaptation reasoning (i.e., Adaptation Reasoner) and configuration execution (i.e., Configurator). Some bundles are also optional, e.g., SLA bundles can be omitted if services without SLA capabilities are preferred. MUSIC allows multiple configurations of the MUSIC middleware by selecting which bundles to be deployed (i.e., depending on the operating system, the device hardware or the role of the node). A MUSIC node can take different roles, such as MASTER, SLAVE, REASONER and Configurator. A MASTER node is responsible for coordinating the whole process of adaptation in the adaptation domain and needs to run a full configuration with both an Adaptation Reasoner and a Configurator. A SLAVE node only provides resources to the adaptation domain (e.g., to host some resource demanding components), thus does not run an Adaptation Reasoner or a Configurator. A (remote) REASONER node performs adaptation reasoning and hosts only the Adaptation Reasoner. For a resource-constraint node, it is possible to run a minimal configuration, i.e., only a Configurator, and perform reconfiguration scripts decided by other REASONER nodes.

The DIVA project has also provided notations and tools for modelling context aware and adaptive systems. DIVA uses aspects as the primary variability mechanism and may complement the MUSIC platform.

More elaborate description of MUSIC is available in Appendix A and B. In conclusion we consider that MUSIC naturally supports the needs for adaptation and evolution exhibited by the scenarios in section 2.2.

5.2.5 Service discovery

Missing from MUSIC is a service registry supporting system wide dynamic publishing and discovery of services. In the MUSIC trials SLP (in P2P mode without the use of registries) was used, but this only works in a limited network context. Also in SLP, discovery is based on a polling approach, meaning that client nodes will regularly issue multicast requests checking the availability of service providers in the environment. This is not a good approach for tiny battery powered devices.

5.2.5.1 SensiNode NanoService Platform

A candidate technology was developed in SENSEI, an EU project that Telenor Corporate Development participated in, namely the SENSEI Resource Directory. It provides a global registry through the federation of multiple local registries and was implemented using COOS (see section 5.2.6.3) for the federation protocol. This is one of the most promising results that came out of the project and the project initiated an effort to standardise the interface to the resource directory through IETF. This effort is continued after the end of the project and appears to be successful. At the time of writing a draft specification has been submitted to IETF (see <http://tools.ietf.org/id/draft-shelby-core-resource-directory-00.txt>)

The implementation done by the SENSEI project is a bit immature. However, SensiNode, one of the SENSEI partners and main supporters of the standardisation effort, is in the process of implementing support for this interface as part of their NanoServices platform⁵.

Our investigation of the proposed standard interface for the resource directory, indicate that this may serve our needs regarding service discovery. However, at the time of writing, the availability and implementation status is only partly known, so this must be considered as a tentative choice.

5.2.6 Communication layer

5.2.6.1 REST

We will assume a REST interface to the communications layer, ensuring compatibility with HTTP and COAP protocols. REST (REpresentational State Transfer) [1] is a resource-oriented software architectural style for building Internet-scale distributed applications and systems. It applies constraints like uniform interfaces, self-descriptive data, stateless communication, cache components, on a client-server architecture. The RESTful architecture is built on principles for encoding (i.e. representation), addressing and accessing resources. Resources in REST are encoded using different representations, such as XML, JSON, YAML, addressed via URI and accessed using Internet-based protocols such as HTTP. REST has been a promising technology for the integration of heterogeneous computational entities and has been applied for sensors and sensor networks. For example, TinyREST⁶ uses HTTP-based REST architecture to obtain and change the state of sensors.

5.2.6.2 COAP

Today web services are the standard way of communicating over the internet for an application. CoRE (Constrained RESTful Environments)[3] working group in IETF[4] are working on defining a RESTful (see

⁵ <http://www.sensinode.com/media/flyers/sensinode-nsp-flyer-web.pdf>

⁶ <http://www.sics.se/realwsn05/papers/luckenbach05tinyrest.pdf>

section 2) protocol that provides such capabilities on computationally constrained resources. Typically embedded sensor networks are made available as services through such constrained devices. CoRE's proposal is the Constrained Application Protocol (CoAP)[5] that would enable these constrained devices to communicate easily through web services. The protocol aims to provide very low overhead and simplicity to satisfy the requirements of the target audience of devices. As such the protocol benefits M2M and sensor networks in general. In our demonstrator we will use the CoAP implementation provided by Sensinode as part of their Nano services platform.

5.2.6.3 COOS

The COOS message bus⁷ (Figure 33) has been used in the Shepherd platform and many relevant devices has been interfaced to it. Therefore we plan to also support communication via this message bus. COOS has two important interfaces the Consumer interface and the Exchange interface. The Consumer interface is implemented by every plugin to the COOS bus and will be called by the bus when an incoming message arrives for the UUID address of that plugin. The Exchange interface is used by the COOS plugins to send messages to the COOS UUID specified along with the message.

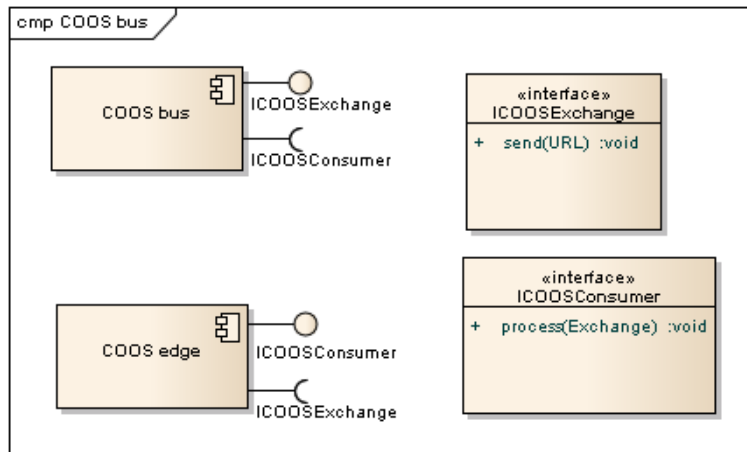


Figure 33 COOS message bus

Translation between the M2M platform and COOS plugins (Figure 34) is done via a translator edge that at all times should be connected to a COOS message bus to support the M2M platform service bindings. The COOS plugin will be published in the service discovery component (Figure 20) with a URI that contains the RESTlet URL and the COOS UUID. This way the service-binding component (Figure 21) will send any message directed at the plugin to the RESTlet URL hosted by the translator edge with the COOS UUID and message as payload. The translator edge will then relay this message to the specified COOS UUID and wait a given amount of time for any reply. If the service (provided by the coos edge) is available on the COOS bus and replies this reply will be given as a reply to the HTTP query to the RESTlet call.

As mentioned in the previous paragraph any coos plugin supported in this M2M platform has to publish itself to the discovery component, this will be done in the same manner as depicted in the “alt if service available” part of Figure 29. The translator edge will then recognize the payload as a message publishing the service, and relay the payload to the service discovery component (with the addition of the COOS UUID)

⁷ <https://telenorobjects.onjira.com/wiki/display/coos/Home>

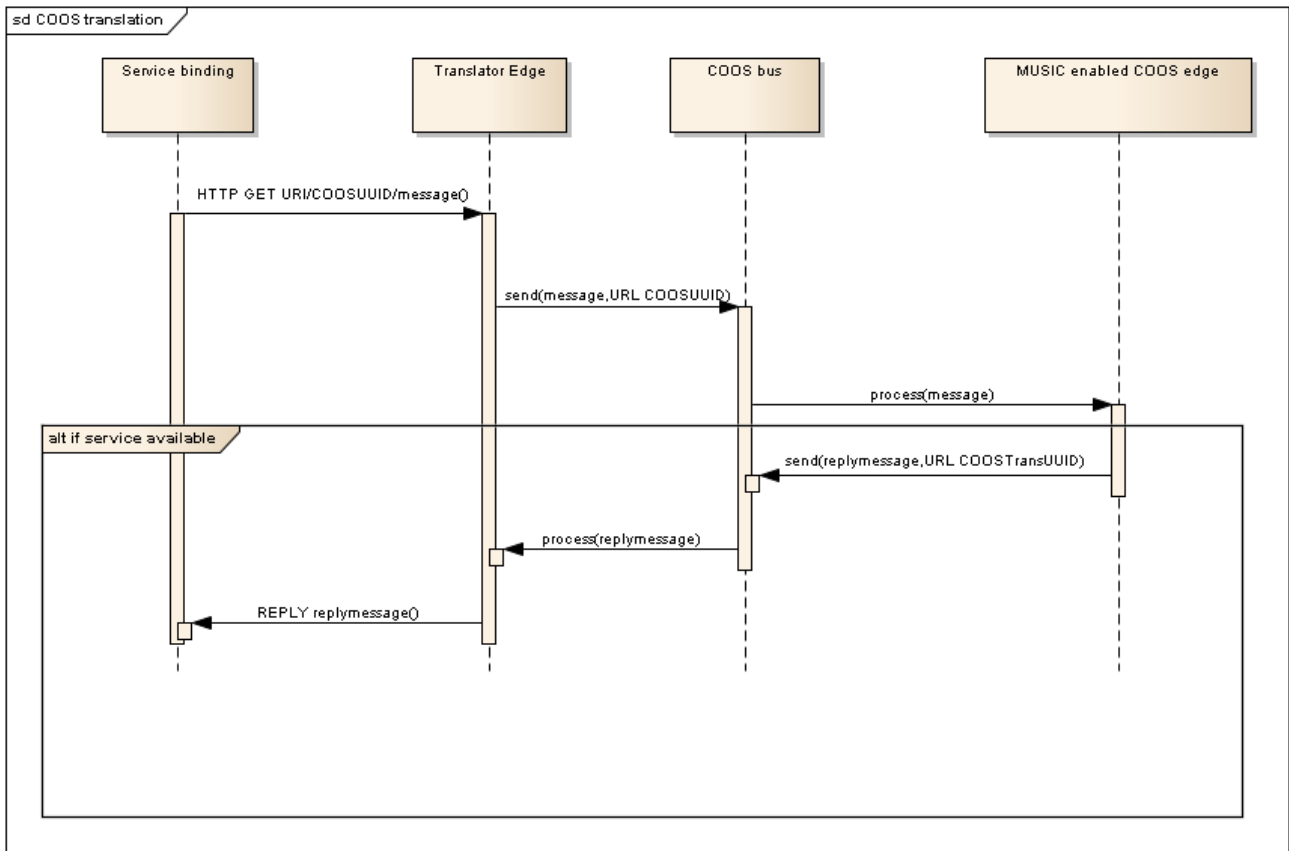


Figure 34 Translation from M2M platform to COOS services.

6 Conclusion

In this document we have documented the architectural analysis and design phase of the first step of the FI-M2M project. The architecture defines a tool and middleware platform aiming to demonstrate the benefits of context awareness and self-adaptation to simplify the deployment and evolution of M2M systems. The document covers scenario and stakeholder analysis, requirements definition, high level design, and technology mapping. AAL has been used as an example application domain to investigate the needs related to the deployment and evolution of M2M systems.

In this first step we have primarily focused on the evolution of deployed system instances by the addition or replacement of available artefacts, and not so much on the continued development of the body of artefacts, which is meant to be addressed in the next step. However, the scenario and stakeholder analysis, and the requirements also to some degree covers the latter aspect.



Technology for a better society
www.sintef.no