Title	**Executive Summary D2.2: Middleware v1 developed**
Work package	WP2: Solution Development and Iterative Adaptions
Date	02.01.2018
Author	Author of the deliverable: Lennart Köster, University of Innsbruck Author of the executive summary: Maximilian Bernard, University of Innsbruck

Consortium Partners

universität innsbruck · SFI · rogvis · Österreichisches Rotes Kreuz Innsbruck · eurac research · Product Development University of Antwerp · SIS CONSULTING · Villa Melitta · fifthplay

CONTENT

This is an executive summary of deliverable 2.2 of the gAALaxy project.

1. INTRODUCTION

Deliverable 2.2. is of type *prototype,* but in order to explain the development process behind the V1 middleware backend, I'd like to add a written document to the released prototype. The prototype itself was released on 23rd of July `17 and was not delayed by this document.
For the first pilot testing phase, the developed middleware is connecting to existing interfaces of the included products and is currently not using any industry standards or available middleware platforms but was rather developed from scratch and thus only includes the first phase's products.
The goal of this *proof-of-concept middleware* (PoC) is to gain knowledge about the bundled products and the implemented use cases from WP1. The lessons learned will then be employed into the V2 middleware which should be built on industry standards, if the feedback, requirements and lessons learned demand for that.

1.1. TECH STACK

The PoC is developed in groovy and static-compiled to Java 8. The nature of a proof-of-concept demanded for an adaptable and quick-to-build, quick-to-deploy tech stack.
The PoC V1 is run as a command line tool using the command line interface as user interface.
User and configuration data is stored as .json configuration files and can be changed/added at runtime – the files will be parsed and added to the gAALaxy middleware.
For the PoC V1, no database and no incoming interfaces where needed since user interaction would be through the command line interface only and configuration would be file-based on the same host.
The PoC V1.5-MTR will be configured through a cloud-based UI which accesses REST API endpoints on the middleware to configure the middleware on a per-user basis.

1.2. ARCHITECTURE

The V1 gAALaxy bundle consists of components which need to be installed 'on premise', e.g. fearless sensors, 2PCS antenna, any smart home component, smart home base station etc. and also components and cloud applications running on the internet, e.g. fearless management platform, smart home management platform, 2PCS management platform, various licencing servers etc.
Therefore, we needed an architecture that allows for on premise devices and cloud services to communicate.
Further, many interfaces between some of the products (mainly 2PCS in combination with another product) have been implemented already and were evaluated as stable during WP2's lab testing. That meant we would not need to re-build those interfaces for PoC V1.

1.3. USE CASES

The Use Cases defined as requirements in WP1 have been modelled as state machines which can be run on a per-user basis.
The gAALaxy PoC V1 loads up use case configuration files per user on start-up (and refreshes the configuration when the configuration file changes), creates new state machine instances and runs each users' use cases until their configuration gets deleted. In PoC V1, new use cases need to be implemented in the middleware, which then needs to be redeployed. This obviously only makes sense for the PoC V1 testing where the scope and requirements of use cases is clearly defined beforehand.

Those use cases show perfectly how different systems (e.g. fearless *OR* 2PCS can be used as emergency devices) can be integrated with smart home systems (e.g. fifthplay *OR* Innogy smarthome) to create meaningful use cases for the primary user. The goal of PoC V1 is to confirm these assumptions through pilot testing.

2. LESSONS LEARNED

- Design processes/functionalities with 'per household implementation' in mind:
  The initial architecture and processes between gAALaxy and both smart home systems was designed around outgoing emails from smart home systems which would have to be parsed by gAALaxy in order to be notified of smart home events which could be relevant for gAALaxy use cases. However, once implemented in the PoC, when we tested the PoC in our lab, we quickly identified that not only are those email triggers another source of failure (each and every single one had to be created by hand on the smart home systems) but also meant that implementation of a gAALaxy box took considerably longer than we wished for. Therefore, we threw out that mechanism and refactored our smart home integration on the middleware, costing about another week of development time. For V2 we should already design the processes with the implementation time in mind.
- Plan for longer deployment period:
  We did not account for an unknown hosting environment and thus did not plan for more than 1 sprint for deployment after release. For the next gAALaxy middleware, especially when considering that we need to run more components/products with new data connections, we should plan for longer periods – e.g. 2 weeks for deployment after release.
- Include more time for additional bug fixing between release, deployment and production use:
  When we first encountered various challenges on the production system, it took longer than expected to initially identify the culprits and research how to fix those. For V2 need to get accustomed a lot earlier to the production environment and make sure that gAALaxy V2 works under any foreseeable circumstances with little to no customization/refactoring.