H2020-SC6-GOVERNANCE-2018-2019-2020

DT-GOVERNANCE-05-2018-2019-2020

ACROSS
Towards user journeys
for the delivery of cross-border services
ensuring data sovereignty

D4.9 User Support Tools – Final

Project Reference No	959157 — ACROSS — H2020-SC6-GOVERNANCE-2018-2019-2020
Deliverable	D4.9 User Support Tools – Final
Work package	WP4 - ACROSS Modules Set-Up
Nature	Other
Dissemination Level	Public
Date	31.07.2023
Status	Final V1.0
Editor(s)	Thilo Ernst (FRH)
Contributor(s)	Anna Opaska (FRH), Steven Schulz (FRH), Claudia Tulasch (FRH), Karl Blumenthal (FRH), Mustafa Can (FRH), Ekkart Kleinod (FRH), Yevheniia Strilets (FRH), Anestis Sidiropoulos (formerly ATC), Enrique Areizaga (TEC), Vincenzo Savarino (ENG)
Reviewer(s)	Vincenzo Savarino (ENG), Enrique Areizaga (TEC), Elena Chrysikou (ATC)
Document description	This report focuses on the ACROSS User Support Tools *User Journey Modelling Tool (UJMT)* and *Virtual Assistant (VA)*. It documents the requirements, design, description of modules, and description of APIs applying to these components. It also describes the implementation of the current release ("Final version") of the software components. Finally, it contains a description of the relevant baseline technologies being used for the implementation of the user support tools.

About

The project is co-funded by the European Commission's Horizon 2020 research and innovation framework programme. Spanning through three years, ACROSS consists of a consortium of 10 partners from 7 countries: Athens Technology Center (coordinator), Tecnalia, Dataport, Engineering, Fraunhofer, GRNET, TimeLex, The Lisbon Council, Waag and VARAM. The project kicked off its activities in February 2021, with an energising online meeting, where all partners took the floor to present their plans to make the project a great success.

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
V0.1	07/07/2023	Initial draft	FRH
V0.2	10/07/2023	Second draft	FRH
V0.3	26/07/2023	Pre-final Draft	FRH
V0.4	28/07/2023	Internal Review	ATC
V1.0	31/07/2023	Final Version (considering the reviewer comments)	FRH

Executive Summary

The main objective of the ACROSS project is to provide the means (tools, methods and techniques) to enable user-centric design and implementation of interoperable cross-border (digital) public services compliant with the current European regulations where the private sector can also interconnect their services while ensuring the data sovereignty of the citizens, who can set the privacy level that will allow the public and private sector to access to their data based on their requirements.

This deliverable "D4.9 User Support Tools – Final" documents the design and essential implementation aspects of two software components developed by Fraunhofer FOKUS(FRH) within Work Package 4 "ACROSS Modules Set-up" and concretely in Task T4.3 "User support tools implementation" of the ACROSS project: The User Journey Modelling Tool, and the Virtual Assistant, including the Final Release of the user support tools (software components). (For the earlier Proof-of-Concept Prototypes, please refer to "D4.7 User Support Tools – Initial" [1] and "D4.8 User Support Tools – Intermediate" [2].)

The focus of the User Journey Modelling Tool (UJMT) is on *modellers* - expert users who think about and design *user journeys*, i.e., among other aspects, plan in which combination and ordering end-users (citizens) will need to access the services offered through the ACROSS platform in a specific situation (or use-case). Modellers have expert knowledge about the available services, but they are not expected to have in-depth IT expertise. Using the UJMT, a modeller will be able to interactively create graphical models of the "service workflows" the end-users will need to go through in each user journey. Each modelled user journey-specific service workflow is converted into a suitable machine-readable presentation and is then handed over to the User Journey Service Engine which, in collaboration with other ACROSS subsystems, enables the orchestration and actual execution of the specified service workflow. Eventually, the workflow can be selected, started and used through the ACROSS Web/Mobile App (and the Virtual Assistant, see below) by the end-user citizen, exactly according to the User Journey model created at the beginning. The UJMT is created based on a carefully selected and powerful open-source package which is substantially modified and extended, and then integrated into the service-oriented ACROSS architecture.

The Virtual Assistant's focus, on the other hand, is on the *end user/citizen* - its purpose is to enable superior ease of use and accessibility (in particular, for users with disabilities) for the user-exposed parts of the ACROSS system - concretely, the ACROSS Web and Mobile App. This goal is pursued by offering dedicated conversational interfaces (i.e., multi-lingual textual chat-bot and speech interfaces) for controlling user interface features of the ACROSS Web and Mobile App by natural language. The Virtual Assistant is implemented by building upon modern web standards and open-source software and integrating it both with the other ACROSS subsystems and with background technology from Fraunhofer

FOKUS using an innovative, minimally invasive integration approach, utilizing a state-of the art service-based architecture.

Compared to the “Initial” version of the User Support Tools (documented in [1]), all User Support Tools have a substantially extended feature set (enabled, to a substantial degree, by the implementation of new subcomponents) that has been closely aligned with the ongoing developments within the use-case focused ACROSS work packages WP2 and WP6, thanks to an improved agile process for gathering and tracking requirements which has been deployed throughout the project in this project phase. Also, the integration of the User Support Tools with the ACROSS platform and the other ACROSS components has been strengthened considerably. This “Final” version completes the User Support Tools with features and improvements that have been requested by the Use Case partners and with optimizations the opportunity for which arose during our development work.

Table of Contents

List of Figures

List of Tables

List of Terms and Abbreviations

Abbreviation	Definition
BPMN	Business Process Modelling and Notation
CPSV	Core Public Service Vocabulary
CPSV-AP	Core Public Service Vocabulary Application Profile
DoA	Description of the Action
EIF	European Interoperability Framework
FDM	FISA Dialog Manager
FISA	Fraunhofer FOKUS Intelligent Speech Assistant
ICT	Information and Communications Technologies
KPI	Key Performance Indicator
MO	Measurable Outcome
OOP	Once-only principle
PA	Public Administration
PoC	Proof of Concept

Abbreviation	Definition
SDG	Single Digital Gateway
UIA	User Interface Adapter
UIC	User Interface Connector
UJM	User Journey Model
UJMT	User Journey Modelling Tool
UJSE	User Journey Service Engine
UJWD	User Journey Workflow Description

1 User Support Tools - Introduction and Overview

1.1 Common ACROSS Context

The ACROSS project envisions two User Support Tools, the User Journey Modelling Tool (UJMT) and the Virtual Assistant (VA). Both are motivated from main objectives / Measurable Outcomes (MOs) as defined within the Description of the Action (DoA) of the ACROSS project, and their detailed success criteria (Key Performance Indicators (KPIs)):

- MO1.1: A User journey methodology, approach **and supporting tool** to define and model user-centric digital public services, complemented with an example, namely, the application to the cross-border mobility life event. […] KPI1.1.2: An **online tool for modelling user journeys** to define the interaction of the user with the digital public services and their orchestration.

- MO1.3: **Multi-lingual Virtual Assistant providing speech and textual chat interfaces** guiding the user in the user journey of the service. The assistant will provide superior accessibility (in particular, for users with disabilities), by offering multi-lingual textual chat-bot and speech interfaces to be integrated with the ACROSS Platform. Success criteria: KPI1.3.1: 100% of the requirements and functionalities specified are implemented. KPI1.3.2: It is able to support in all the steps of the user journey.

These and the other MOs have their roots in European legislations and initiatives such as Single Digital Gateway (SDG), Once-Only principle (OOP) and European Interoperability Framework (EIF). MO1.3 also is strongly motivated by the European Accessibility Act, which has the primary goal of benefitting persons with disabilities and elderly people by providing more accessible products and services in the European market.

1.2 Methodology and Structure of the Deliverable

As the two User Support Tools have very different purposes and roles within ACROSS, and collaborations to other ACROSS components, they will be described independently, each within its own main chapter. However, a common description format will be used, covering

- the Objectives pursued by each tool together with the scope in which the tool is relevant,
- the context motivating the creation and influencing the design and implementation of each tool (both on European level and specifically within ACROSS),
- the specific requirements that have been gathered for the tool and that steer its design and implementation

- the design of the tool (including architecture, sub-modules, interfaces, and collaborations)
- a description of the implemented features delivered at the second milestone
- and a description of the baseline technologies used in the implementation of the tool.

2 User Journey Modelling Tool (UJMT)

2.1 UJMT - Objectives and Scope

The User Journey Modelling Tool (UJMT) is an online supporting tool for defining and modelling user-centric digital public and private services. In the beginning of the project, the following main objectives of the UJMT were defined:

- Provision of the functionalities that are necessary for modelling user interactions with digital public and private services as user journeys
- Provision of corresponding machine-readable descriptions for the service orchestration to the User Journey Service Engine (UJSE) [3, 4]
- Integration of available public administration (PA) and third-party private service information from the Service Catalogue [3, 4]

As the project progressed, the UJMT evolved towards these objectives, with new requirements arising from the technical partners and/or pilot partners in each phase of the project. This resulted in an expansion of the UJMT with further features towards additional objectives, such as the management of User Journey Models (UJMs) in a central storage and the provision of additional User Journey data to the Citizen Frontend.

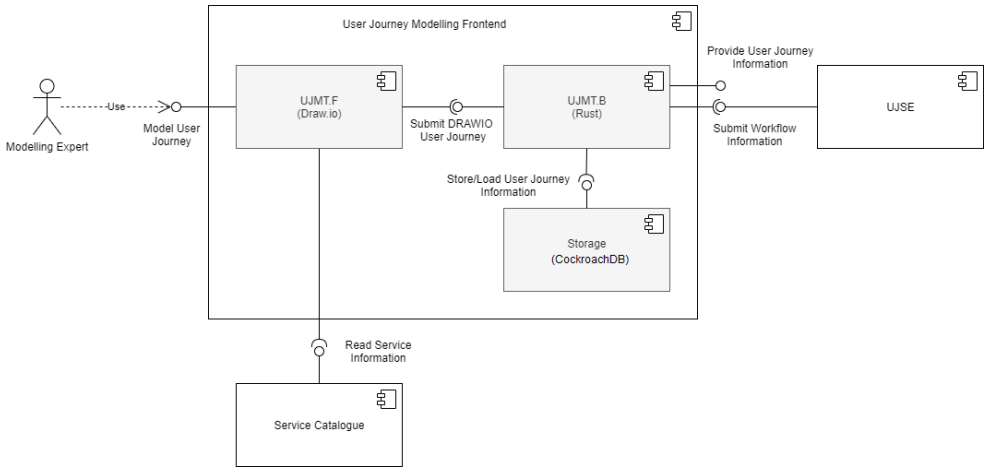


Figure 1: UJMT Overview

The UJMT consists of three main components:

- The frontend **UJMT.F** is based on the open-source modelling tool Draw.io and allows the modelling expert to interactively create UJM for ACROSS. The UJMT retrieves the necessary service information from the Service Catalogue and makes it available to the modelling expert.
- In the storage **UJMT.S**, users can centrally store UJMs with additional User Journey specific data and manage UJ publications in the UJSE.
- The backend **UJMT.B** creates a machine-readable User Journey Workflow Description (UJWD)[1] from a UJM, that describes the orchestration based on the modelled workflow for the UJSE. In the final version, these descriptions are provided in JSON format which the UJSE also provides to the Citizen Frontend. The UJMT.B furthermore provides an interface to make User Journey specific data from the UJMT.S available to other components in the ACROSS platform.

The three UJMT Modules are described in more detail in section 2.4.1.

The UJMT aims to support the specification of abstract and concrete workflows. While for concrete workflows, origin and destination country are defined and services are specified for all actions, abstract workflows describe more general processes that can be specified for many country combinations. Based on the initial state of considerations within the use cases work packages, it was decided to use a simple, sequential representation, covering the individual workflow steps as successive UJ phases.

In order to find out which adjustments had to be made to the Intermediate Prototype in the last project phase, a test workshop with the pilot partners was carried out, in which, on two separate dates, the functionalities of the Intermediate Version of the UJMT were presented and some UJM drafts were created, some of which can be seen in Figure 2 and Figure 3.

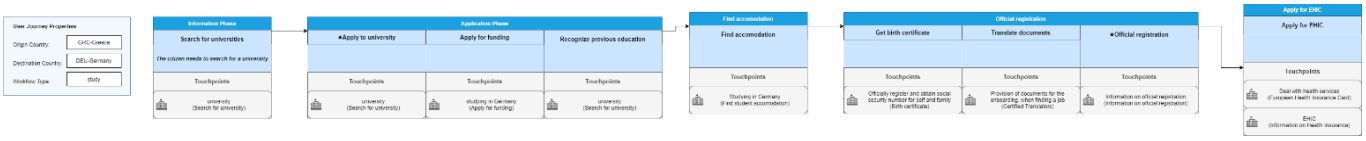


Figure 2: UJ Model draft for the scenario 'From Greece to Germany for Studying' created in the Test Workshop

[1] In the component card for the UJMT [19] we referred to the orchestration description as User Journey Workflow Template (UJWT). This term had been introduced to describe a more abstract orchestration description that meanwhile (i.e. after continued design discussions between the ACROSS technical partners) has been discarded.

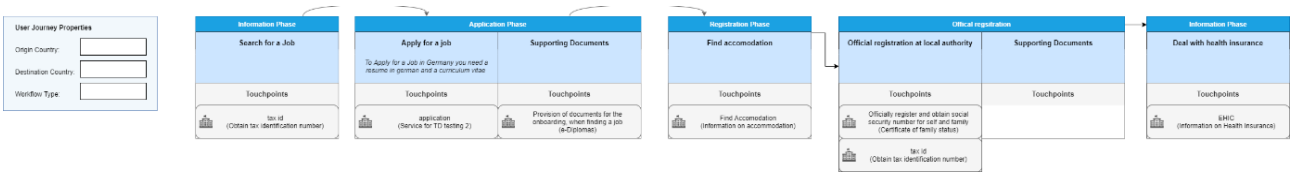


Figure 3: UJ Model draft for the scenario 'From Greece to Germany for Working' created in the Test Workshop

Following this workshop, the pilot partners created a total of five UJMs for the defined pilot scenarios in their own work. The goal was to test the complete process from the creation of the UJ in the UJMT to its publication in the Citizen Frontend. Figure 4 shows the finalized version of the draft in Figure 2 which has been published as a functional UJ in the Citizen Frontend.

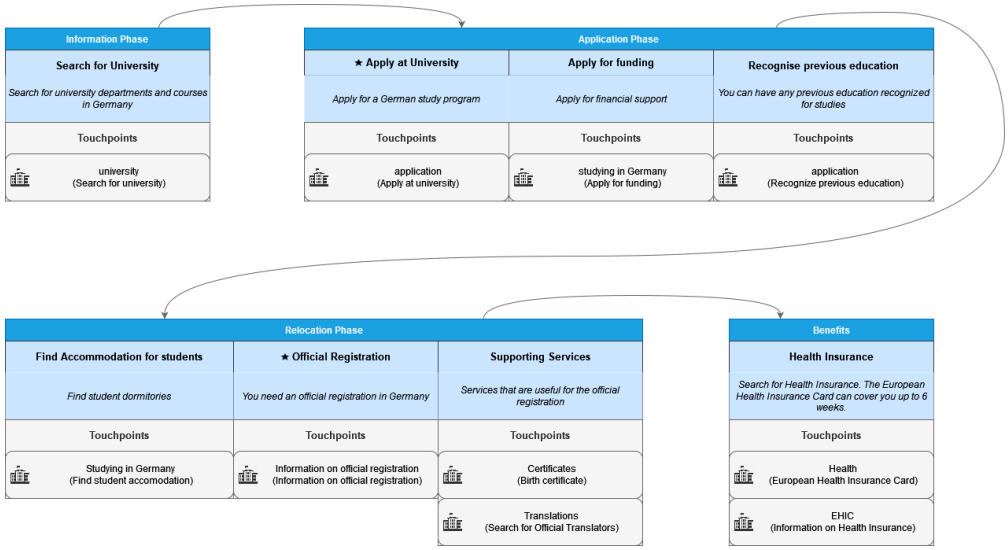


Figure 4: UJ Model for the scenario 'From Greece to Germany for Studying'

From the additional requirements formulated by the pilot partners, specified requirements for the UJMT were derived in cooperation. A core aspect in the last phase of the UJMT implementation was the addition of multilingual modelling elements. All the resulting requirements are listed in section 2.3.5.

Figure 7 shows the UJ from Figure 4 in Greek. The Greek names and descriptions are an integral part of the model in the Final Prototype version. The language displayed in the model can be changed directly in the UJMT.

Based on the new requirements, the final UJ design was also slightly revised for the final version. The adjustments are described in more detail in section 2.5.1.

The Final version of the UJMT prototype allows the definition of multilingual phases and actions, as well as the assignment of several actions to each phase, to create an abstract workflow model. To concretise

the model, location-specific services/touchpoints can be assigned to each action. The tool allows manual assembling as well as generating of abstract and concrete workflow models and the addition of further UJ information. An executable orchestration description can be automatically generated from the model. The UJ publications in the UJSE can be managed via the UJ storage.

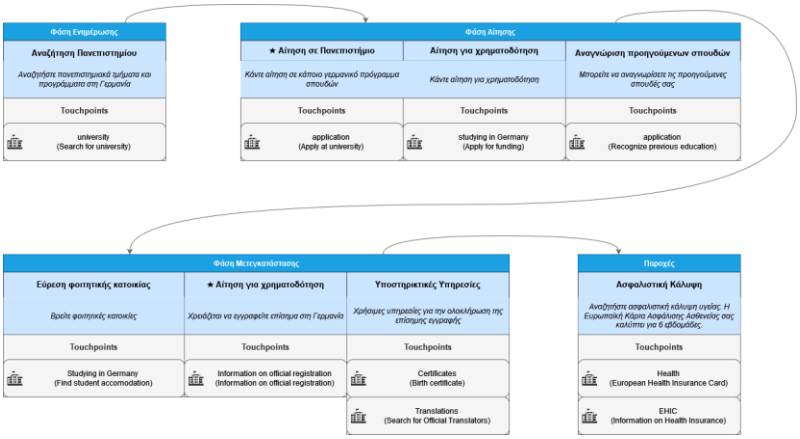


Figure 5: UJ Model for the scenario 'From Greece to Germany for Studying'

2.2 UJMT - ACROSS Context

This section describes the context that has influenced the definition of the UJMT, both on the European level and within the ACROSS project.

2.2.1 Relevant European initiatives and legislation

A considerable number of European projects have been bringing forward innovation in the field of business process modelling and workflow management within different application domains over the last decade. Examples include BPM4PEOPLE [5], Productive4.0 [6], HORSE [7], and COMPOSITION [8]. The intermediate level of abstraction exemplified in business process/workflow formalisms such as Business Process Modelling and Notation (BPMN) has proven very successful both as regards the modelling side, where process experts can express their designs without being distracted by overwhelming details of the individual steps, and as sound foundation for process/workflow execution and orchestration.

In recent years, new agile, user-centric design methodologies such as User Journey Mapping or Customer Journey Mapping have emerged and are becoming increasingly popular in product and service design on an international scale and across multiple industries. On the European level, this trend is exemplified by projects like PASSME [9, 10], which pushed for optimization of the air travel experience based upon the notion of the "passenger journey", or GRAVITATE-HEALTH [11], which focuses on optimizing "patients' journeys" in the digital health information domain. The ACROSS project, too, is built upon the premise

that User Journeys provide a superior framework for collaboratively thinking about and designing the interaction of users (in the chosen application domain of ACROSS: citizens) with a complex Information and Communications Technologies (ICT) system.

User Journeys, however, are inherently more abstract and loosely-defined when compared to business process or workflow models, even though they clearly contain workflow aspects. The development of the ACROSS User Journey Modelling Tool strives to bridge this gap. The UJMT's goal is to empower User Journey Modelling experts to lay down and refine the workflow aspects of the User Journey Maps they create in the form of a reusable, digital *abstract workflow* model. After a subsequent concretization step, this model can directly be used to orchestrate and execute workflows within the ACROSS platform. The individual steps of these *concrete workflows* will be the elementary services that the ACROSS platform is built to make available to its users - the citizens.

These elementary Public Services are another aspect where existing European initiatives are very relevant for the development of the UJMT. In particular, several present and future results of the ISA² programme [12] of the EU will serve as the basis for empowering the modelling experts to discover, select and integrate concrete Public Services into their User Journey Models, both on abstract and concrete workflow level. Even though details remain to be worked out, several parts of the ISA² service model are clearly relevant here, most importantly, the Core Public Service Vocabulary (CPSV), the Core Public Service Vocabulary Application Profile (CPSV-AP) [13], and the European taxonomy for public services [14].

2.2.2 Approach and Relation to other Work Packages and Deliverables

The R&D work being done on the UJMT is closely related to several other Work Packages of ACROSS and their respective Deliverables. First and foremost, the primary motivator and inspiration of the R&D activities is the work being done in WP2 "ACROSS new Governance Model" T2.1 "User Journey Methodology definition" and WP6 "Use cases deployment, evaluation & impact assessment" T6.1 "Use cases definition and planning". From both these Work Packages and Tasks, important requirements to the UJMT are gathered, while the latter also provides insights about the nature and details of the concrete services (to be accessed within the use cases), which the UJMT will help orchestrate and make available through the ACROSS platform. Of course, with the UJMT to work in close coupling to several other parts of the ACROSS framework, there is also a close relationship with WP5 "Platform Integration & Mobility Applications" and all its tasks. For the same reason, there are close ties with the other Tasks within WP4 "ACROSS Modules Set-Up").

With respect to intra-project collaboration and synergy, the approach taken for implementing the UJMT R&D activities thus relies on two pillars:

a) close collaboration and communication with the responsible WP2 and WP6 partners, for gathering requirements to the UJMT
b) close collaboration with the WP5 partners, especially regarding T5.1 System Architecture, for achieving well-founded and strong integration with the relevant other parts of the ACROSS platform (most importantly, the User Journey Service Engine and the Service Catalogue)

Apart from that, the methodology employed in the R&D work on the UJMT emphasizes continuous orientation on the current SOTA both in industry and academic research, a strong reliance on mature open-source software, and consistent use of agile development practices in a form tailored to the environment of an applied research organization.

2.3 UJMT – Requirements

The UJMT covers some general ACROSS requirements and fulfils special requirements for the UJMT component. Those requirements are listed in this section.

For the Final Prototype, the process of requirements collection has been overhauled and optimized throughout the entire ACROSS project, by tightening the collaboration between the use-case analysis and development work in ACROSS WP2 and WP6 on the one hand, and the software component design, development and integration work within ACROSS WP4 and WP5 on the other hand. Improved agile processes were established and more flexible collaboration tools were introduced. Within this much more effective framework, based on the initial requirements documented in “D4.7 User Support Tools – Initial” [1] and additional requirements documented in “D4.8 User Support Tools – Intermediate" [2], existing requirements were refined into more detailed and technical ones, and additional requirements were formulated, both based on the latest results of the use case work in WP2 and WP6. The list of final Prototype requirements resulting from this process is maintained on an ongoing basis using a kanban-style agile methodology in a collaborative issue management system (Trello).

2.3.1 Initial Requirements from WP5

Req_10 is the foundational requirement from WP5 applying to the UJMT, whereas the remaining requirements shown below are general requirements to the components implemented in ACROSS, which also apply to the VA (to an appropriate extent).

Table 1: Requirements to the UJMT from WP5

Id	Title	Description	Type	Category
Req_09	Free access to other countries' e-services	Citizens should be able to access other countries' services	non functional	Platform architecture and interoperability
Req_10	User Journey Experience	Citizen should be able to navigate in a straightforward clearly defined way through the whole process of the User Journey provided user experience. I would also like to have support during the steps of the moving abroad process.	non functional	Platform architecture and interoperability
Req_15	Easy to use service integration and orchestration tools	In order to create cross border services, the platform has to support Public and Private providing a set of tools and applications that will help them to easily implement service integration.	functional	Connectors to integrate the private and public sector offering
Req_19	Reliability and Integrity	The implementation of ACROSS should follow open standards and use well-known and widely accepted technologies in order to ensure integrity. The ACROSS platform has to be reliable assuring integrity of the components/tools that are part of it.	non functional	Platform architecture and interoperability
Req_29	No vendor lock-in	I want the ACROSS reference architecture to be technologically agnostic to avoid vendor lock-in.	non functional	Platform architecture and interoperability
Req_30	Open source	I want the ACROSS reference architecture to reuse already available open source solutions and only create or improve those aspects that are not covered by the existing solutions	non functional	Platform architecture and interoperability

2.3.2 Key Performance Indicators for Initial Requirements from WP5

In order to assess the degree to which the requirements will be fulfilled by the UJMT implementation, the following Key Performance Indicators (KPIs) have been defined[2]:

Table 2: Proposed Key Performance Indicators (UJMT)

Requirement	Description	Proposed KPI	Unit
Req_09	Free access to other countries' e-services	Amount of (freely accessible) services in published User Journeys	Average % per User Journey
Req_09	Free access to other countries' e-services	Number of Public and Private Services in the UJMT	Number per origin country
Req_10	User Journey Experience	Number of submitted workflows	Number per origin country
Req_15	Easy to use service integration and orchestration tools	Amount of required user support	Number of requests
Req_19	Reliability and Integrity	Ratio of UJMT functionalities accessible via REST API	%
Req_19	Reliability and Integrity	Maintenance Cost	PM per month
Req_29	No vendor lock-in	UJMT containerized components	%
Req_30	Open source	UJMT open-source components	%

2.3.3 Initial Requirements from WP2 (ACROSS New Governance Model) and WP6 (Use Cases)

In addition to the general requirements, the following initial requirements have been agreed upon in discussion with the WP2 partners:

REQ-UJMT-1 As a Modelling Expert in ACROSS I want to be able to work on models in a graphical, interactive manner, on an appropriate level of abstraction.

REQ-UJMT-2 As a Modelling Expert in ACROSS I want to be able to add comments to modelling elements.

[2] Meanwhile, project-wide success criteria have been defined in D6.2 “Use Case Evaluation and Impact Assessment” that supersede these KPIs.

2.3.4 Initial Specific Requirements

The following fundamental requirements resulted from the work and exchange with the project partners on the tool in its initial state. The implementation of corresponding functionalities was continued for the intermediate version.

2.3.4.1 User Journey Modelling Requirements

REQ-UJMT-3 As a Modelling Expert in ACROSS I want to be able to easily create UJMs for different cross-border scenarios via graphical interactions such as drag and drop.

REQ-UJMT-4 As a Modelling Expert in ACROSS I want to be able to store the created UJMs.

REQ-UJMT-5 As a Modelling Expert in ACROSS I want to be able to load and edit previously stored UJMs.

REQ-UJMT-6 As a Modelling Expert in ACROSS I want to be able to export the created UJMs as PDF.

2.3.4.2 Modelling of Abstract Workflows

REQ-UJMT-7 As a Modelling Expert in ACROSS I want to be able to create abstract workflow steps as part of the UJM and combine them into abstract workflows.

2.3.4.3 Transition to Concrete Workflows for Orchestration

REQ-UJMT-8 As a Modelling Expert in ACROSS I want to be able to technically refine abstract workflows into concrete workflows in order to prescribe the service orchestration for the UJSE.

REQ-UJMT-9 As a Modelling Expert in ACROSS I want to be able to map abstract workflow steps to concrete services from the Service Catalogue.

REQ-UJMT-10 As a Modelling Expert in ACROSS I want to be able to map abstract workflow steps to concrete sub-workflows that contain several concrete services from the Service Catalogue.

REQ-UJMT-11 As a Modelling Expert in ACROSS I want to be able to provide all the necessary data for the service orchestration in the UJM.

2.3.4.4 Providing Building Blocks / Templates

REQ-UJMT-12 As a Modelling Expert in ACROSS I want to be able to use pre-made templates that support the user journey modelling.

REQ-UJMT-13 As a Modelling Expert in ACROSS I want to be able to add my own templates for further use.

2.3.5 Intermediate Requirements from WP2 (ACROSS New Governance Model) and WP6 (Use Cases)

For the intermediate version, the following new requirements that resulted from the modelling of the pilot scenarios were discussed and partially implemented:

REQ-UJMT-14 As a Modelling Expert in ACROSS I want to be able to map several services to one action.
REQ-UJMT-15 As a Modelling Expert in ACROSS I want to be able to categorize actions.
REQ-UJMT-16 As a Modelling Expert in ACROSS I want to be able to make actions mandatory.
REQ-UJMT-17 As a Modelling Expert in ACROSS I want to be able to add a description for a User Journey.
REQ-UJMT-18 As a Modelling Expert in ACROSS I want support in the ordering of potentially dependent services.
REQ-UJMT-19 As a Modelling Expert in ACROSS I want to be able to define personas.

2.3.6 Intermediate Specific Requirements

Furthermore, the following intermediate requirements resulted from the ongoing work and exchange with the project partners on the tool:

REQ-UJMT-20 As a Modelling Expert in ACROSS I want to be able to filter services by defining workflow type, destination country and origin country.
REQ-UJMT-21 As a Modelling Expert in ACROSS I want to be able to read the service information in my native language.

2.3.7 New Requirements from WP6 (Use Cases)

For the final version, the following requirements were established during modelling of the pilot scenarios and the follow-up discussions with the WP6 Partners.

REQ-UJMT-22 As a Modelling Expert in ACROSS I want to be able to define the action and phase names in multiple languages.
REQ-UJMT-23 As a Modelling Expert in ACROSS I want to be able to define the action and UJ descriptions in multiple languages.

2.3.8 New Specific Requirements

Moreover, the following final requirements resulted from the ongoing work and exchange with the project partners on the tool:

REQ-UJMT-24 As a Modelling Expert in ACROSS I want to have trust indicators about the status of the user journey

REQ-UJMT-25 As a Modelling Expert in ACROSS I want to have a specified text formatting in the tool.

2.3.9 Implementation Matrix

The following table shows the implementation status of the identified requirements:

Table 3: Implementation Matrix (UJMT)

	Integrated Version	Intermediate Version	Final Version
Requirements Initial Version			
REQ-UJMT-1	Implementation	Adjustments	Adjustments as described in 2.5.4, 2.5.6, 2.5.8
Error! Reference source not found.	Base Tool Functionality	-	-
Error! Reference source not found.	Implementation	Adjustments	Adjustments as described in 2.5.1, **Error! Reference source not found.**, 2.5.5, 2.5.6, **Error! Reference source not found.** and 2.5.8
Error! Reference source not found.	-	Concept	Implementation as described in 2.5.9
Error! Reference source not found.	-	Concept	Implementation as described in 2.5.9
REQ-UJMT-6	Base Tool Functionality	-	-
Error! Reference	Implementation	Adjustments 2.5.1**Error! Reference source not found.**	-

source not found.			
REQ-UJMT-8	Implementation	Adjustments 2.5.1	Adjustments as described in 2.5.1 and 2.5.8
Error! Reference source not found.	Implementation	Adjustments2.5.1	Adjustments as described in 2.5.8
Error! Reference source not found.	Concept	-	Out of scope
Error! Reference source not found.	Concept	Implementation **Error! Reference source not found.Error! Reference source not found.**	Adjustments as described in 2.5.10
REQ-UJMT-12	Concept	Implementation **Error! Reference source not found.**	Adjustments as described in 2.5.4 and 2.5.6
REQ-UJMT-13	Base Tool Functionality	-	Concept and implementation as described in 2.5.4 and 2.5.9
Additional Requirements Intermediate Version			
REQ-UJMT-14	-	Implementation 2.5.1	-
REQ-UJMT-15	Implementation	Adjustments 2.5.1	-
REQ-UJMT-16	-	Implementation 2.5.1**Error! Reference source not found.**	-
REQ-UJMT-17	-	Implementation	Adjustments as described in 2.5.1 and 2.5.2.1
REQ-UJMT-18	-	-	Out of scope
REQ-UJMT-19	-	-	Out of scope

REQ-UJMT-20	-	Implementation **Error! Reference source not found.**	-
REQ-UJMT-21	-	-	Implementation as described in 2.5.8
Additional Requirements Final Version			
REQ-UJMT-22	-	-	Implementation as described in 2.5.1 and 2.5.2.1
REQ-UJMT-23	-	-	Implementation as described in 2.5.1 and 2.5.2.1
REQ-UJMT-24	-	-	Implementation as described in 2.5.9
REQ-UJMT-25	-	-	Base Tool Functionality

2.4 UJMT - Architecture

The following section provides an overview of the main modules of the UJMT.

2.4.1 UJMT - Modules

2.4.1.1 UJMT Frontend

The UJMT.F is the graphical frontend tool for interactively, in a diagram-like style, modelling UJMs for ACROSS. The modelling of central workflow related aspects directly affects the intended service orchestration. The UJMT.F is being implemented as an extension of the open-source project Draw.io. It is able to export a file-based model representation in an open and documented structured data format (based on XML) and can be reliably deployed in the ACROSS platform. The web user interface enables the Modelling Experts in ACROSS to interactively construct and modify UJMs, and can incorporate service information from the Service Catalogue.

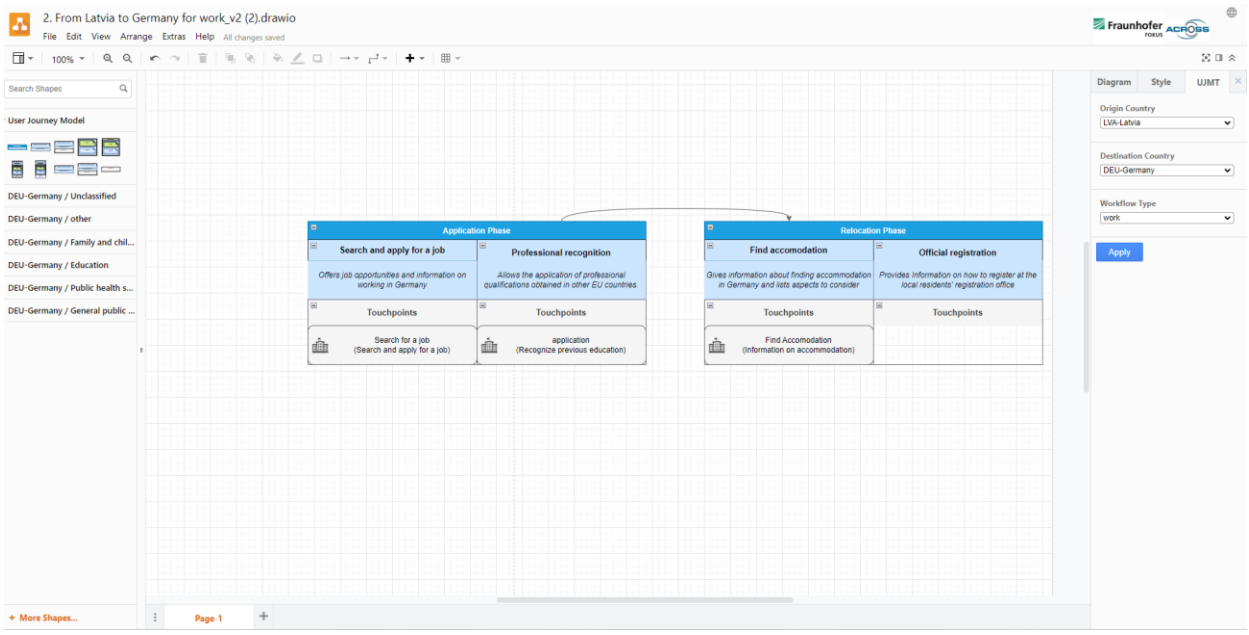


Figure 6: UJMT User Interface and example model

2.4.1.2 UJMT Backend

The UJMT.B is the backend component that provides support for implementing the intended service orchestration, based on the objects and relationships in a UJM built in the UJMT.F. From the model, the UJMT.B will create a machine-readable concrete orchestration description based on the modelled workflow (referred to as a User Journey Workflow Description, or UJWD). This orchestration description can be provided to the UJSE, where it is used to orchestrate and execute the User Journey Service Workflows. Furthermore, it provides a REST API that allows other ACROSS components to access data stored in the UJMT.S.

2.4.1.3 UJMT Storage

The UJMT.S is a database for UJMs created in the UJMT.F. It serves as a central storage that holds UJM with abstract as well as concrete workflows and can be used for status management when providing the UJMs to the UJSE. User Journey related information aside from the workflow with relevance for the citizens is provided here.

2.4.2 UJMT - Interfaces and Collaborations

2.4.2.1 User Journey Service Engine

The UJMT can provide the created UJWDs to the UJSE, as well as update or delete previously provided UJWDs. In the UJSE, the UJWDs are used for the instantiation of the concrete User Journey Service Workflows. To communicate with the UJSE, an interface (REST) was implemented.

2.4.2.2 Service Catalogue

The UJMT has an interface to the Service Catalogue to receive information about already registered services. In the UJMT, the Modelling Expert is able to use the information for the creation of concrete workflows.

2.4.2.3 Citizen Frontend

The UJMT provides an interface (REST) for the Citizen Frontend to request information about the stored UJMs and, in particular, their publication and approval status information.

2.5 UJMT - Description of Final Version

The goal of the Final Prototype version was the further clarification and implementation of the existing requirements, as well as the development and implementation of new requirements with the pilot partners, to provide a suitable, expandable UJMT prototype that is fully integrated in the ACROSS platform and provides all the functionalities needed to create and publish UJs for the pilot scenarios defined in the ACROSS project, as well as further scenarios for potential future use cases.

2.5.1 User Journey Definition

For the Intermediate Prototype version, the User Journey's definition was sharpened and differentiated from the workflow. In addition, the properties of concrete and abstract workflows were determined, and the results were presented in an entity relationship diagram. For the Final Prototype version, this diagram has been updated and expanded with new elements. Figure 7 shows the final version of the entity relationships in the UJMT.

For the Intermediate Prototype version, it was decided that a DRAWIO document may contain at most one UJM. This design decision was retained for the Final Prototype version.

As in the Intermediate Prototype version, UJ and workflow are fundamentally separated in the Final Prototype version, whereby each UJ may contain a maximum of one workflow. The workflow can be abstract (REQ-UJMT-7) or concrete (REQ-UJMT-8) depending on the completeness of the provided meta information and the assigned services (REQ-UJMT-9). Concrete workflows have associated origin and destination countries as well as a type specification, and each action is assigned a service from the Service Catalogue.

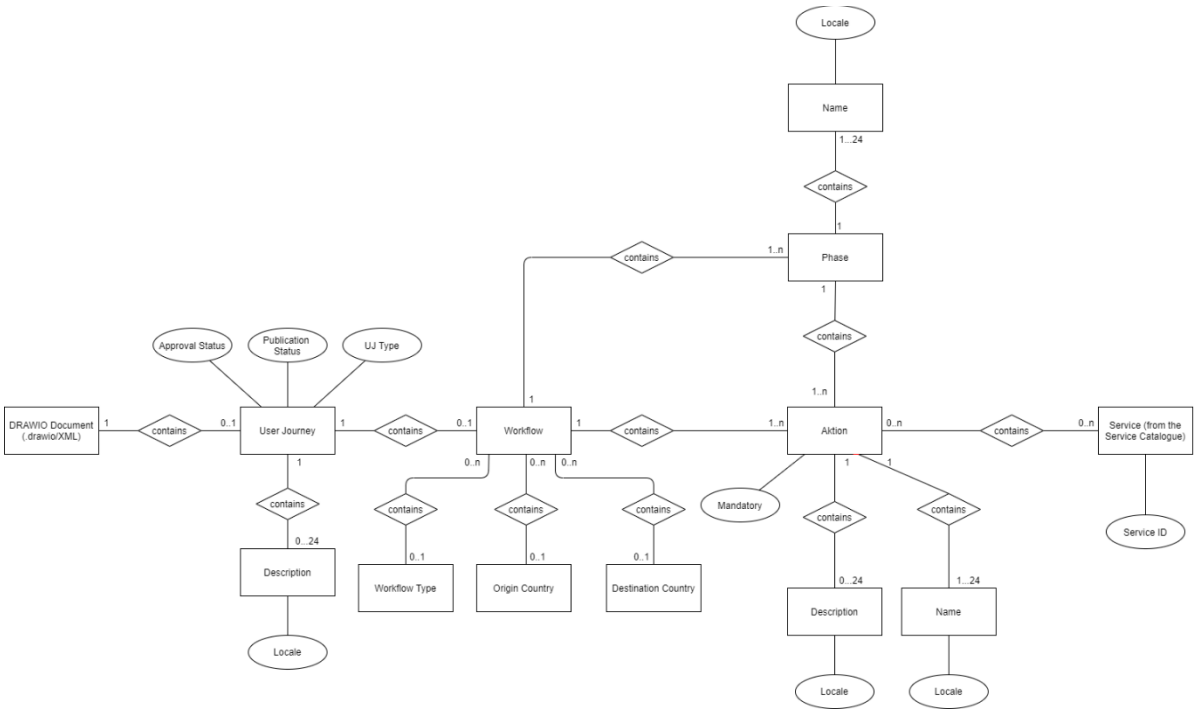


Figure 7: Entity Relationship Diagram for User Journey Modelling (Final Version)

Each action can be mapped to multiple services from the Service Catalogue, to provide the citizen a selection of suitable services if necessary (REQ-UJMT-14). As for the workflow design, the categorization of actions has become an integral part (REQ-UJMT-15), which results in each action having to be mapped to a phase. Consequently, a workflow consists at least of one phase with one assigned action. For each action, the Modelling Expert can choose whether it should be mandatory or optional (REQ-UJMT-16).

Since the Intermediate Prototype version, each UJ can be assigned a textual description (REQ-UJMT-17). In the Final Prototype version, this requirement was expanded to include multilingualism (REQ-UJMT-23). This multilingualism was also implemented for the naming of phases and actions (REQ-UJMT-22) as well as the action descriptions (REQ-UJMT-23). For the implementation it was decided to support input values for the 24 official European languages[3]. The implementation is described in more detail in Section 2.5.2.

Finally, status values were introduced for the UJ management, which provide information about the publication and approval status (REQ-UJMT-24).

[3] The 24 official European languages include Bulgarian, Croatian, Danish, Dutch, English, Estonian, Finnish, French, German, Greek, Irish, Italian, Latvian, Lithuanian, Maltese, Polish, Portuguese, Romanian, Slovak, Slovenian, Spanish, Czech and Swedish.

2.5.2 Implementation of Multilingual Support

In order to be able to offer information for the Modelling Expert in several languages in the UJMT, a distinction was made between the requirement to make language settings for the display language in the tool itself (described in more detail in Section 2.5.2.2) and the requirement to change the language displayed in the model (described in more detail in Section 2.5.2.1).

2.5.2.1 Multilingual Models

For the following parts in the UJM, the UJMT offers the Modelling Expert to define values for several languages:

- Phase Names
- Action Names
- Action Descriptions
- UJ Descriptions

In the implementation, these values can still be entered as labels of the specific graphical elements directly in the model. The value is then entered for the currently selected display language.

The display language of the model can be selected in the menu. This makes it easy to determine for all elements in the UJM whether entries are still missing in a specific language, and the modelled UJ can be viewed and adjusted in its entirety in the selected language.

In addition, the Modelling Expert can right-click on a phase, an action or an action description to open a dialog via the 'Edit Name' or 'Edit Description' option, in which all element related multilingual values can be defined in one view. For the UJ description, this dialog can be opened via the menu.

Figure 8 shows the dialog windows for multilingual names and descriptions in the UJMT.

Dialog for defining names of phases and actions *Dialog for defining descriptions*

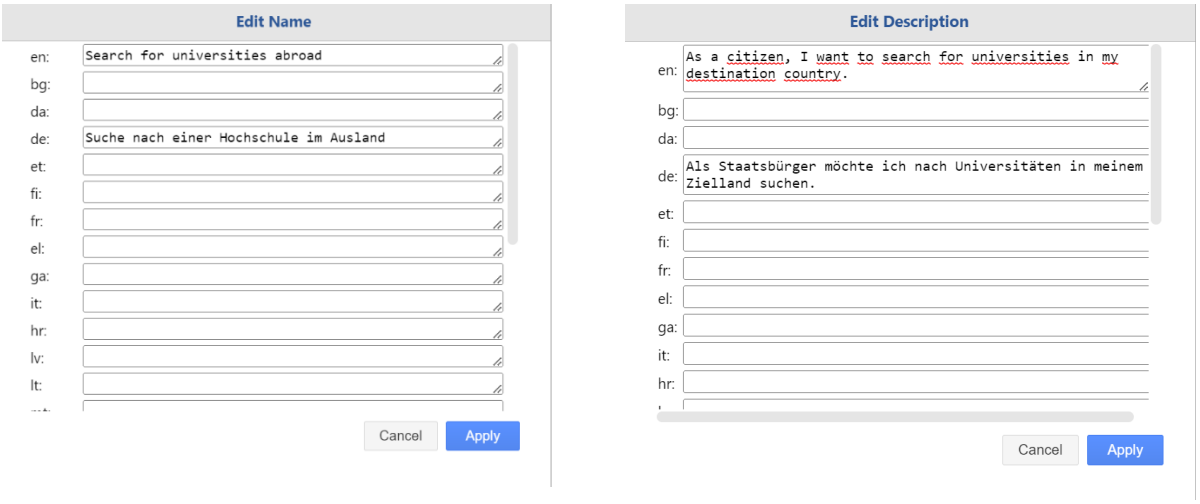


Figure 8: Dialogs for defining multilingual values in the UJMT

2.5.2.2 Multilingual Tool

Draw.io is already offered for many languages. For the Final Prototype version, the multilingual display was extended for all implementations. For this purpose, in adaptation to the implementation in the code base, separate properties files were created for each language, which are read and evaluated in the UJMT depending on the language selection. This is limited e.g., in that some values that are displayed to the Modelling Expert are directly read from the Service Catalogue where they are only available in English, such as the select options for selecting the workflow type. By more precisely coordinating the values in the Service Catalogue and in the UJMT, a more uniform representation could be achieved in the future.

2.5.3 Modelling and Publication Process

The process for creating and publishing UJ can be divided into the following phases:

- UJ Creation
- UJ Editing
- UJ Concretisation
- UJ Publication

The next sections describe how the Modelling Expert is supported by the UJMT in each phase.

2.5.4 Tool Support for UJ Creation: Abstract User Journey Models Templates

To facilitate UJ modelling, we have created a possibility to centrally store and load UJMs with abstract workflows with the UJ Storage. This enables the Modelling Expert to define and use templates for UJs to achieve a faster way to model the UJ. The clear advantages are that the Modelling Experts can start with

a basic UJ structure for a specific use case, which also means they do not have to provide all the extensive multilingual input themselves. However, the Modelling Experts can make any changes needed.

Figure 9 shows a very simple but conceivable UJM with an abstract workflow for the workflow type "study". It contains phases and actions from the various UJMs for the "Studying Abroad" scenarios that the partners have created throughout the last project phase with the UJMT. Figure 9 shows the model in English, but other display languages can be selected in the UJMT. The model can be concretised by specifying origin and destination country, adjusting the workflow, expanding the descriptions if needed and adding the appropriate services from the Service Catalogue.

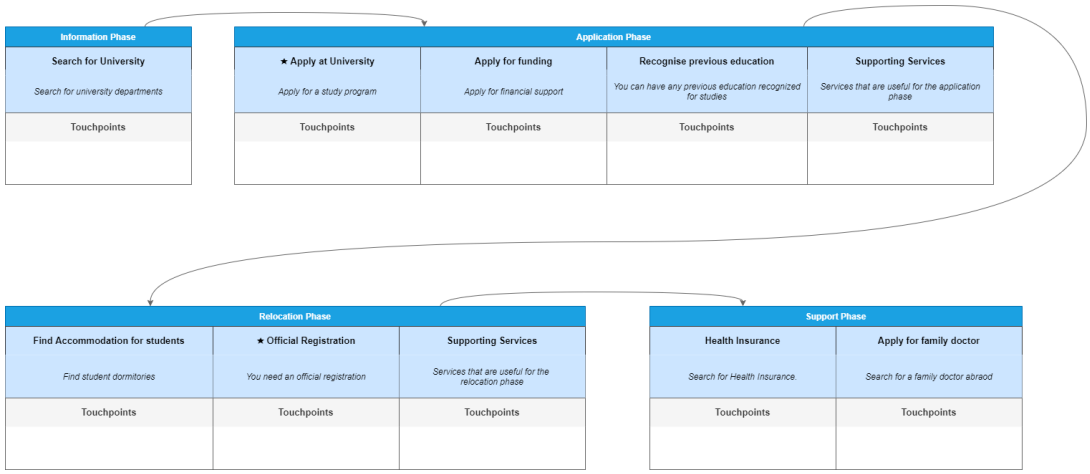


Figure 9: Example for a UJ with abstract workflow for the workflow type "study"

2.5.5 Tool Support for UJ Creation: Final "Generate User Journey" Dialog

As part of the PoC for the first milestone in the ACROSS project, we implemented a dialog that simplifies the start of UJ modelling. The Modelling Expert can define phases and actions in the dialog and generate pre-defined graphical elements from the input. This allows the Modelling Expert to start modelling with a prepared set of modelling elements which can be rearranged and completed with additional modelling elements from the left sidebar. The generated elements contain the necessary information for the orchestration.

The Modelling Expert can call the 'Generate User Journey' dialog from the menu. In the dialog the Modelling Expert can define phases and related actions of the UJ and provide additional information. For the Final Prototype version, the 'Generate User Journey' dialog will be redesigned to better reflect the displayed UJ in the Citizen Frontend. Since the arrangement of phases and actions in the workflow has been rethought and additional configuration options were added during the project, the new dialog will

enable the definition of multilingual phases and actions in one common view. Figure 10 shows our mock-up for the redesigned dialog.

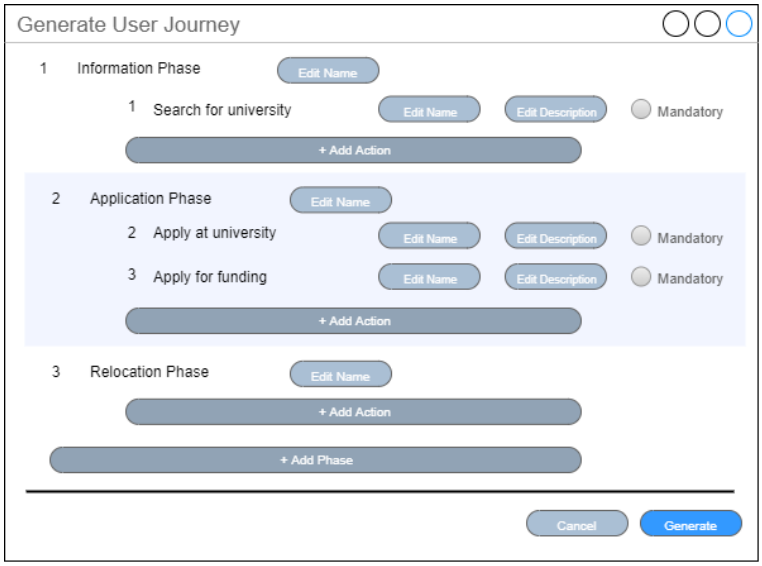


Figure 10: Mock-up for the new 'Generate User Journey' Dialog

From the input data, the UJMT creates a structured set of modelling elements as a first draft of the UJ. From here, the user can manually assemble the modelling elements and add further elements from the left sidebar.

2.5.6 Tool Support for UJ Editing: Predefined Modelling Elements

To make creating and editing UJMs as easy as possible for Modelling Experts, we have prepared a set of modelling elements to be used and made them available in the left sidebar in addition to the option of generating a UI frame with the 'Generate User Journey' dialog.

All the elements have predefined properties, e.g., guiding tooltips, as well as predefined vertex properties, that allow the users, e.g., to only connect or manipulate certain elements. Since the modelling options in the base tool Draw.io are very comprehensive, the options for creating the UJM have been restricted as much as possible to simplify modelling process.

Figure 11 shows a selection of some of the predefined modelling elements in the UJMT. The elements are displayed in a separate tab in the left sidebar next to the canvas and can be used at any time.

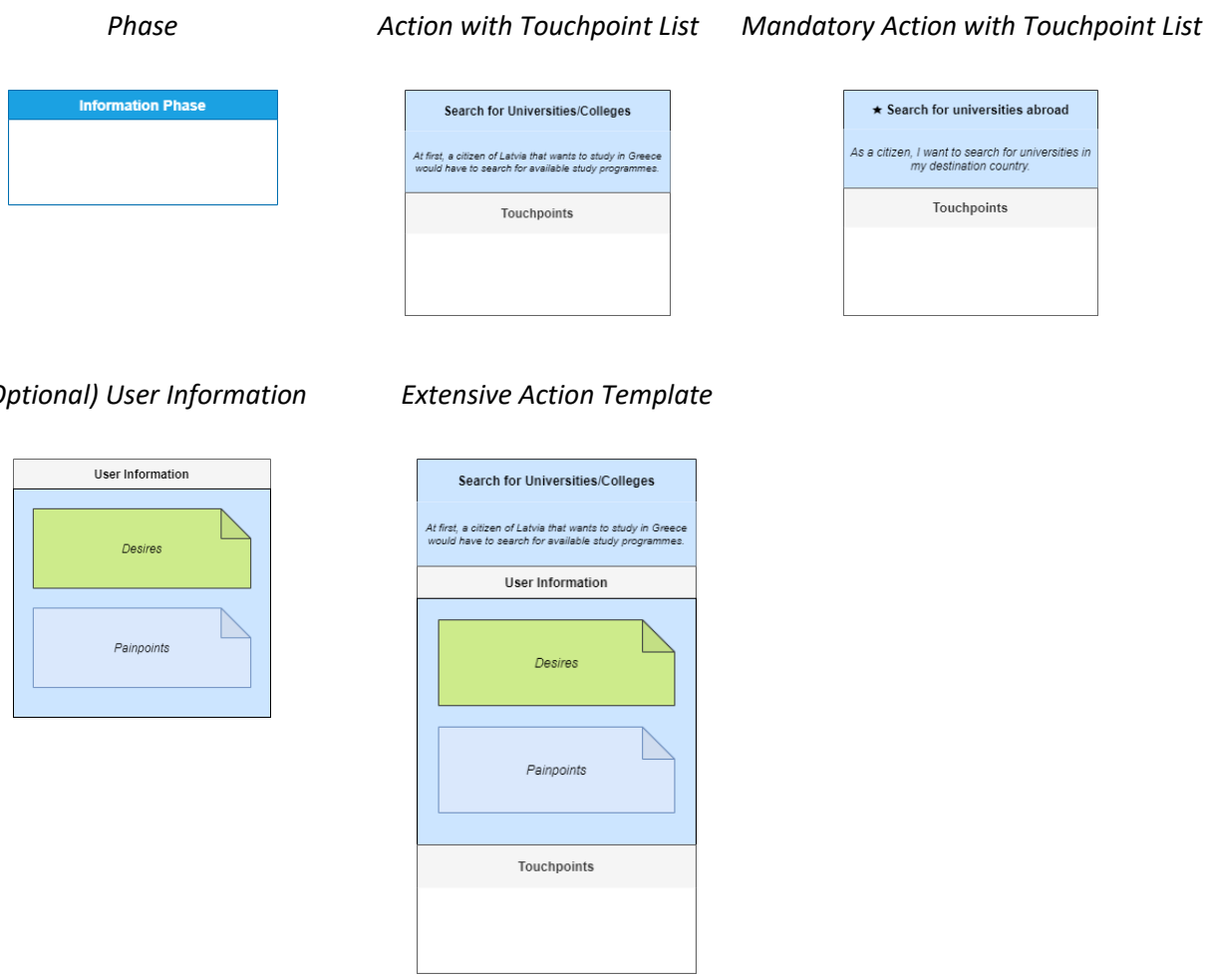


Figure 11: Selection of predefined modelling elements in the UJMT

2.5.7 Tool Support for UJ Editing: UJ Validation

In Section 2.5.6, we described how the properties of the predefined elements already limit the possibilities for UJ modelling and thus support the Modelling Expert in editing UJMs. To support the editing process even more, we introduced the option of modelling with UJ validation in the Intermediate Prototype version. With this setting, the UJM is validated in the backend in the event of changes. The results of the validation are displayed in the view via highlights (around touchpoint elements that are in a touchpoint list and are thus recognized as part of the workflow) and additional labels (for actions recognized as part of the workflow due to correct placement in a phase). Figure 12 shows these markings in a section of an example model.

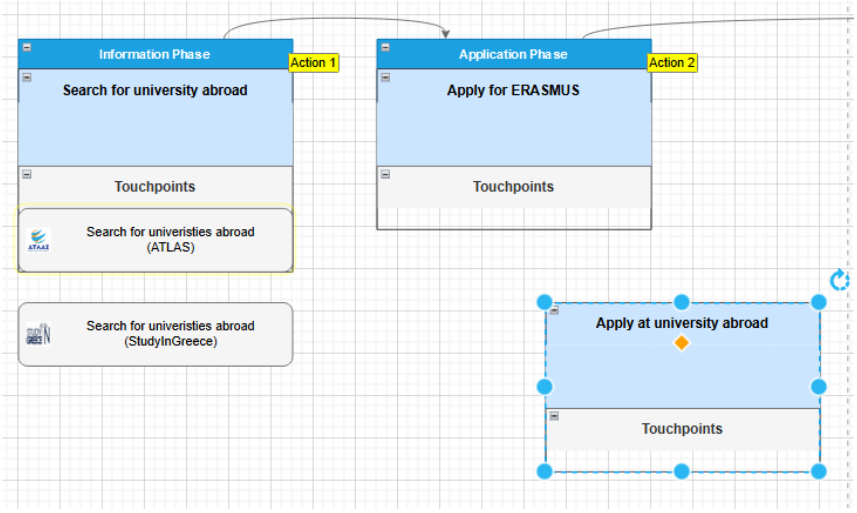


Figure 12: Screenshot of a part of a marked UJM with two associated actions (yellow labels) and one associated service (yellow highlight)

The user feedback via UJ validation ensures that the Modelling Expert can see whether phases, actions and services have been modelled as desired and required, and whether they have also been linked correctly in the model. The following aspects are checked and determine the user feedback:

- All phases on the canvas must be connected by arrows.
- Circle connections and connections from one phase to several other phases are not allowed.
- Actions must be placed inside a phase to belong to the User Journey.
- A concrete User Journey contains at least one action and at least one touchpoint (service) for each action.
- A concrete User Journey has the Properties *origin_country*, *destination_country* and *workflow_type*.

The validation criteria have been expanded for the Final Prototype version with the following aspects:

- In a concrete User Journey, all phases have an entry for the English language field in the 'Edit Name' dialog.
- In a concrete User Journey, all actions have an entry for the English language field in the 'Edit Name' dialog.

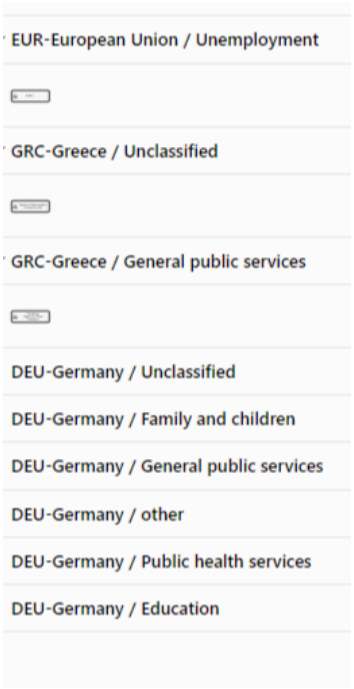
The validation is also applied before generating and submitting documents to the UJSE.

2.5.8 Tool Support for UJ Concretisation: Service Catalogue Integration

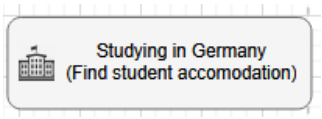
One of the main objectives for the UJMT was the integration of the Service Catalogue and the provision of graphic elements for the services in the user interface. The implementation from the Intermediate version of the Prototype was adapted for the Final version through finer categorization. Since the services

are provided as modelling elements in the left sidebar, they must be categorized in an appropriate manner. In the Final Prototype version, the services are sorted by the countries and thematic areas specified in the Service Catalogue. The left sidebar provides a search function, with which services can be found even if the number of services in the Service Catalogue increases quickly and easily.

Categorized Services in the left Sidebar



Graphical representation of the Service 'Studying in Germany'



Tooltip with Service Data from the Catalogue

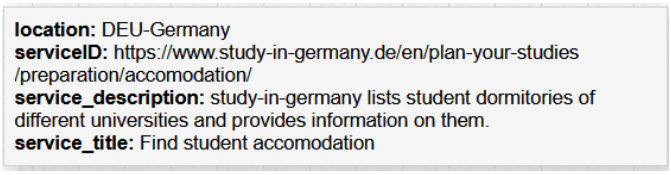


Figure 13: Screenshot of the left sidebar in the UJMT

The graphical elements for services can be used to concretise a workflow by assigning them to actions via the touchpoint list.

A selection of the data that the Service Catalogue defines for each service is included as properties in the graphical element and displayed for the user as a tooltip. In the Final version, this data includes:

- The **service ID** by which the service is referenced in the Service Catalogue
- the **location** that specifies the country for which this service is offered.
- a textual **service description** in the selected model language. If the service description is not specified for the selected language in the Service Catalogue, the description is displayed in English. If this option is not available either, the first description given will be displayed. This procedure was adopted from the implementation in the Citizen Frontend.

If necessary, further data from the Service Catalogue can be displayed. Part of this data is transferred to the UJMT Backend for BPMN generation when submitted to the UJSE.

In addition to selecting elements from the left sidebar, there is an alternative option of converting any graphic element into a special service by right-clicking on the service selection menu.

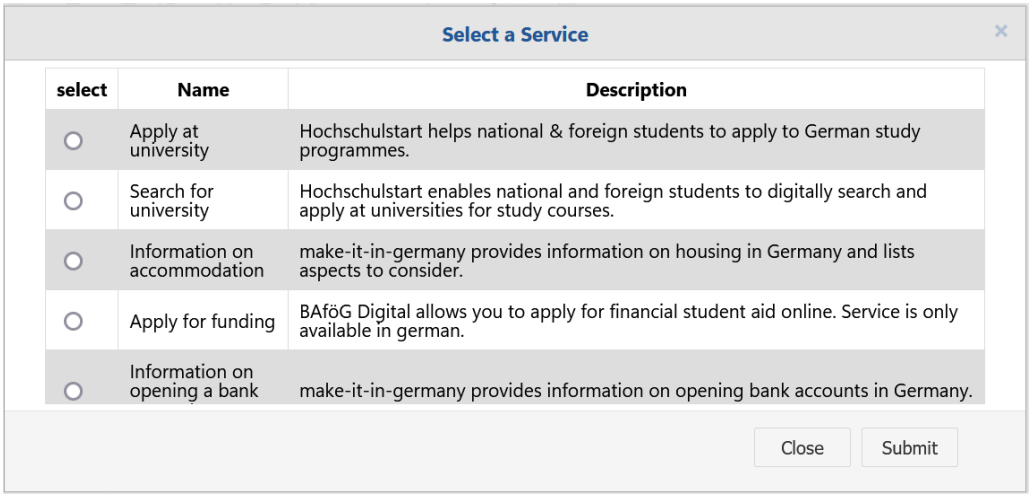


Figure 14: 'Select a Service'-Dialog

As a further support, the services in the service bar and the 'Select a Service' dialog can be filtered by selecting and applying destination country, origin country and workflow type in the right sidebar. The services that do not apply to the selected countries and the workflow type will not be shown to the user, which prevents the user from using a service that is not intended to be in the user journey.

2.5.9 Tool Support for UJ Publication: Status Management

The Final Prototype version of the UJMT contains a database with which Modelling Experts can centrally manage UJMs. UJ management includes creating, customizing and deleting abstract and concrete UJMs, as well as publishing and deactivating UJs in the Citizen Frontend. In the Final Prototype version of the tool, UJMs can be managed via status changes which can be initiated by the Modelling Experts in a separate dialog. Figure 15 illustrates the states and state transitions of a UJ developed for the Final version.

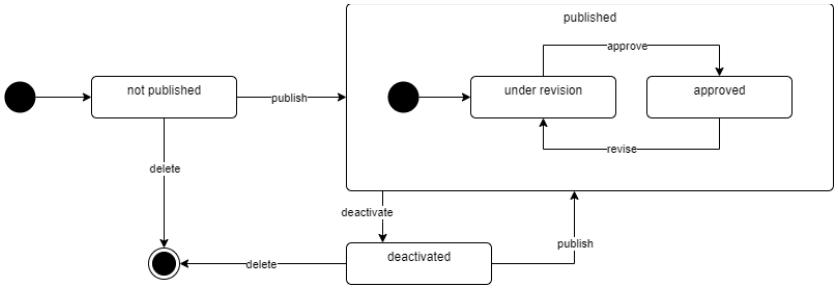


Figure 15: User Journey State Diagram

A UJ can have one of the following three states: *not_published*, *published*, and *deactivated*. All UJs that are created in the UJMT but not stored in the storage cannot be published and therefore have the status *not_published*. If the UJ is stored in storage, this status is initially retained. If the UJ is concrete and thus meets defined criteria, it can be published via the UJSE. For this purpose, the Modelling Expert can publish the UJ via the dialog. Once the UJ is successfully sent to the UJSE, its state changes to *published*. A published UJ will be made available in the Citizen Frontend, which means it is possible for a citizen to instantiate the specified workflow. A published UJ can be deactivated in the dialog, for example if the User Journey is no longer accurate. The UJ is then deleted from the engine and its status is set to *deactivated*. A deactivation does not affect already instantiated workflows which will remain in the UJSE until the citizen terminates the UJ. Deactivated UJs can be deleted and thus removed from the Storage. If a deactivated UJ is not removed from the Storage, it can again be published in the Engine.

Published UJ can assume two sub-states depending on stored approval indicators: *under revision* and *approved*. When published, each UJ initially assumes the status *under revision*, so the user is aware that this UJ is yet to be reviewed by professionals. This status can be set to *approved* in the UJMT via the dialog. Should an approved UJ be deactivated, its status will be set back to *under revision* upon re-release. The exact procedure of the approval process for User Journeys is still to be discussed with the partners.

During the project, it emerged that only one UJ should be published for each combination of origin country, destination country and workflow type. This design decision means that UJs can only be published in the same way via the storage: if a UJ has already been published for a specific scenario, it must first be deactivated before another UJ can be published for the same scenario. Therefore, applying updates for UJs follows the same process of deactivating the old version and releasing the new version.

In the storage, the following values are recorded for each UJ:

- **id**: unique identifier of the UJ
- **title**: title of the related DRAWIO file
- **drawio_xml**: XML file automatically created in Draw.io that contains the UJ diagram

- **uj_type**: result of the UJ validation which can be either *abstract* or *concrete*
- **workflow_type**: if specified in the UJMT: the workflow type, which can be either *work* or *study*, as specified by the Modelling Expert
- **origin_country**: if specified in the UJMT: country in which the UJ starts, as specified by the Modelling Expert
- **destinantion_country**: if specified in the UJMT: country in which the UJ ends, as specified by the Modelling Expert
- **status**: state of the UJ, either *not_published*, *published* or *deactivated*
- **trust_indicator**: if UJ status is *published*: sub-state of the UJ, either *under_revision* or *approved*
- **engine_id**: if UJ status is *published*: identifier of the UJ in the UJSE
- **json_description**: if UJ status is *published*: JSON representation of the workflow which describes the workflow of the UJ to the UJSE
- **timestamp**: temporal information on when the UJ was added to the Storage

2.5.10 Tool Support for UJ Publication: Generation of the Orchestration Description

Another main objective for the UJMT was the implementation of a generator for the automatic creation of an orchestration description from a UJM for the UJSE. Two generators were implemented for this purpose during the project: one generator for creating BPMN diagrams that can be used in workflow engines for workflow automation, and another generator for creating a workflow description in JSON format, which was originally used for displaying the UJ in the Citizen Frontend.

Both generators are part of the UJMT.B. Before each generation, a validation of the UJM is performed to check if the modelled workflow is concrete.

During the project, it turned out that in the implementation of the UJSE, the usage of the JSON representation was more and more preferred over the BPMN diagram, since it allowed to enable the citizen to carry out the modelled actions in any order. Since this requirement and hence the related design decision had not yet been made at the beginning of the project, it was decided among the technical partners to first implement a BPMN generator. Thus, the BPMN generation is currently not used in the Final Prototype version of the ACROSS platform; both engine and Citizen Frontend use the JSON representation.

The currently used JSON structure is shown below in Figures 13-16. The transmitted data for each UJ contain *origin_country*, *destination_country* and *workflow_type*, as well as a multilingual description and the workflow description itself as a list of phases (see Figure 16).

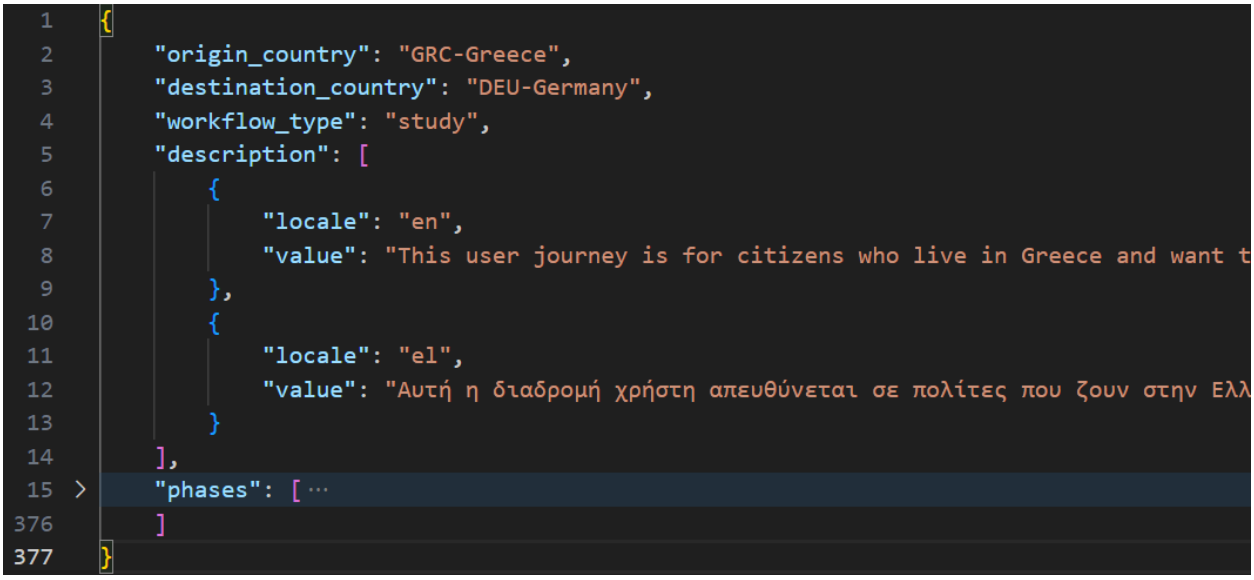


Figure 16: Part of a generated UJ orchestration description with focus on the general UJ information

Figure 17 shows the transmitted data per phase. The index of the phase in the list is passed in the value *order*. This was a direct request from the technical partners responsible for the UJSE. To obtain the assignment of the phase to the related graphical element in the UJM, the ID of the graphical element is transferred in *drawio_id*. This is followed by the multilingual *name* of the phase and a list of the assigned actions.

Figure 17: Part of a generated UJ orchestration description with focus on phase information

Figure 18 shows the data for each submitted action, which is structured similarly to the phase data. In addition to the *name*, each action can have a multilingual *description*. The services assigned to the action

are passed as a list in *touchpoints*. Finally, *is_mandatory* shows whether the action is optional or mandatory.

Figure 18: Part of a generated UJ orchestration description with focus on action information

Finally, Figure 19 shows the data that is passed for each service: in addition to the *drawio_id*, these are the *service_id* which the UJSE needs to call the service, a *service_url*, the assigned country (*location*) that provides the service (this was a direct request from the technical partners responsible for the Citizen Frontend), as well as a name and a description. The transmission of *name* and *description* will no longer be needed in the future, since this information will be read directly from the Service Catalogue.

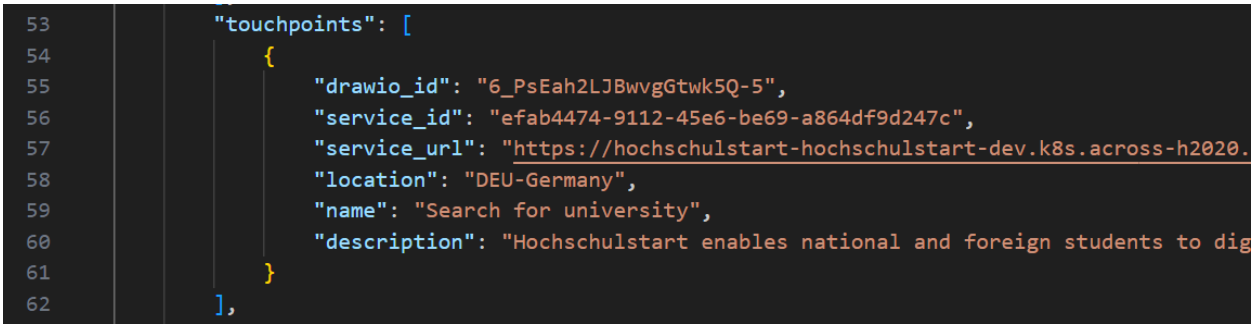


Figure 19: Part of a generated UJ orchestration description with focus on service information

2.6 UJMT - Baseline technologies

The UJMT uses different standard baseline technologies. The most relevant of these are listed below:

Representational state transfer (REST): architectural style for implementing hypermedia systems in the Web. In a narrower meaning that is used here, it provides a definition (rather guidelines) of stateless web APIs. Web APIs that conform to REST guidelines are called RESTful APIs. REST is not standardized per se, but it is well defined by papers and convention.

Structured Query Language (SQL): is a standardized programming language, which is mainly used to manage data in relational database management systems (RDBMS), like CockroachDB. With SQL it is possible to extract and organize User Journey in the UJ Storage.

JavaScript: is a programming language, it is a core technology of the Web, just as HTML or CSS. It conforms to the ECMAScript standard but comes in different flavours by different vendors. Nonetheless, a core JavaScript language definition is supported in all flavours, thus making the use of the language safe for our needs.

Rust: is a strongly typed programming language which emphasizes performance and security. The serde crate and variants like the yaserde crate allow easy serialization and deserialization which is used for efficiently translating the DRAWIO XML input into BPMN XML and JSON output in the UJMT Backend.

mxGraph: is a JavaScript library for browser-based interactive graph applications. The library is a fundamental part of Draw.io.

Business Process Modelling and Notation (BPMN): is a machine-readable graphical standard notation for describing business processes in a model. BPMN is maintained by the Object Management Group (OMG). The current version is BPMN 2.0, and the usual exchange format is XML.

CockroachDB: is a distributed SQL database management system. It is a scalable system, which is designed to run in the cloud and has a high fault tolerance. Thus, it is safe for our needs.

2.7 UJMT - Conclusions and Opportunities for Further Research

With the functionalities presented, the Final Prototype of the UJMT offers a basis of support for Modelling Experts who want to create and publish UJM in the ACROSS platform. The main objectives described in Section 2.1 were met and most of the requirements for the UJMT that arose in the project were implemented.

The Prototype represents a good starting point for future extensions and refinements. In particular, the stronger integration and use of the UJ storage should be mentioned, via which various UJ related information can be passed on to other components in the ACROSS platform.

The concrete UJM for the defined scenarios should be completed and published during the remaining project period. To benefit even more from the advantages of the central storage, potential users of the UJMT would also have to create and provide abstract workflows, which can be used as templates for future UJ creation.

In the implementation, there is potential in expanding the user feedback through validation and further restriction of the properties of the graphical elements. With further adjustments in these areas the usability could significantly improve.

Finally, during the project period, it should be determined which options are already available in the Draw.io base tool for collaborative UJ modelling and whether these parts of the implementation could be integrated into the implementation of the UJMT.

3 Virtual Assistant (VA)

The Virtual Assistant provides conversational interfaces (i.e. chat or speech interfaces using natural language) to the user-facing components of ACROSS, i.e. the Web and Mobile App. Thus, the ACROSS applications can be used traditionally by keyboard and mouse or touchscreen, but the VA empowers the user to control them through natural language as well.

3.1 VA - Objectives and Scope

The main technical objective of the Virtual Assistant is to make features of the ACROSS web and mobile apps controllable through conversational interfaces, i.e. natural language. This objective is motivated by two more general, non-technical objectives.

First**, ease of use and convenience**: Conversational control to many users seems more natural and convenient than using a classical "point and click" graphical user interface. Many users prefer to ask a chat-bot for information over having to click through a complex website for finding it. With a voice interface, there is the additional advantage of not having to use one's hands, which, apart from mere convenience, may considerably ease uses e.g. in non-standard environments or situations.

Secondly, **accessibility**: The advantages mentioned above are even more relevant for disabled citizens, who by utilizing this additional interaction channel can work around their impairments, so the barrier to use the ACROSS apps is removed. For instance, people with reduced vision can access the application through voice interaction. Similarly, there are analphabetic or dyslexic people or who are unable to read or write or uncomfortable with reading and writing, to whom a speech interface would also be strongly preferable. Or a person with a slight motor impairment that affects fine control of a pointing device while still allowing keyboard use might prefer to communicate with the app and the underlying services through a chat-bot style textual interface.

Even though this is not an objective in a technical sense, from which specific requirements could be derived, a third aspect should also be mentioned: Conversational interfaces enjoy broad popularity nowadays, both in their textual form (e.g. as chat-bots for customer contact purposes in online shops) and as voice assistant devices (in the consumer electronics space). Given that, making existing services accessible through natural or even spoken language can be expected to render these services more innovative and attractive in general.

These arguments, especially the latter one, continually gain further relevance with the ongoing generative AI revolution, exemplified by systems as OpenAI's and Microsoft's ChatGPT, or Google's Bard, which within a timeframe of just a few months gained millions of active users.

(More considerations regarding the large language models, or LLMs, that are the foundation of systems like ChatGPT, w.r.t the VA are documented in 3.7.)

3.2 VA - ACROSS Context

This section describes the context that has influenced the definition of the VA, both on the European level and within the ACROSS project.

3.2.1 Relevant European initiatives and legislation

Connecting Europe Facility – CEFDigital eTranslation service

Within the Digital Europe programme, the Connecting Europe Facility [15] has developed a set of Digital Service Infrastructures Building Blocks that can be reused in any European project to facilitate the delivery of digital public services across borders and sectors. Amongst these is the CEF eTranslation service [16] which is able to automatically translate formatted documents and plain text between any pair of EU official languages, as well as Icelandic and Norwegian. This mature and operational machine translation service can be directly invoked through an API and thus is an ideal foundation for the multilingual aspects of projects like ACROSS. Concretely, even though its integration is not yet covered within the initial design, the CEF Digital eTranslation service will serve as an important basis for the multilingual operation of the ACROSS Virtual Assistant in the future.

One of the general objectives pursued with the development of the Virtual Assistant is accessibility. Apart from being a laudable goal in general, this objective is also clearly rooted in European legislation. Concretely, the following EU regulations are strongly relevant - their content, but also their rationales (documented in the preambles) attest the considerable esteem the European Union and its member states devote to issues of accessibility, in particular with respect to IT products and services:

European Accessibility Act

This regulation [17], formally known as “DIRECTIVE (EU) 2019/882 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 17 April 2019 on the accessibility requirements for products and services” aims to “increase the availability of accessible products and services in the internal market and improve the accessibility of relevant information.” and specifies a list of product and services categories to which additional accessibility requirements and constraints will apply after a certain deadline (June 28, 2025). Apart from a number of hardware product categories, this list also includes software product categories, amongst other:

- consumer banking services

- e-commerce services

- "websites" and "mobile device-based services including mobile applications" if they are "elements of air, bus, rail and waterborne passenger transport services, except for urban, suburban and regional transport services"

The services mentioned above are very relevant to ACROSS and the Virtual Assistant, in that they clearly represent examples for private-sector services that can be needed within service workflows according to the ACROSS use cases.

Web Accessibility Directive

This directive [18], formally known as "Directive (EU) 2016/2102" aims to provide people with disabilities with better access to websites and mobile apps of public services. It obliges websites and mobile apps of public sector bodies to meet specific technical accessibility standards and includes provisions for regular monitoring of public sector websites and apps by Member States.

This regulation, too, is highly relevant to the ACROSS project and the Virtual Assistant, because it explicitly focuses on the web-based and mobile app offerings of the public sector, which clearly includes many of the services that can be needed within service workflows according to the ACROSS use cases. But notably, this category would also include a future, operational version of the ACROSS web and mobile app itself.

3.2.2 Approach and Relation to other Work Packages and Deliverables

The Virtual Assistant is part of the citizen front end part of ACROSS. It is closely coupled to the ACROSS web application in order to provide the aforementioned additional conversational control of the applications. The user journey service developed influences the Web UI, and thus, indirectly, the Virtual Assistant.

Because of the minimally-invasive integration approach used, which does not require in-depth modification of the UI it is used to control, the Virtual Assistant can be regarded mainly as an add-on to the ACROSS apps, instead of an interwoven ingredient.

As is requires no direct couplings with other ACROSS components, the VA builds on the output of the other work packages (notably requirements arising from the use cases work in WP2 and WP6), but has minimal impact in the reverse direction.

3.3 VA – Requirements

The Virtual Assistant covers some general ACROSS requirements and fulfils special requirements for the VA component. Those requirements are listed in this section.

For the Intermediate Prototype, the process of requirements collection has been overhauled and optimized throughout the entire ACROSS project, by tightening the collaboration between the use-case analysis and development work in ACROSS WP2 and WP6 on the one hand, and the software component design, development and integration work within ACROSS WP4 and WP5 on the other hand. Improved agile processes were established and more flexible collaboration tools were introduced. Within this much more effective framework, based on the initial requirements first documented in "D4.7 User Support Tools – Initial" [1], existing requirements were refined into more detailed and technical ones, and additional requirements were formulated, both based on the latest results of the use case work in WP2 and WP6, ad were documented in "D4.7 User Support Tools – Intermediate" [2]. (An earlier (coarse) requirements list conceptually remains in force but is not repeated in its entirety here; it can be found in [2]).

The list of Intermediate and Final Prototype requirements resulting from this process is not repeated here but is maintained on an ongoing basis using a kanban-style agile methodology in a collaborative issue management system (Trello).

3.3.1 Initial Requirements from WP5 and general ACROSS requirements

WP5 has gathered a first set of requirements for the whole ACROSS platform and ICT modules [19]. Some of these requirements are directly related to the VA while others are generic IT requirements or non-functional requirements. Those relevant to the VA are shown in this subsection. Req_03 is the foundational requirement from WP5 applying to the VA, whereas the remaining requirements shown below are general requirements to the components implemented in ACROSS, which also apply to the VA (to an appropriate extent).

Table 4 Requirements to the VA from WP 5

Id	Title	Description	Type	Category
Req_03	User Journey Chatbot implementation	A Multi-lingual Virtual Assistant API should be provided as a service. All the User applications should be able to connect to that API and benefit in their User Experience.	functional	Virtual assistant to guide the citizen

Req_07	DevOps Processes Setup	A full end-to-end pipeline of processes should be set up to ensure the successful integration, deployment, testing and delivery of the services. The DevOps processes, development and operations should be integrated into a single-minded entity with common goals: high-quality software, faster releases, and improved users' satisfaction.	non functional	Platform architecture and interoperability
Req_10	User Journey Experience	Citizen should be able to navigate in a straightforward clearly defined way through the whole process of the User Journey provided user experience. I would also like to have support during the steps of the moving abroad process.	non functional	Platform architecture and interoperability
Req_12	Scalability	The ACROSS platform should be designed to be scalable in terms of computational load, number of users accessing applications and amount of data storage. In particular, the platform should be able to scale horizontally (e.g. add more nodes to a computational network)and vertically (e.g. add resources such as Memory, CPU to a single node in a system).	non functional	Platform architecture and interoperability
Req_19	Reliability and Integrity	The implementation of ACROSS should follow open standards and use well-known and widely accepted technologies in order to ensure integrity. The ACROSS platform has to be reliable assuring integrity of the components/tools that are part of it.	non functional	Platform architecture and interoperability

Req_22	Privacy and Data Protection	The ACROSS platform has to be compliant with the EU legislation regarding privacy and data protection. It should adopt all the necessary technologies, standards and methods to protect privacy of the users of the platform services and to secure stored information that could be considered private.	non functional	Security and Privacy
Req_30	Open source	I want the ACROSS reference architecture to reuse already available open source solutions and only create or improve those aspects that are not covered by the existing solutions	non functional	Platform architecture and interoperability
Req_33	Accessibility	The front-ends of the system should comply with the current Web Accessibility Directives and in particular with EN301549 (included in WCAG-2.1)	non functional	Web&Mobile applications
Req_35	Usability and adaptability	The provided solutions in the platform should be user-friendly and easy to use and should be multilingual. No piece of text that might be displayed to a user shall reside in source code and solution and user should be able to select the preferred language. The implementation of the system should follow open standards and use well-known and widely accepted technologies in order to ensure ease of use.	non functional	Platform architecture and interoperability
Req_36	Minimal browser support.	The component user interface (where available e.g. dashboards, forms, etc.) should provide support for the wide range of widely used browsers.	non functional	Web&Mobile applications

In addition, several general ACROSS requirements were identified from project documents or in discussions with project partners, which the Virtual Assistant shall abide to.

REQ-GEN-1. As a developer in ACROSS I want the Virtual Assistant to be easy to integrate with the other software and infrastructure components of ACROSS during development and operation.

REQ-GEN-2. As a developer in ACROSS I want integration with the Virtual Assistant to not hamper the functionality of other ACROSS components, in particular the Web or Mobile Application.

REQ-GEN-3. As a developer in ACROSS I want the Virtual Assistant to use the Apps, and, indirectly, the User journey Services Engine and the results of the User Journey Modelling Tool in order to access the elementary Public Services.

3.3.2 Key Performance Indicators for Initial Requirements from WP5

In order to assess the degree to which the requirements will be fulfilled by the UJMT implementation, the following Key Performance Indicators (KPIs) have been defined[4]:

Table 5: Proposed Key Performance Indicators (VA)

Id	Title	Proposed KPI	Unit
Req_03	User Journey Chatbot implementation	Ratio of ACROSS UI (i.e. Citizen WebApp) features accessible through VA vs. total UI features	%
Req_07	DevOps Processes Setup	Degree to which VA is integrated into the DevOps Processes of ACROSS and, concretely the ADCROSS Citizen WebApp	%
Req_10	User Journey Experience	Does the VA support the user in planning and executing their user journeys?	Y/N
Req_12	Scalability	Does the VA adhere to ACROSS architectural principles enabling scalability?	Y/N

[4] Meanwhile, project-wide success criteria have been defined in D6.2 “Use Case Evaluation and Impact Assessment” that supersede these KPIs.

Req_19	Reliability and Integrity	Does the implementation of the VA follow open standards and use well-known and widely accepted technologies, thereby assuring integrity of the components/tools that are part of it?	Y/N
Req_22	Privacy and Data Protection	Is the VA compliant with the EU legislation regarding privacy and data protection.	Y/N
Req_30	Open source	Does the VA reuse already available open source solutions and only create or improve those aspects that are not covered by the existing solutions?	Y/N
Req_33	Accessibility	Does the VA retain the ability of the front-ends of the system to comply with the current Web Accessibility Directives and in particular with EN301549 (included in WCAG-2.1)?	Y/N
Req_35	Usability and adaptability	Is the provided solutions user-friendly, easy to use, and multilingual?	Y/N
Req_36	Minimal browser support.	Does the VA provide support for the wide range of widely used browsers?	Y/N

3.3.3 Initial VA – Requirements from WP2 (ACROSS New Governance Model) and WP6 (Use cases)

REQ-VA-1. As a user of the Virtual Assistant I want the Virtual Assistant to provide access to the steps of the process defined by the User Journey in a sufficiently functionally similar manner as when traditional interaction would be used.

REQ-VA-2. As a user of the Virtual Assistant I want the Virtual Assistant to communicate and to steer the sequence of individual steps that need to be executed for the individual user journeys.

3.3.4 Initial VA-specific requirements

3.3.4.1 Conversational UI control

REQ-VA-3. As a user of the Virtual Assistant I want to have additional *conversational* control over the application via natural language and speech for a convenient and accessible User Journey Experience.

REQ-VA-4. As a user of the Virtual Assistant I want the Virtual Assistant to be closely integrated with the ACROSS application UI.

REQ-VA-5. As a user of the Virtual Assistant I want the Virtual Assistant to provide access to an appropriate part of the features the UI is providing.

REQ-VA-6. As a user of the Virtual Assistant I want interaction through the Virtual Assistant to be as equivalent to traditional UI interaction as possible (with a conversational interaction mode) regarding input, navigation, and output.

3.3.4.2 Multi-language support

REQ-VA-7. As a user of the Virtual Assistant I want the Virtual Assistant to have multilingual capability.

3.3.4.3 Customization for Services and Workflows

REQ-VA-8. As a user of the Virtual Assistant I want the Virtual Assistant to be customizable to the specific services and workflows.

3.3.4.4 Other VA-specific requirements

REQ-VA-9. As a developer and user in ACROSS I want the Virtual Assistant to work without slowing down the browser, server, and general user experience (beyond the extent that is inevitable due to the limited interaction bandwidth of natural language and speech)

3.3.5 Requirements from WP2 (ACROSS New Governance Model) and WP6 (Use Cases) for Intermediate Version

REQ-VA-10. Get information about user journey status: As a user I want to ask the virtual assistant about the current status of my journey to see what steps still need to be processed by me.

REQ-VA-11. Change CWA WebApp-UI-Settings: As a user I want to change platform settings, font size and display language via the virtual assistant

REQ-VA-12. Fill form of workflow step: As a user I want to be assisted by the VA to fill out at an application form for a workflow step

REQ-VA-13. Get information about services: As a user I want to find information about Services i.e. Search for Services based on keywords, Information about specific service.

REQ-VA-14. Switch topic during a form filling process: As a user I want to get assisted by the VA while I'm doing my user journey within ACROSS, so that I have all relevant information about services.

3.3.6 Implementation Matrix – VA

Table 6: Implementation Matrix (VA)

	Initial Version	Intermediate Version	Final Version
Requirements Initial Version			
REQ-VA-1	Concept (UJ steps not yet individually accessible in CWA)	Implemented: Individual UJ steps accessible through WebAssist 3.5.1) as far as accessible in CWA	Further extension
REQ-VA-2	(steering not yet accessible in CWA)	Implemented: Steering possible through WebAssist 3.5.1) as far as accessible in CWA	Further extension
REQ-VA-3	Conversational control available for limited subset of CWA features through WebAssist 3.5.1)	Implemented: Conversational control available through WebAssist 3.5.1) for most features of CWA	Further extension
REQ-VA-4	Partial integration based on UIC (3.4.2.1)	Implemented: Full, close integration based on UIC (3.4.2.1)	Adjustments
REQ-VA-5	Access available for limited subset of CWA features through WebAssist 3.5.1)	Implemented: Access available for large subset of CWA features through WebAssist 3.5.1)	Further extension
REQ-VA-6	Interaction equivalency for limited subset of	Implemented: Interaction equivalency for large subset of interactions based on WebAssist 3.5.1)	Further extension

	interactions based on WebAssist 3.5.1)		
REQ-VA-7	Mutilinguality designed but not yet implemented,	Implemented: Partial Integration of Machine Translation (MT)component (3.5.4) and new voice interface components ASR and TTS (3.5.5)	Extensions/Full integration of MT, ASR, TTS
REQ-VA-8	Substantial functionality hardcoded for prototype, very weak customizability.	Implemented: Declarative customization of most aspects of VA functionality based on WebAssist 3.5.1)	Adjustments
REQ-VA-9	Principal viability of the VA integration approach w.r.t. REQ-VA-9 has been demonstrated, but fulfilment of the requirement was not yet analysed in-depth for all VA subcomponents.	Implemented: All new developments and optimizations in the VA and its subcomponents have been strictly aligned with this requirement. In particular, REQ-VA-9 and a strong focus on UX was strictly applied during development of the new subcomponents Q&A chatbot (3.5.2) TTS, ASR (3.5.5) and MT (3.5.4) and during selection of open-source packages supporting these components.	Adjustments
Requirements Intermediate Version			
REQ-VA-10	Concept (UJ steps not yet individually accessible in CWA)	Implemented: Individual UJ steps accessible through WebAssist 3.5.1) as far as accessible in CWA	Extensions/Adjustments
REQ-VA-11	PoC	Implemented: All WebApp-UI-Settings accessible through VA	Extensions/Adjustments

		(3.5.1)	
REQ-VA-12	Concept	Implemented: Form filling accessible through WebAssist 3.5.1) as far as accessible in CWA and practical e.g. upload fields cannot be VA-controlled because the user needs to interact with OS functionality outside the browser)	Adjustments
REQ-VA-13	-	Initial coupling between new VA subcomponent Q&A-Chatbot and Service Catalogue implemented (3.5.2)	Complete integration of Q&A Chatbot (3.5.2) into VA
REQ-VA-14	-	Concept for conversational multiplexing tested (3.5.3)	Implementation & integration of conversational multiplexing (3.5.3)

3.4 VA - Design

3.4.1 VA - Architecture

The following figure provides an overall view of the main subsystems and modules of the VA and how they collaborate with the different parts of the ACROSS platform.

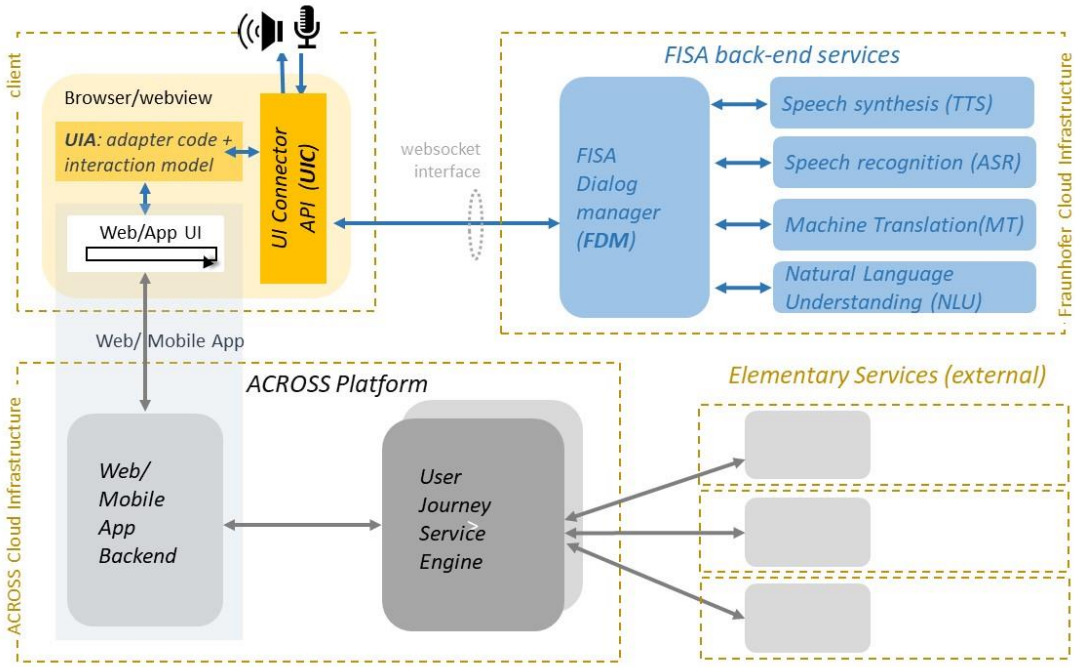


Figure 20: Subsystems and modules of the VA and their collaboration with the ACROSS platform (orange: front-end components, blue: back-end services)

3.4.2 VA - Modules

The ACROSS Virtual Assistant consists of two modules: The Web User Interface (UI) Connector (UIC) and the WebUI Adapter (UIA). It also collaborates with the FISA Dialog Manager (FDM) - this component is not developed within ACROSS but is background material[5]. Nevertheless, it will be summarily described below for completeness. Also, the protocol interface connecting UIC and FDM is described in 3.4.2.4 .

3.4.2.1 VA UI Connector (UIC)

The UI Connector (UIC) connects the UI and the backend bi-directionally. It provides a generic, application independent library for connecting a WebUI with the FISA Dialog Manager (FDM) in order to add a conversational (chat and/or speech) interface to the WebUI. This way, every WebUI can be connected to the Dialog Manager, and the same holds for the ACROSS mobile App (as its development is based on the

[5] Relying on background material that currently is not open-source inevitably induces risks, concretely regarding ACROSS WP5 requirement Req_29 [19]. On the plus side, reliance on background material makes sense economically though, in that more functionality can be achieved with the given project resources. This version of this deliverable also documents the interface used by the FDM. That way, switching to another dialog manager implementation would be possible if need be, and any vendor lock-in risk is mitigated.

same web technology as the ACROSS Web App, essentially mirroring it). This functionality fulfils requirement REQ-VA-4.

A WebUI to be controlled by the Virtual Assistant is extended with a WebUI Adapter, which through the UIC communicates with the Dialog Manager (FDM). By bridging the gap between the WebUI and the UIC, the WebUI Adapter effectively makes the WebUI controllable from the FDM.

Technically, the UIC creates and maintains a websocket connection to the FDM. For each conversational interaction to execute, the UIC receives a request containing an interaction state (also called interaction point) ID from the WebUI Adapter, and delivers it to FDM through the websocket connection using a dedicated wire protocol. (For maintaining the websocket connection, the VA implementation employs a Socket.IO open-source component, which reduces the implementation effort and increases resilience against important failure modes.)

For the other direction, the UIC decodes messages from the FDM received through the websocket connection, and forwards them to the WebUI Adapter. This includes input values or navigation requests recognized from user utterances within the ongoing conversation.

The implemented FISA wire protocol is a bi-directional streaming protocol and uses, amongst others, command, text, and audio packets based on a protocol schema (specified using features of the Socket.IO implementation). It is documented in 3.4.2.4.

The UIC is also responsible for communicating with the client system audio resources (microphone and speaker) available through the browser or mobile webview. The UIC thus provides all audio input and output handling (as far as needed within the client for voice interfaces). In this context, it also addresses browser autoplay restrictions (to the maximum extent allowed by the browser vendors).

3.4.2.2 VA WebUI Adapter (UIA)

The WebUI Adapter (UIA) is the “remote control” of the WebUI. It adapts and couples the WebUI to the UI Connector, which in turn enables communication with the FDM backend.

The UIA implements detailed, application specific adapter code for a given WebUI. This way it enables ways for remotely controlling the WebUI in both directions: sending user actions and receiving according signals; as well as sending interaction states and receiving spoken output for the user.

In addition to implementing application-specific details of how to control the *individual WebUI elements* (e.g. code for filling specific input fields), the UIA is also responsible for letting the UIC *control the WebUI*

on the macroscopic (interaction workflow[6]) level, namely by supporting *navigation* within the WebUI in a conversational manner.

In traditional (screen/keyboard/pointing device) interaction, the user's navigation needs are covered by web browser functionality: The user may at any point select (click into) a specific input field in order to use it next, thus making this field obtain keyboard input focus. In form-based UIs, the browser also often offers the option of just pressing the TAB key to go to the next input field, or SHIFT-TAB to go back to the previous one. However, this simplified navigation only works for well-structured web pages. In a conversational UI interaction mode, it is essential to always have a reliable basis for interaction workflow-level navigation, especially so since the fall-back option of just explicitly selecting the field the user has decided to use next is not practical (because there is no pointing device for instantaneously expressing this intent in two dimensions).

Hence, the WebUI adapter also has to provide a basic navigation model of the WebUI to the UIC. This model (also called the Navigator) must be able, at any current interaction point within the interaction workflow, to determine all "target" interaction points the user might want to navigate to from here. I.e., the *next and the previous interaction point*, the *next and the previous page or screen*, the *first and last interaction point* (on a page/screen), and all "*jump targets*" reachable from the current page/screen, such as help pages/screens or pop-ups.

Whenever the UIA has determined the user's intent to navigate to a new interaction point, this interaction point is passed on to the FDM through the UIC. The backend then starts a conversational interaction for that interaction point, e.g. by generating a system utterance to be presented to the user. For example, the system utterance might request the user to utter input data for the input field associated to this interaction point. This system utterance is then passed through the websocket connection to the UIC where it is shown/played to the user.

The following user utterance is streamed back by the UIC to the FDM, where its content is analyzed.

When the FDM has recognized a valid input value in the user's utterance, an appropriate command is streamed to the UIC, which triggers execution of application-specific code within the UIA for value assignment to the input field. Or, if the user responded with a navigation request, a specific command expressing that intent is sent from the FDM to the UIC, which then invokes the navigation model part of the UIA (also called the Navigator) to make the navigation actually happen in the WebUI.

3.4.2.3 FISA Dialog Manager (FDM)

The FISA Dialog Manager (FDM) is part of the aforementioned back-end that receives user input or interaction state, analyses textual user utterances and/or speech and generates system utterances (text and/or speech) or commands accordingly.

In detail, the FDM enables conversational control for WebUIs by handling all dialog management needs for conversationally controlled interactions. It provides NLU / intent recognition by different means, e.g. by pattern matching or constraint rules. The FDM enables data input recognition (also called "slot filling"), i.e. assigning values recognized in user input texts/utterances to "input slots" corresponding to input elements of the WebUI. The FDM also provides conversational response generation, i.e. creation of natural language output (system utterances) in text/speech form.

The FDM is also able to support spoken-language interfaces, by delegating to sub-services for speech recognition or speech synthesis. These components will not require separate, direct connections to the front-end – all data communications for them will be channelled through the FDM, where they are processed by its Event Handler component.

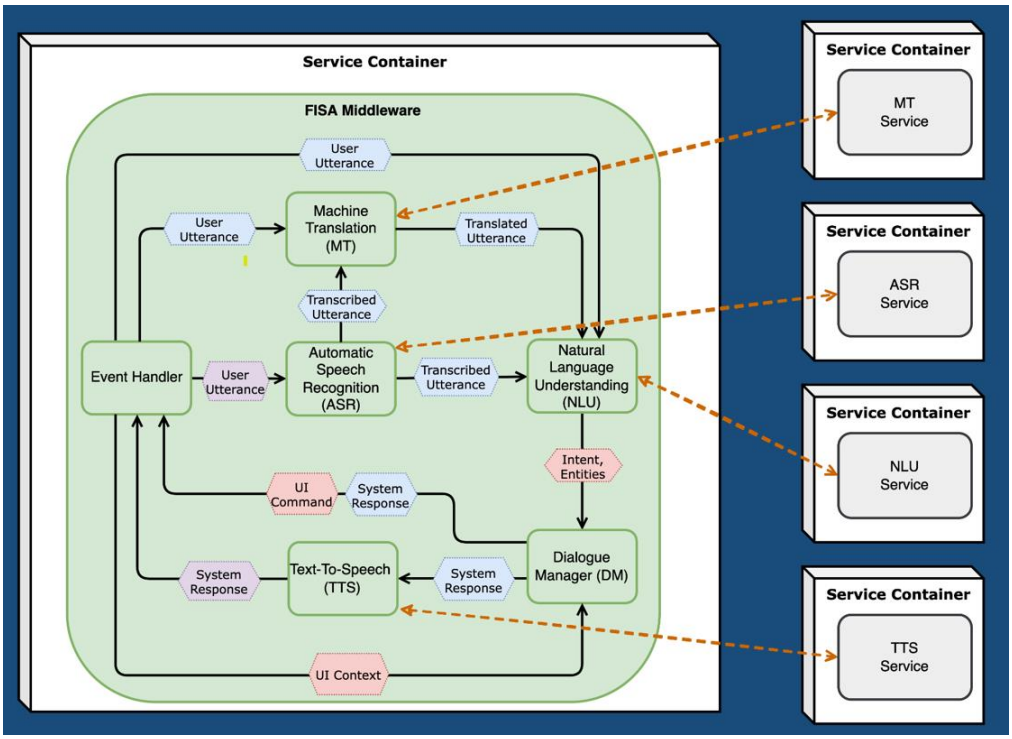


Figure 21: Overview of FISA Dialog Manager (FDM) internal architecture

In order to fulfil its task, the FDM receives messages from the WebUI via the UIA and UIC. The messages must conform to the FDM protocol. Based on these messages, the FDM reacts to conversational user input

(i.e. textual or speech utterances) during an ongoing conversational interaction, and to client-triggered interaction point changes (which abort any still-ongoing conversational interaction and start a new one). Interaction point changes may originate from the Navigator part of the adapter code (in reaction to an FDM message originally caused by a user utterance, see below), or from traditional screen interactions of the user (because the user may want to switch back-and-forth between conversational and traditional interaction).

The FDM sends messages, again conforming to the FDM protocol, to the WebUI via UIC and UIA in order to provide answer texts/utterances, provide value assignment requests resulting from user intents, communicate WebUI navigation requests resulting from user intents, and control the audio input (i.e. microphone activation) and output (playback of system utterances) behaviour to occur on client-side.

The FDM has restricted state context, it only "sees" either a single interaction point or a small collection of interaction points, e.g. the one corresponding to all interaction elements on a single page or screen. The FDM does not persist information beyond the end of the conversational interactions being executed for this interaction point (or set of points).

3.4.2.4 UIC-FDM interface documentation

The following subsections documents the current structure of the wire protocol used on the Websocket connection between the UI Connector (UIC) and the FISA Dialog Manager (FDM).
All Event names are prefixed with the sender:

- `UIC\_\*` = Events emitted from UIC to Middleware
- `MW\_\*` = Events emitted from Middleware to UIC

3.4.2.4.1 Default Events

3.4.2.4.1.1 UIC_CHAT

Transmit a user chat message to the server.

- **When**: When the user sends a chat message
- **Data**:
  - Chat message [string]
  - Selected pipe parameters [object]. **Important**: choice of MT_MODEL

3.4.2.4.1.2 UIC_AUDIO_START

Announce to the server the start of a user audio stream.

- **When**: When the user starts the voice assistant (or presses the 'REC' button in UIC chat)
- **Data**:

- Selected pipe parameters [object]. **Important**: choice of ASR_MODEL and MT_MODEL

3.4.2.4.1.3 UIC_AUDIO_CHUNK

Transmit one chunk of a user audio stream.

- **When**: Constantly between UIC_AUDIO_START and UIC_AUDIO_STOP
- **Data**:
  - Audio Chunk [bytes]

3.4.2.4.1.4 UIC_AUDIO_STOP

Announce to the server the end of a user audio stream.

- **When**: When the user stops the voice assistant (or presses the 'REC' button in UIC chat)
- **Data**: None

3.4.2.4.1.5 MW_READY

Announce to the client that the server is ready.

- **When**: Initially, after successful connection and authentification of the client
- **Data**: Pipeline parameters [object]: Set of available parameters (choices, toggles)

3.4.2.4.1.6 MW_LOG

Inform the client about events taking place in the server.

- **When**: Anytime during an active connection.
- **Data**:
  - message [string]: Description of an event in the server
  - level [string]: Importance of the event (INFO, DEBUG, WARNING)
  - (optional) data [object]: Data associated with the event

3.4.2.4.1.7 MW_ERROR

Inform the client about an error in the server.

- **When**: When an error occures in the server.
- **Data**:
  - message [string]: The error message
  - (optional) trace_id [string]: ID of a trace file that has been generated

- (optional) trace_url [string]: URL to a trace file that has been generated
- (optional) pod [object]: Pod with component outputs that produced this error

3.4.2.4.1.8 MW_CHAT

Transmit a chat message to the client.

- **When**: When the pipeline outputs a chat message.
- **Data**:
  - message [string]: The chat message
  - (optional) trace_id [string]: ID of a trace file that has been generated
  - (optional) trace_url [string]: URL to a trace file that has been generated
  - (optional) pod [object]: Pod object with component outputs that produced this chat message

3.4.2.4.1.9 MW_COMMAND:

Transmit a UI command to the client.

- **When**: When the pipeline outputs a UI command.
- **Data**:
  - command [object]: The UI command
  - (optional) trace_id [string]: ID of a trace file that has been generated
  - (optional) trace_url [string]: URL to a trace file that has been generated
  - (optional) pod [object]: Pod object with component outputs that produced this command.

3.4.2.4.1.10 MW_STATUS

(This event was newly introduced in the Final Release.)

Inform the UIC about status changes in the Middleware. This also includes information about the current active WebAssist mode and connectivity information from language services.

- **When**: Collection of different events that all inform about a middleware status change
- **Data**:
  - (optional) webassist [boolean]: Bool that represents the active Dialogue System (and mode)
  - (optional) ssm [object]: Solid State Model – a representation of all interactable parts in the UI

Web-Assist events

3.4.2.4.1.11 UIC_GET_PSSM (Acknowledged event)

Request the PSSM for the current URL from the server.

- **When**: When a URL change occurs in the frontend.
- **Request Data**:
  - url [str]: The new frontend URL
- **Response Data**:
  - pssm [object]: Subset of the SSM

3.4.2.4.1.12 MW_GET_CONTEXT (Acknowledged event)

Request the current UI context from the client.

- **When**: When the WebAssist component tries to parse a user message.
- **Request Data**: None
- **Response Data**:
  - current_state [string]: ID of the current state
  - states [object]: Set of currently visible and interactable states

3.4.3 VA - Interfaces and Collaborations

3.4.3.1 WebUI - UIA Interface

The WebUI – UIA interface connects the Web/Mobile App with the Virtual Assistant, namely its WebUI Adapter (UIA).

It is an external interface.

The interface is defined in terms by one or more of the following mechanisms:

- specific JavaScript (JS) API calls of the UI framework in which the WebUI is implemented
- specific URLs for opening specific pages or screens of the WebUI
- selected concrete DOM Elements belonging to the WebUI and uniquely identified by a suitable mechanism (e.g. tags)
- selected specific DOM event bindings, for selected events to which the VA is intended to react

This interface is application/WebUI-specific.

3.4.3.2 UIA - UIC Interface

The UIA - UIC Interface connects WebUI Adapter (UIA) and UI Connector (UIC).

It is an internal interface.

The interface is an asynchronous JavaScript API provided by the UIC. Method calls for actions originate from the UIA. Callbacks defined within the UIA for actions originate from the UIC (and are triggered by messages arriving from the FDM).

This interface is documented by UIC JSdoc.

3.4.3.3 UIC – FDM Interface

The UIC – FDM Interface connects UI Connector (UIC) and FISA Dialog Manager (FDM).

It is an internal interface.

The interface provides a websocket connection using a specific FDM wire protocol. This interface and protocol are documented within the FDM.

The UIC is responsible for the connection life cycle, i.e. (re-)creating a connection whenever it is needed and destroying the connection when no longer needed.

There are security considerations for the interface: The FDM needs to accept connections from any browser/mobile device that executes the ACROSS WebUI. In order to protect the websocket server e.g. from DOS attacks, a suitable (fixed-key or time-based) authentication mechanism is employed.

3.4.3.4 FDM – UJSE Collaboration

*(This collaboration was newly introduced in the Final Release.)*The FDM – UJSE collaboration connects the Q&A Chatbot component of the FDM with the User Journey Service Engine.

This collaboration is used within the Q&A Chatbot component and consists of simple HTTP-API calls to obtain Workflow-specific service data from the UJSE API. This data is needed so that the Q&A Chatbot can constrain its knowledgebase on the currently necessary services only.

3.5 VA – Implementation Description

3.5.1 WebAssist core functionality

The Implementation of the Virtual Assistant is based on the current implementation of the ACROSS Web/Mobile App. It demonstrates the use of the Virtual Assistant in substantial areas of the ACROSS Web/Mobile App. The functionality described in the following is the foundation for the VA to fulfil requirements REQ-VA-1, REQ-VA-2, REQ-VA-3, REQ-VA-5, REQ-VA-6, REQ-VA-8, REQ-VA-10, REQ-VA-11, REQ-VA-12 and REQ-VA-13.

The VA makes a substantial set of interactions offered by the ACROSS Web/Mobile App prototype controllable through a conversational (natural language) interface. It focuses on conversational navigation for interactions applying to a) general control of the ACROSS Web/Mobile App and b) control of selected user journey-relevant interactions offered by the ACROSS Web/Mobile App.

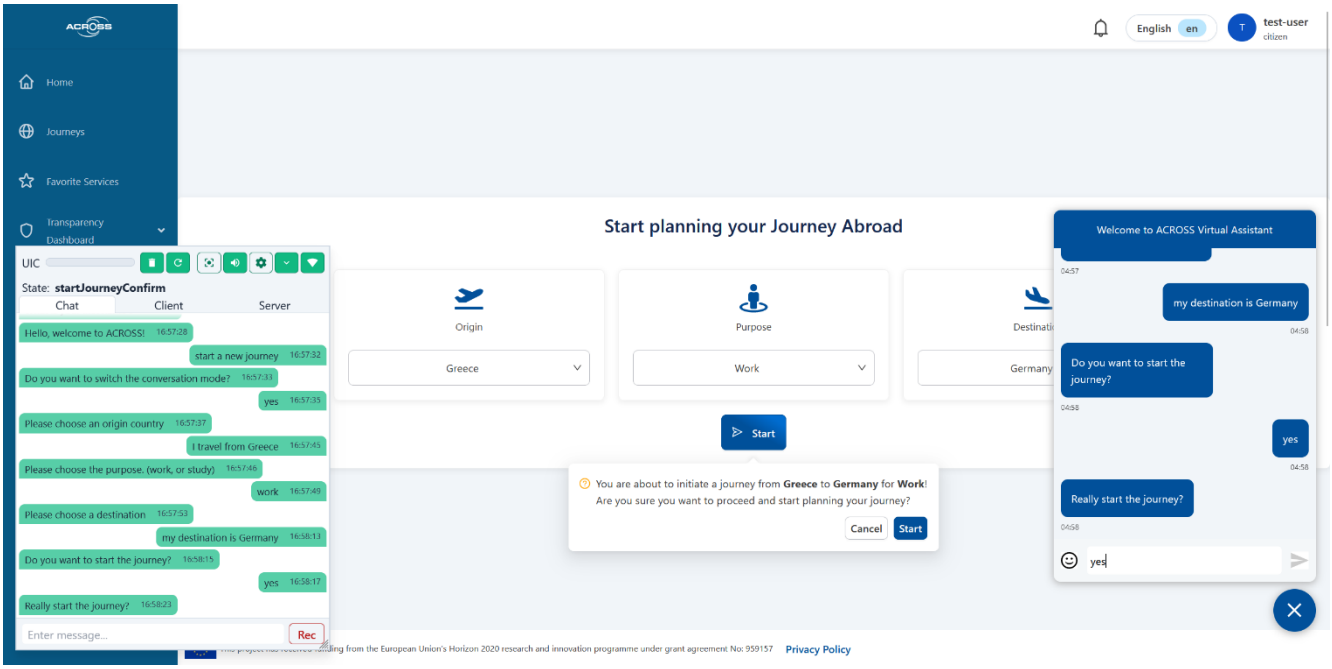


Figure 22: Screenshot: Conversational interaction (WebAssist) with the ACROSS WebApp through the VA

In order to achieve these abilities for the addressed interactions, the Virtual Assistant has been implemented as follows:

On the front-end side,

- All WebUI elements that have been selected for VA control have been marked with unique VA-IDs such that the VA can identify them within the DOM

```
110         <CustomCardHeader title={t('dashboard.planning')} />
111         <S.WrapperRow gutter={[20, 20]} justify="start">
112           <Col xs={24} sm={24} lg={8} xl={6}>
113             <Motion delay={0.25} transitionEnd>
114               <S.CardWrapper
115                 bordered
116                 padding={20}
117                 title={
118                   <Space direction="vertical" align="center">
119                     <S.MainIcon>
120                       <FaPlaneDeparture />
121                     </S.MainIcon>
122                     {t('dashboard.origin')}
123                   </Space>
124                 }
125               >
126                 <Select
127                   id='origin-country'
128                   placeholder={t('dashboard.selectOrigin')}
129                   loading={loading}
130                   onChange={(val) => setOrigin(val as string)}
131                   options={origins}
132                   allowClear
133                   width="100%"
134                   value={origin}
135                 />
136               </S.CardWrapper>
137             </Motion>
138           </Col>
```

Figure 23: Elements of the React.js-based UI of the ACROSS Web/Mobile App are marked with IDs in order to expose them to VA control

- UI Adapter (UIA) code was written for coupling the selected WebUI interactions with the UI Connector.

```
213  const toggleSettingsDropDown = () => {
214    const settingsDropDown = document.querySelector('#profile_button')?.childNodes[0];
215    if (settingsDropDown) {
216      (settingsDropDown as HTMLInputElement).click();
217    }
218  };
219
220  const languageSetter = ({}={}, {}={}, value: string) => {
221    const lang = getCountryCode(value);
222
223    toggleSettingsDropDown();
224
225    const languageButton = document.querySelector('#language_button');
226    if (languageButton) {
227      (languageButton as HTMLInputElement).click();
228    }
229
230    const li = document.querySelector(`li[data-menu-id='language-${lang}']`);
231    if (li) {
232      (li as HTMLInputElement).click();
233    }
234
235    toggleSettingsDropDown();
236
237    let uicLanguage = lang;
238    if (uicLanguage == 'gr') uicLanguage = 'el';
239    UIConnector.setOptions({ userLang: uicLanguage });
240    if (UIConnector.m.protocolClient.connectionState === 'ready') {
241      UIConnector.stop();
242    }
243    UIConnector.start();
244  };
```

Figure 24: UI Adapter (UIA) code example: VA-controlling the country selection list by (223) opening the drop-down list, (230) clicking the correct entry, and (235) closing the drop-down list

- The ACROSS Web/Mobile App WebUI components, their dependencies (i.e. UI framework packages) and the VA components UIA and UIC have been integrated into a single ACROSS front-end subsystem which is executed together in the browser/webview.

```
services > webapp-ui > src > TS App.tsx > ...
20  import type { AuthClientTokens } from '@react-keycloak/core';
21  import { Toaster } from 'react-hot-toast';
22  import { notificationLightColorsTheme } from './styles/themes/light/lightTheme';
23  import { notificationDarkColorsTheme } from './styles/themes/dark/darkTheme';
24  import { notificationController } from './controllers/notificationController';
25  import { useTranslation } from 'react-i18next';
26  import { Locale } from 'antd/lib/locale-provider';
27
28  import UIConnector from './@uic';
29  UIConnector.toggleChat();
30  UIConnector.toggleChat();
31
32  const App: React.FC = () => {
33    const { language } = useLanguage();
34    const { t } = useTranslation();
35    const theme = useAppSelector((state) => state.theme.theme);
36    const keycloakProviderInitConfig: KeycloakInitOptions = {
```

Figure 25: Importing the UI connector

This extended ACROSS front-end Web/Mobile App front-end communicates both with the standard ACROSS back-end services and with the FDM.

On the back-end side, the necessary FDM customizations have been done in order to support all selected interactions as well as meta-interactions (e.g. for navigation). Now most essential aspects of VA functionality need no longer be hard-coded but can be declaratively customized by YAML files:

- An interaction state model was specified for all selected interactions, associating the selected WebUI elements (referenced by their VA-IDs) with conversational prompt texts and conversational intents according to their type

```yaml
- id: "choosePurpose.out"
  content_id: "travel.choosePurpose.message"
- id: "choosePurpose.in"
  selector: "#purpose"
  type: radio
  rules:
    - intents:
        - webassist_choose_work
      cmd:
        name: set_value
        state: "choosePurpose"
        value: "Work"
    - intents:
        - webassist_choose_study
      cmd:
        name: set_value
        state: "choosePurpose"
        value: "Studies"

- id: "chooseDestination.out"
  content_id: "travel.chooseDestination.message"
- id: "chooseDestination.in"
  selector: "#destination-country"
  type: radio
  rules:
    - intents:
        - webassist_choose_destination
      entities:
        - name: country
      cmd:
        name: set_value
        state: "chooseDestination"
        value: country
    - intents:
        - webassist_form_GPE
      entities:
        - name: GPE
      cmd:
        name: set_value
        state: "chooseDestination"
        value: GPE
```

Figure 26: Interaction state model (cutout)

- To improve multilingual interaction, we added pre-translated conversation-prompts to the state model. This gives us the ability to define VA-prompts in different languages in a separate configuration file. Each prompt is identified with a string and can then be associated to a VA-ID inside the interaction model configuration file. For example, the Output for the state: “choosePurpose” (Figure 26, Line 29) is defined with the content-id: “travel.choosePurpose.message” in Figure 27, line 9.

```
9     "travel.choosePurpose.message": {
10        "de": "Bitte wählen Sie den Verwendungszweck aus. (arbeiten oder studieren)",
11        "en": "Please choose the purpose. (work, or study)",
12        "el": "Επιλέξτε το σκοπό. (δουλειά ή σπουδές)",
13        "lv": "Lūdzu, izvēlieties mērķi. (strādāt vai mācīties)"
14    },
15
16    "travel.chooseDestination.message": {
17        "de": "Bitte wählen Sie ein Ziel aus",
18        "en": "Please choose a destination",
19        "el": "Επιλέξτε έναν προορισμό",
20        "lv": "Lūdzu, izvēlieties galamērķi"
21    },
22
23    "travel.startJourney.message": {
24        "de": "Möchten Sie die Reise beginnen?",
25        "en": "Do you want to start the journey?",
26        "el": "Θέλετε να ξεκινήσετε το ταξίδι;",
27        "lv": "Vai vēlaties sākt ceļojumu?"
28    },
```

Figure 27: Config file for pre-translated VA-prompts (cutout)

- For all interactions to be controlled conversationally, textual example user utterances have been defined, which serve as NLU training material (for the ML-based conversational intent recognition that the FDM provides). Note that the user does not need to use the exact same wording as in the examples – the neural intent recognition is able to identify the correct intent from user utterances that are syntactically different but semantically similar.

```
- intent: webassist_explore_journeys
  examples: |
    - Explore Journeys
    - Explore my Journeys
    - Show my Journeys
    - Show me my Journeys
    - Show me my Journeys please

- intent: webassist_change_language
  examples: |
    - change the language to [german](language)
    - change the language to [greek](language)
    - change the language to [latvian](language)
    - change the language to [english](language)

- intent: webassist_start_journey
  examples: |
    - Start journey
    - I want to travel
    - Let's go
    - let's go
    - Let's begin
    - lets go
    - lets start a journey!
    - lets go for a journey
```

Figure 28: Intent definition with training samples (cutout).

- The FISA framework has been configured to integrate and connect all needed FDM subcomponents and services

3.5.2 Q&A Chatbot (newly integrated in Final version), coupling with ACROSS Service Catalogue

The ”WebAssist” functionality of the VA (which enables conversation control of the ACROSS Web/Mobile App’s WebUI) is the VA’s core functionality. However, within the ACROSS project, requirements appeared that could not be satisfied within this narrow framework – most importantly, to allow the VA to answer questions that are not directly associated to the WebUI and its current state. To this end, work has been done towards creating a new subsystem within the VA, a so-called “Q&A chatbot”, which is equipped to deal with such more general conversational requests issued by the end-user.

The Q&A Chatbot has been built based on the open-source RASA conversational AI framework [20] and is already able to answer simple conversational information queries.

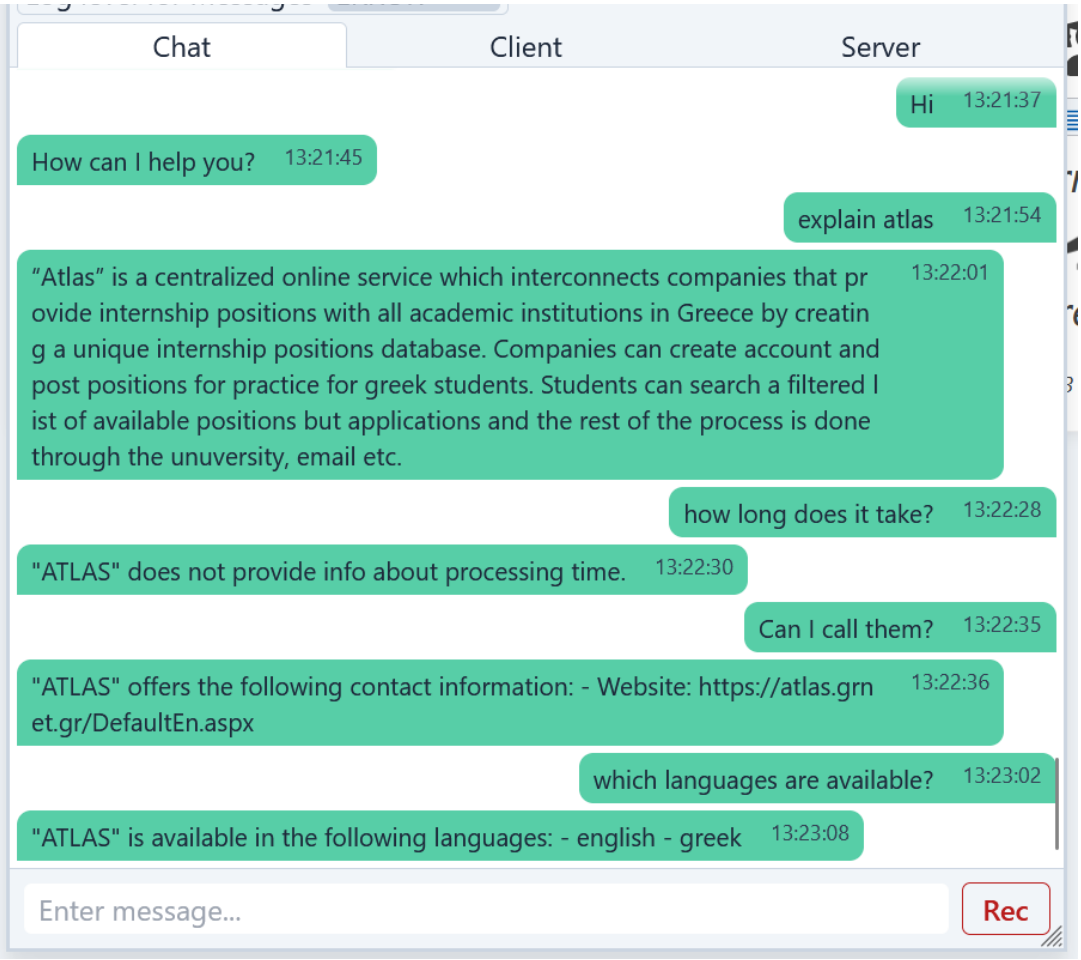


Figure 29: Session with the ACROSS Q&A chatbot inside the UIC-Chat, demonstrating how the bot accesses and presents information from the ACROSS Service Catalogue

As demonstrated by the chat session above, also a basic connection between the Q&A chatbot and the ACROSS Service Catalogue (SC) has been established (which necessitated an extension to the ACROSS architecture). This connection supports conversational queries about services stored in the catalogue, such that a user might ask (before or during us of a service) for additional information about that service, and receive suitable information from the SC through the VA, i.e. conversationally as well. Thus, the functionality described in this subsection enables the VA to fulfil requirement REQ-VA-13.

(New in the final version, following discussion at ACROSS Plenary in Athens):

If the user is currently inside a journey, the VA should only respond with information that is based on the currently visited journey rather than fetching the information from the whole service catalogue. This necessitated some new collaborations through which a new connection between the VA and the Service Engine had to be made. In the final version, the VA, when asked about service specific information, fetches

data from the SE to retrieve info about what service are currently subject and then pre filters the internal data representation of the service catalogue to only match with the services currently of subject.

3.5.3 Integration between WebAssist and Q&A chatbot

Consider a user filling out a web form associated to a service he/she needs to execute within a user journey workflow, doing so by means of the VA's WebAssist functionality. Between answering the VA's prompts an urgent question occurs to the user. So, instead of answering the last VA prompt (say "Please tell me your age"), the user confronts the VA with his/her question, e.g. "How long will it take for this service to complete after I have submitted all inputs?". Now a service made accessible by ACROSS will often be a gateway to a complex and time-consuming administrative back-office processes, so it is understandable that a user might be concerned regarding its expected elapsed time. However, in this situation, the VA cannot interpret and answer the user's utterance within the framework of its WebAssist functionality. What is needed here is a seamless switch to the Q&A chatbot functionality of the VA, to which the user's question must be forwarded, and where it will be answered. Additional user questions, e.g. about details, might follow, before the user is eventually ready to return to the original form-filling task. And if/when he/she is ready for that, the VA must switch back the conversational channel so it is connected to the WebAssist subsystem yet again, and make it continue just where it was interrupted. To achieve this, a *conversational multiplexer mechanism* has been designed and tested. This mechanism enables one and the same conversational session (i.e. chat or voice channel connecting the VA to a user) to be connected, sequentially, to different conversational back-end systems, concretely, the WebAssist and Q&A chatbot subsystems of the VA. Also developed were initial techniques for automatically recognizing conversational situations in which a switch to the other subsystem is needed. This is fully integrated in the final version to enable the VA to fulfil the requirements.

3.5.4 Machine Translation – MT

The goal of the VA to provide a *multilingual* conversational interface to the ACROSS platform can only be achieved by leveraging modern machine translation technologies. Multi-lingual operation of the VA works by using English as a "Pivot language" in which all incoming user utterances are translated first. That way, all dialog management functionality need only to be implemented for the pivot language. Conversely, all *dynamic* conversational system responses (i.e. those responses that have not been prepared in advance and, as text resources, thus would have been subject to traditional internationalization) are generated in the pivot language and need to be MT-translated into the current user language. The initial planning of the project envisioned an integration of the external MT service operated by the CEF (Connecting Europe Facility) Machine Translation with ACROSS. Following a closer evaluation, this plan had to be dropped,

because the CEF AT service turned out to be not sufficiently suitable. This service is optimized for batch translation requests of (often larger) documents coming in through different channels, which are managed in a kind of queueing system. Although this scheme is presumably very useful for the original purpose of the CEF AT, it is strongly sub-optimal for an interactive application such as the ACROSS VA, because of the unsatisfactory and unpredictable response times it exhibits. To fulfil the general open-source policies of the ACROSS project, we implemented a machine translation framework called EasyNMT [21]. This mature open-source machine translation system provides high-quality translation models[7] for a large number of language pairs, including models for all of the languages needed in the ACROSS Pilots. Unfortunately, we encountered bugs during testing and deployment of the framework on our internal network, which led to heavy performance spikes when using the language models inside EasyNMT. The reason for that is mainly because of the nature of language models, which are primarily designed to run on GPUs. As a good replacement, we identified the python library CTranslate2 [22], that precompiles the models to be better processable by modern CPUs (such that GPU acceleration is not needed). This led to the conclusion to implement our own custom MT-Service that utilizes the improvements of CTranslate2 in combination with the broadly availability of the "Opus-MT" model family. This combination greatly enhances the performance and stability of the MT-Service compared to the intermediate version. Integration of the MT-Service is finished in the final version and will strongly contribute to fulfilment of REQ-VA-7.

3.5.5 Voice interfaces – ASR and TTS

Although voice communication with the VA has been a stated goal of its development from the start, there was a considerable lack of clarity in the early phases of the project as to how this goal could be practically achieved. At that time there were no free and open options for Automatic Speech Recognition (ASR) and Text-To-Speech (TTS) speech synthesis – neither in the form of open-source software nor as freely accessible online services. As regards external (ASR and TTS) services, we decided to not strongly pursue this avenue because it could easily lead to conflicts with data protection and privacy objectives which ACROSS adheres to. Especially regarding ASR, audio of spoken language can be regarded as biometric data that can be used to identify the speaker even if the content of the spoken text would not allow such identification at all. As the legal issues are complex, the safest option was to rule out scenarios in which such data would travel to third parties, so external ASR services were no longer considered.

For a while, it seemed that web browsers would soon contain good ASR and TTS implementations that could be used in the project. This would have been attractive since especially mobile OSes often contain

[7] We have selected the "Opus-MT" [32] model family as they cover all needed language pairs for the project.

powerful on-device ASR and TTS components these days. Unfortunately, although suitable browser APIs (concretely, the W3C Web Speech API) have been defined for several years now, actual browser support is patchy to this day, especially when considering which languages are covered. Therefore, fulfilling the ACROSS requirements regarding browser compatibility (concretely, Req_36 above) would not have been possible by building on in-browser ASR and TTS solutions. Finally, the ASR functionality of the WebSpeech API would likely have been too limited for the VA in any case. Thus, in-browser ASR and TTS components did not materialize as sufficient options for the requirements of the ACROSS VA.

Fortunately, since around 2020, the fields of ASR and TTS witnessed a very strong development thanks to the recent progress in AI and Machine Learning, which led to the emergence of powerful open-source solutions for these tasks. After an evaluation phase, we selected the following open-source packages for the ACROSS VA:

- ASR: Vosk [23]. This ASR package supports the majority of languages needed for the ACROSS Pilots, provides high recognition quality, and exhibits very good performance: it does not need a GPU and even works on lightweight devices such as the Raspberry Pi.
- TTS: Coqui TTS [24]. This TTS package provides a selection of the best deep-learning based TTS methods currently known. Predefined models currently do not cover the majority of languages needed for the ACROSS Pilots yet, but at least provide sufficient functionality for demos by supporting English and German. In any case, Coqui is currently the best option available and exhibits good acceptance and growth within the open-source TTS community, so there is hope that additional languages will be covered soon.

Integration of Vosk ASR and Coqui TTS is completed in the final version and will strongly contribute to fulfilment of REQ VA-7.

3.6 VA - Baseline technologies

The Virtual Assistant uses some standardized baseline technologies. Those technologies are:

WebSockets: a communication protocol, standardized by the IETF. It provides full-duplex communication channels using a single TCP connection. It is a mature, well-defined, well-known standard for communication.

Socket.IO is a quasi-standard cross-language API based on the IETF WebSocket protocol. It enables real-time, bidirectional and event-based communication between the browser and the server. It is developed in an open and distributed manner under the guidance the Open Source Collective, sponsored by almost 200 companies and organizations and routinely used by an even larger professional audience.

Document Object Model (DOM): Standard interface for manipulating HTML documents, treating the documents as tree structures with identifiable objects. It is standardized by the W3C and the WHATWG. DOM is used to programmatically access, read and modify the tree or the contents of its nodes.

Representational state transfer (REST) is an architectural style for implementing hypermedia systems in the Web. In a narrower meaning that is used here, it provides a definition (rather guidelines) of stateless web APIs. Web APIs that conform to REST guidelines are called RESTful APIs. REST is not standardized per se, but it is well defined by papers and convention.

JavaScript (also called ECMAScript) is a programming language, it is a core technology of the Web, just as HTML or CSS. It conforms to the ECMAScript standard but comes in different flavors by different vendors. Nonetheless, a core JavaScript language definition is supported in all flavors, thus making the use of the language safe for our needs.

Open source machine learning frameworks like:

- **Rasa** (3.5.2) a framework to automate text and voice-based conversation. Delivers intent and entity classification for our internal NLP-Pipeline.
    - **spaCy** a library (used inside Rasa) for advanced NLP in python. The primarily use-case for the VA are pre-trained models for named entity recognition to enhance our NLP-Pipeline.
- **Vosk** (3.5.5) a toolkit for offline speech recognition with support for all necessary languages.
- **Coqui** (3.5.5) a library for advanced speech synthesis (Text-to-Speech), built on the latest research and with support for many languages.
- **CTranslate2** (3.5.4) a library for efficient processing of transformer-based language models. This enables the use of more “heavy” machine translation models inside our MT-Service without a need for GPU hardware.

3.7 VA – Conclusions and next steps

The Final version of the ACROSS VA fully validates the principal approach of integration of the VA in ACROSS and demonstrates the potential the VA offers for extending any desired set of possible Web/Mobile App interactions with a conversational interface. It strongly goes beyond the functionality of the Initial Prototype, introducing a host of extensions. These do not merely provide quantitative improvements, e.g. regarding the coverage of conversational control of the ACROSS Web/Mobile App. Instead, they provide qualitatively new and important functionality:

- The Q&A Chatbot subsystem enables conversations beyond the narrow scope of controlling the ACROSS Web/Mobile App, and by its coupling with the ACROSS Service Catalogue, allows the user execute queries about services conversationally.
- The Voice interface subsystems ASR and TTS enable communication with the VA and thus with the ACROSS platform in natural, spoken language, and support a substantial subset of the target languages needed for the ACROSS pilot use cases.
- The Machine Translation (MT) subsystem enables the ACROSS platform to reduce language barriers by automatically translating the user utterances in a conversation (be it via chat or through the voice interfaces) from the user's native language into the VA's internal system language (also called "pivot language"), English. It is also able to translate system utterances back to the user's native language where needed.

With these recent extensions, the ACROSS VA is getting ever closer to leveraging the full potential of conversational user interfaces for ACROSS, and bringing its original vision to reality.

With the final version of the VA, apart from extending the scope of the interactions covered, and covering additional requirements arising from the use case work packages WP2 and WP6, improved integration was the overarching goal and has been made. On the one hand side, the different subsystems are more closely integrated than in the intermediate version, on the other hand, integration with the ACROSS platform and the components the VA collaborates with are strengthened and deepened. Also, some subcomponents and the way these subcomponents collaborate have been further optimized in order to achieve the best possible VA User Experience.

With recent innovations in the field of large language models (LLM) we believe that these models will play a very important role in and, more to the point, indeed have already started to revolutionize the fields of natural language understanding, natural language processing, and conversational systems like the ACROSS Virtual Assistant. The fact that Microsoft and Google are augmenting even their search engines with chat interfaces, together with the observation that web search is one of the most everyday and lowest-barrier uses for computers and smartphones today, is a powerful indication that the era of conversational interfaces is arriving. The ACROSS VA does not yet itself build on the new and very powerful large language models (LLMs) that are the basis for ChatGPT and Bard, because these models have only become available in recent months. Consequently, the VA is relatively limited in its capabilities to understand the user and to generate system replies – nevertheless, it can rightfully claim to be on the forefront of innovation regarding introduction of the *concept* of conversational user interfaces into new application domains. Even though completing this task will be left to a future project, leveraging the full power offered by the new LLMs will be much easier given the preparation and integration work that has

already been done within ACROSS. Integrating LLMs, especially as a replacement for the natural language understanding module currently used by the VA (Rasa NLU), will considerably enhance the user experience of conversational interactions. We definitely plan to further pursue this direction of work in future projects.

4 References

[1] “ACROSS Deliverable D4.7 User Support Tools - Initial”.

[2] ACROSS, “D4.8 User Support Tools – Intermediate”.

[3] ACROSS, “D5.2 System Architecture & Implementation Plan - Final”.

[4] ACROSS, “D5.4: ACROSS Platform Prototype and Applications - Intermediate”.

[5] “Final Report Summary - BPM4PEOPLE (Business Process Modelling for Participatory Enterprises, Organizations, and Public Aministration Bodies),” [Online]. Available: https://cordis.europa.eu/project/id/285929/reporting.

[6] “Productive4.0,” [Online]. Available: https://productive40.eu.

[7] J. Erasmus, I. T. P. Vanderfeesten, K. Traganos, R. Keulen and P. W. P. J. Grefen, “The HORSE Project: The Application of Business Process Management for Flexibility in Smart Manufacturing,” 2020.

[8] “The COMPOSITION project, Ecosystem for Collaborative Manufacturing Processes – Intra- and Interfactory Integration and Automation,” [Online]. Available: https://www.composition-project.eu

[9] “PASSME Website,” [Online]. Available: https://passme.eu.

[10] “PASSME project page in CORDIS,” [Online]. Available: https://cordis.europa.eu/project/id/636308

[11] “Gravitate–Health project page in CORDIS,” [Online]. Available: https://cordis.europa.eu/project/id/945334

[12] “About ISA² - European Commission,” [Online]. Available: https://ec.europa.eu/isa2/isa2_en

[13] “Core Public Service Vocabulary Application Profile - Joinup.eu,” [Online]. Available: https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/solution/e-government-core-vocabularies/core-public-service-vocabulary-application-profile.

[14] “European taxonomy for public services - Joinup.eu,” [Online]. Available: https://joinup.ec.europa.eu/sites/default/files/news/2019-09/ISA2_European%20taxonomy%20for%20public%20services.pdf.

[15] “Connecting Europe Facility,” [Online]. Available: https://ec.europa.eu/inea/en/connecting-europe-facility

[16] “CEF eTranslation service,” [Online]. Available: https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/Machine+translation

[17] “European Accessibility Act,” [Online]. Available: https://ec.europa.eu/social/main.jsp?catId=1202

[18] “Web Accessibility Directive,” [Online]. Available: https://directive2102.eu/

[19] “ACROSS D5.1 System Architecture & Implementation Plan – Initial”

[20] https://github.com/rasahq/rasa

[21] “https://github.com/UKPLab/EasyNMT,” [Online].

[22] [Online]. Available: https://github.com/OpenNMT/CTranslate2

[23] https://alphacephei.com/vosk/

[24] https://github.com/coqui-ai/TTS

[25] “Excalidraw - Github,” [Online]. Available: https://github.com/excalidraw/excalidraw.

[26] “jgraph/drawio: Source to app.diagrams.net - GitHub,” [Online]. Available: https://github.com/jgraph/drawio.

[27] “Camunda Modeler - GitHub,” [Online]. Available: https://github.com/camunda/camunda-modeler

[28] “Bonita Studio - GitHub,” [Online]. Available: https://github.com/bonitasoft/bonita-studio

[29] “BPM4PEOPLE project page in CORDIS,” [Online]. Available: https://cordis.europa.eu/project/id/285929

[30] https://github.com/argosopentech/argos-translate

[31] A. D. ". A. &. I. P. –. Initial".

[32] J. Tiedemann and S. Thottingal, “OPUS-MT — Building open translation services for the World,” in *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*, Lisbon, Portugal, 2020.