Deliverable D4.3

Final Validation Procedure

WP 4

Project Acronym & Number:	SmartCLIDE – GA 871177
Project Title:	Smart Cloud Integrated Development Environment supporting the full-stack implementation, composition and deployment of data-centred services and applications in the cloud
Status:	Final
Dissemination Level:	Public
Authors:	TOG
Contributors:	CONTACT, INTRA, UNP and WT
Document Identifier:	D4.3 Final Validation Procedure.docx
Date:	20.07.2022
Revision:	2.0
Project website address:	www.smartclide.eu

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No *871177*

Project Partners

Institut für angewandte Systemtechnik Bremen GmbH (ATB), Germany
INTRASOFT International SA (INTRA), Luxembourg
Fundacion Instituto Internacionale de Investigacion en Intelligencia Artificial y Ciencias de la Computacion (AIR), Spain
University of Macedonia (UoM), Greece
Ethniko Kentro Erevnas Kai Technologikis Anaptyxis (CERTH), Greece
The Open Group Ltd (TOG), United Kingdom
Eclipse Foundation Europe GmbH (ECLIPSE), Germany
Wellness Telecom SL (WT), Spain
Unparallel Innovation LDA (UNP), Portugal
CONTACT Software GmbH (CONTACT), Germany
Kairos Digital, Analytics and Big Data Solutions SL (KAIROS DS), Spain

Dissemination Level

PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Document Control

Version	Notes	Date
0.2	First outline and inputs from industrial pilot partners	25.05.2021
0.4	Introduction and methodology sections	12.06.2021
0.5	Initial validation tests for Pilot Cases and KPIs	29.06.2021
0.6	Updated validation tests from UNP, Intra and Contact	29.07.2021
0.7	Updated KPIs/focus areas from UNP, Intra and Contact	01.08.2021
0.8	Inclusion of validation tests from WT	03.09.2021
0.9	Inclusion of inputs from WT for KPIs/focus areas	22.09.2021
1.0	Final reviews and QA version for EC submission	27.09.2021
1.4	Initial draft of updated sections and revisions for full prototypes	01.06.2022
1.5	Additional test runs for final validations by industrial pilot partners	13.06.2022
1.8	Revisions of test runs to align with full prototype functionality	20.06.2022
1.9	Further revisions of test runs and streamlining of test procedures	15.07.2022
2.0	Final reviews and QA version for EC submission	20.07.2022

Abbreviations

D	Deliverable
e.g.	exempli gratia in Latin meaning: for example
EC	European Commission
etc.	et cetera in Latin meaning: and so forth
EU	European Union
i.e.	id est in Latin meaning: that is to say
IDE	Integrated Development Environment
IoT	Internet of Things
KPI	Key Performance Indicator
M	Project month
QA	Quality Assurance
QoE	Quality of Experience
QoS	Quality of Service
T	Task
VM	Virtual Machine
V&V	Verification & Validation
w.r.t.	with respect to
WP	Workpackage
WPn	Workpackage number *n*

Executive Summary

This deliverable describes the validation procedures that will be applied for the full prototype of the SmartCLIDE solution. This deliverable provides a description of the validation environments that will be used by each of the industrial Pilot Case partners, along with the individual Test Runs that will be executed to determine the degree of achievement in fulfilling the industrial needs for a Cloud-based IDE. Also included are the procedures for Pilot Case partners to report bugs and suggested revisions to the research and development teams for inclusion in the final prototype of the solution delivered at the end of the project. A final section addresses the overall performance indicators and targets that will form the basis for the Assessment Methodology and associated Assessment Scenarios under Workpackage 5, which complement the full prototype validation testing by quantifying the business, operational and other impacts delivered by the project technologies for each of the industrial Pilot Case partners.

Table of Contents

List of Figures

1 Introduction

1.1 Overview

Software validation is part of the software engineering tasks of Verification and Validation (V&V), which is a disciplined approach to determine that software systems meet their specifications and fulfil their intended purpose. The starting point for V&V in SmartCLIDE are the requirements identified early in the project (see D1.2), which specifies and prioritises the capabilities and features needed by the industrial Pilot Case partners. These have been further elaborated with the specification of the Use Case Scenarios that represent the types of software development and deployment tasks each Pilot Case partner expects to be able to carry out through the introduction of a Cloud-based IDE.

This Final Validation Procedure deliverable is designed to prescribe the scope, approach, and resources of the testing activities to be carried out to validate the full prototype of the SmartCLIDE IDE. The plan identifies the items to be tested from an industrial application viewpoint, the features to be tested, and the types of testing to be performed by the Pilot Case partner teams in their role as end users of the project technologies.

The output of these validation procedures for the full prototype technologies is the confirmation from the end users' perspective that the features and capabilities for Cloud-based development meets the identified user's needs, while also potentially highlighting any issues or implementation decisions that might be improved to maximise meeting the end users' expectations in the final prototype delivery at the end of the project.

1.2 Validation standards and practices

Software Engineering standards addressing V&V of software systems have been established and widely adopted by industry for more than 30 years with the first IEEE 1012 standard of definitions and practices published in the late 1980s. The more recent and relevant standards that have been used as the basis for the project validation procedures are the following:

- **IEEE 1012-2016** – System, Software, and Hardware Verification, and Validation, is the latest revision of IEEE 1012 standard published in 2017, which defines the V&V processes in terms of specific activities and related tasks. This standard considers that V&V may be performed at the system, subsystem, software element, or hardware element level, or on any combination. Early versions of this standard focused on V&V for software, while later versions also included hardware.
- **ISO/IEC/IEEE 12207:2017** – establishing standard definitions of the software lifecycles and life cycle processes. It contains processes, activities, and tasks to be applied during the software product or service development, operation, maintenance and disposal.

The SmartCLIDE project workpackages and tasks were also prepared in compliance with the above Software Engineering standards and definitions.

1.3 Definition of Validation

In accordance with the above referenced standards, the SmartCLIDE validation tasks and plans contained in this deliverable use the following definition:

Validation is assuring that a software system meets the user's needs

A clear understanding of the term is important, otherwise assumptions concerning what tasks are involved and responsibilities can lead to some confusion regarding the scope and purpose of validation testing. Validation testing occurs when there are concrete features and capabilities that can be evaluated by the end users of the system. It is a key process to ensure the project what is being or has been developed meets the expectations of the users.

Verification testing is also an important and complimentary process within the project, which focuses on addressing whether the SmartCLIDE software was correctly developed. Verification testing in SmartCLIDE is carried out by the research and development teams responsible for each component and the integrated SmartCLIDE IDE. Validation testing in SmartCLIDE in accordance with ISO and IEEE standards is carried out by the industrial Pilot Case partners to assure the system meets their needs for carrying out Cloud-based software design, development, testing, deployment and maintenance tasks. A common reminder to distinguish between the two elements of V&V is:

- **Verification:** "Are we building the product right?" – e.g. are there defects and bugs in the code?
- **Validation:** "Are we building the right product?" – e.g. is the software usable and meets the needs of the user?

This deliverable describes the validation testing that is carried out by the Pilot Case partners as end users of the full SmartCLIDE prototype.

1.4 Relationship to other deliverables

This deliverable focuses on the validation plans for the full prototype of the SmartCLIDE IDE by the industrial Pilot Case partners in the final months of the project. The validation tasks are based on the specification of the Pilot Case systems and the associated requirements prioritised from the perspective of each of the industrial domains targeted by the project, as provided in deliverable *D1.2 – Requirements Analysis*, and the representative use cases and scenarios specified by each of the Pilot Case partners in deliverable *D1.3 – Use Case Scenarios*. The use cases serve as a detailed specification of the SmartCLIDE users' future needs and are intended to describe what the SmartCLIDE technologies are envisioned to deliver in terms of new functionalities and capabilities for Cloud-based software development. As such, the use cases formed the basis for the validation testing to be carried for both the early prototype and full prototype of the SmartCLIDE IDE.

As an update to the earlier validation planning deliverable D4.1 scheduled at M19 addressing a subset of the planned capabilities, this deliverable addresses the validation of the entirety of the high priority industrial requirements established early in the project by the Pilot Case partners. The same methods described in the earlier report are again applied, but the validation testing is substantially more extensive as all of the planned features of the SmartCLIDE IDE will have been fully implemented. A further deliverable *D5.1 – Assessment Methodology*, will specify the quantitative measures that will be applied to assess the business, operational and other improvements provided by the final project technologies across a set of performance indicators and targets collected from the Pilot Cases. This deliverable will be complemented by *D5.2 Specification of pilot scenarios and assessment planning*, which describes the scenarios Pilot Case partners will utilise to evaluate the performance indicators.

1.5 Structure of this document

This deliverable is structured as follows:

- Section 2 – describes the validation environments and validation test runs of the full SmartCLIDE prototype to be carried out by each Pilot Case partner.
- Section 3 – presents the bug tracking procedures to be followed by the Pilot Case partner validation teams to capture issues to be addressed during the final prototype developments to be completed by the end of the project.
- Section 4 – summarises the KPIs, targets, assessment methods, and focus areas for each of the Pilot Cases for the eventual quantitative impact assessment of the full prototype. These are provided for consideration during validation testing to enable detection of potential shortcomings or limitations that might prevent impact targets from being achieved by the full prototype.

Concluding remarks are provided and sources for additional information are footnoted throughout the document.

1.6 Contributors

The Pilot Case partners CONTACT, INTRA, UNP and WT have been the main contributors to this deliverable identifying the environments and test runs that will be utilised for validating the full prototype of the SmartCLIDE IDE. TOG has contributed the methodologies and procedures, and has acted as overall editor for this deliverable.

2 Validation Test Cases

The validation test cases for both the early and full prototype are designed based on the Functional and Technical Requirements (D1.2), and the Use Case Scenarios (D1.3). As the full prototype of the SmartCLIDE solution (D4.4) provides a complete of the functionalities targeted by the project, all of the Pilot Case Scenarios are represented for validation purposes. The validation testing focuses on providing industrial feedback and guidance to the final prototype (D4.5) at the end of the project based on the test cases applied to the full prototype functionalities.

The collection use cases that are expected to be operational form the validation Test Runs that will be implemented by each Pilot Case partner in collaboration with the R&D partners. The test runs and functionalities to be validated are summarised in Table 1.

Table 1: SmartCLIDE capabilities focus for Pilot Case validations

Pilot Case	Full Prototype Validation Test Runs
Intrasoft	Design and development functionalities
	Test optimisation and deployment
	Improve code quality of service
	Assess performance of deployed services
Wellness Telecom	Customized deployment
	Service management
	QoS Monitoring
	Scaling of the Communication Platform
Unparallel Innovation	Creation of services
	Classification of services
	Estimate deployment costs
CONTACT Software	Collaborative development
	Performance analysis
	Improve code quality of service

It should be noted that the Test Runs specified by each Pilot Case partner are based on current understanding of features planned for inclusion in the full prototype of the SmartCLIDE IDE. It is possible some individual tests within a test run may need to be adapted for use with the full prototype depending on the maturity or small variances in the specifics of the implementation of the feature.

The following describes the detailed Test Runs that will be used by each Pilot Case partner for the validation testing of the full prototype of the SmartCLIDE solution.

2.1 Intrasoft International Pilot Case validation

A frequent problem in designing and developing software products at INTRASOFT among separate and geographically distributed developer teams is to ease the design and code development process, optimize collaboration among the team members, and to optimise the design, development and testing time. In this scenario, which is an extensive one, using SmartCLIDE capabilities for code/service re-use, service discovery, language support, and service discovery from external sources, along with debugging and cost analysis reports.

2.1.1 Validation Environment

A subset of specific Pilot Cases and Use Case Scenarios have been carefully identified and selected for the role of carrying out the testing and validation of the core SmartCLIDE environment and its capabilities.

The whole set of use case scenarios embraces a flow from the need to develop and design a new functionality to testing, deployment, and support. This new functionality or this service can also begin with a template or use a discovery service to fetch it. Next, we have the optimization, deployment, and testing cases that will be performed on top of the service. Last follows the use cases that involve reporting tools, security, and repository access.

More specifically the selected Use Cases and scenarios will cover the testing of the following SmartCLIDE services and its components, which will be included in the validation environment

- Service Creation, Composition and Testing component.
- Discovery of services component
- Security component
- Run-time Simulation & Monitoring / Visualisation component
- User interface component
- Service deployment component

The common environment provided for the full prototype of SmartCLIDE is deemed suitable for the purposes of these validation tests, since there is no requirement to integrate with PERSEUS-specific components to validate the prototype. The following summarises the Test Runs that will be executed for validating the full prototype of the SmartCLIDE solution.

2.1.2 Test Run 1: Design and development functionalities

Test Name		**IS-0001 Creation of a service from a template**
Actors		Developer (SmartCLIDE User)
Triggers		A developer wants to create a new service for an existing system or a completely new one.
Preconditions		None. It should always be possible to create a new service from a template. If service templates are provided as an additional plugin, it has to be installed.
Normal Flow	Description	1. The developer has a service for a certain purpose in mind. 2. The developer navigates to the template repository 3. The developer chooses an adequate template of a sortable list (e.g. by filtering meta data)
	Postconditions	The system provides the developer with a code skeleton of the chosen service (in a new or existing tag/ file)

Test Name		**IS-0002 Create services with data abstraction levels**
Actors		Developer (SmartCLIDE User)
Triggers		A developer wants to implement a set of services (e.g. a database system) and abstract data from said services.
Preconditions		None. It should always be possible to create new services.
Normal Flow	Description	1. The developer composes a set of services. 2. The developer defines the abstraction of data (e.g. data flows, data formats) from the implemented services with the help of the system (e.g. tool tips, auto-completions)
	Postconditions	A set of services with the desired data abstraction levels (data flows, input and output formats) has been created.
Alternative Flows and Exceptions		1. The services already exist and may even be in use. 2. SmartCLIDE provides the developer with help to create data abstraction for the services.

Test Name		**IS-0003 Create and deploy a service from the IDE**
Actors		Developer (SmartCLIDE User)
Triggers		A developer wants to create a new service for an existing system or a completely new system one using command line tools.
Preconditions		Necessary packages installed on the host in order to provide SmartCLIDE with the ability to use them. In addition to that, all necessary tools for a local Kubernetes instance are installed.
Normal Flow	Description	1. The developer uses SmartCLIDE to implement a new service. 2. The developer uses the built-in functionality to verify the written code with an installed linter(e.g. using Pylint for Python code) 3. The developer uses the built-in functionality to deploy the new service (whilst providing SmartCLIDE with a valid configuration) with kubectl in a local Kubernetes environment.
	Postconditions	The service has been successfully deployed.
Alternative Flows ant Exceptions		1. The developer uses SmartCLIDE to implement a new service. 2. The developer uses the built-in functionality to verify the written code (e.g. using Pylint for Python code) 3. The developer uses the built-in functionality to deploy the new service with kubectl using a saved configuration within SmartCLIDE

Test Name		**IS-0004 Discover resources and services**
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		A SmartCLIDE user wants to discover existing services of a system.
Preconditions		A system that consists of one or more services is deployed/ in use or just known to SmartCLIDE (via the registry).
Normal Flow	Description	1. The user starts SmartCLIDE and navigates to its search engine 2. The user inputs necessary data into the search engine (e.g. credentials, type of service/ resource) and specifies that he/ she wants to search through the REST API 3. The user uses SmartCLIDE to search for services and resources
	Postconditions	SmartCLIDE provides the user with an overview of services and resources within the searched system
Alternative Flows and Exceptions		1. The user starts SmartCLIDE and navigates to its search engine 2. The user inputs necessary data (e.g. credentials, type of service/ resource) into the search engine and specifies a URL (or a rancher namespace) to be searched. 3. The user uses Smart CLIDE to search for services and resources. 1. The user starts SmartCLIDE and navigates to its search engine 2. The user inputs necessary data into the search engine (e.g. credentials) and specifies that he/ she wants to search through the REST API 3. The user uses SmartCLIDE to search for services and resources 4. After the initial search, the user switches to a configuration of another environment (e.g. another Kubernetes cluster)

Test Name		**IS-0005 Search for deployed services**
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		The user wants to look up already deployed services.
Preconditions		Several services have been implemented and deployed via SmartCLIDE.
Normal Flow	Description	1. The user navigates to the SmartCLIDE registry (e.g. via a tool bar) 2. The user filters the registry in order to specify the search

Test Name		**IS-0005 Search for deployed services**
		results
	Postconditions	SmartCLIDE provides the user with a list of deployed services based on the inputs/ filters given by the user.
Alternative Flows and Exceptions		1. The user uses the built-in command line tool to look up deployed services. 2. The user specifies additional arguments to the command line call to filter the search results.

Test Name		**IS-0006 A non-expert user creates a new service with assistance**
Actors		Non-trained user (SmartCLIDE user)
Triggers		A non-trained user wants to create a new service or add a service to an existing system.
Preconditions		None.
Normal Flow	Description	1. The user has a service for a certain purpose in mind. 2. The user navigates to the template repository. 3. The user chooses an adequate template of a sortable list (e.g. by filtering meta data). SmartCLIDE provides short documentation to each template (description, purpose etc.) 4. The user adds the necessary specifics to the template code, SmartCLIDE provides tips and possible solutions via the auto-complete feature.
	Postconditions	A ready to deploy service is created from the process.
Alternative Flows and Exceptions		1. The user navigates to the SmartCLIDE registry of deployed services. 2. The user selects the system to which a service should be added. 3. SmartCLIDE provides the user with an overview of the system and its dependencies (maybe even some sort of documentation). 4. The user navigates to the template repository. 5. The user chooses an adequate template of a sortable list (e.g. by filtering meta data). SmartCLIDE provides short documentation to each template (description, purpose etc.) 6. The user adds the necessary specifics to the template code, SmartCLIDE provides tips and possible solutions via the auto-complete feature. 7. The user deploys the created service to the existing system. 8. **Additional postcondition**: The deployed service finds its way into SmartCLIDE's service registry.

Test Name		IS-0008 Test services from within SmartCLIDE
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		The user wants to test already existing services.
Preconditions		One or more services and corresponding tests already exist. The needed command line tool to execute the tests is installed on the host.
Normal Flow	Description	1. The user uses the built-in command line of SmartCLIDE to execute the written tests. 2. The command line prints the results.
	Postconditions	SmartCLIDE marks the service(s) as tested or adds the test coverage to the meta data of the service.

Test Name		IS-0015 Accessing Git repositories
Actors		Developer (SmartCLIDE User)
Triggers		The developer wants to push changes made to an existing service or a new file to a Git repository.
Preconditions		The necessary tools (e.g. Git bash) are installed on the host, the source code has been cloned.
Normal Flow	Description	1. The user makes changes to an existing service. 2. The user saves the file. 3. The user commits the changes via the built-in CLI. 4. The user pushes the commits to a remote repository via the built-in CLI.
	Postconditions	The commits have been successfully pushed to the defined repository.
Alternative Flows and Exceptions		1. The user makes changes to an existing service. 2. The user saves the file. 3. The user uses the context menu to commit the changes. 4. A window pops up, in which the user enters the commit message. 5. The user uses the context menu to push the commits to a remote repository. 6. A window pops up if the repository is private and no credentials are available.

Test Name	IS-0018 Language support
Actors	Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User), IT Administrator (SmartCLIDE User)
Triggers	A user wants to create a new service or change an existing one.

Test Name		**IS-0018 Language support**
Preconditions		If needed additional language support packages need to be installed.
Normal Flow	Description	1. The user creates a new file. 2. The user implements the desired solution, while SmartCLIDE provides the syntax highlighting.
	Postconditions	SmartCLIDE shows support for the selected language by showing correct highlighting of syntax.
Alternative Flows and Exceptions		1. The user opens an existing file. 2. The syntax of the code is automatically correctly highlighted. 3. The user implements the changes.

Test Name		**IS-0019 Access SmartCLIDE via a browser**
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User), IT Administrator (SmartCLIDE User)
Triggers		A user wants to access SmartCLIDE from within a browser, since it might not be installed on the local machine.
Preconditions		A SmartCLIDE container or virtual machine has to be deployed somewhere reachable (local docker container, remote access to virtual machine etc.)
Normal Flow	Description	1. The user opens a browser. 2. The user navigates to the correct url and opens SmartCLIDE
	Postconditions	All of SmartCLIDE’s functionality is available via the web interface.

Test Name		**IS-0020 Specify the licence of a service**
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		A user wants to specify the license of a service without knowing, which one would be appropriate.
Preconditions		An implemented service exists without any license specified.
Normal Flow	Description	1. The user opens the source code of the service (one file). 2. The user navigates to the license mechanism of SmartCLIDE. 3. The user filters the possible licenses by providing information regarding the service (purpose, commercial or not) 4. The user chooses a license based on the suggestion of SmartCLIDE

Test Name		IS-0020 Specify the licence of a service
	Postconditions	The license is inserted at the top of the service's source code.

Test Name		IS-0027 Measuring and debugging SmartCLIDE
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		A user wants to debug SmartCLIDE or look at usage metrics.
Preconditions		The logging has to be activated and set to the necessary level.
Normal Flow	Description	1. The user takes a look at the recent logs SmartCLIDE has written. The debug level may be changed at runtime. 2. The user opens SmartCLIDE and navigates to the usage data section.
	Postconditions	SmartCLIDE provides the user with a set of metrics gathered from normal usage.

2.1.3 Test Run 2: Test optimisation and deployment

Related Tests: *IS-0005 Search for deployed services* and *IS-0008 Test services from within SmartCLIDE*

Test Name		IS-0010 Deploy a service from the CLI within SmartCLIDE
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		A user wants to deploy an already implemented service via SmartCLIDE. The necessary tools (e.g. Kubectl, AWS, Helm, Docker) are installed.
Preconditions		A service ready to be deployed exists.
Normal Flow	Description	1. The user navigates to the built-in command line interface. 2. The user provides the system with a valid configuration or uses a saved one within SmartCLIDE. 3. The user deploys the implemented service via the CLI.
	Postconditions	The service has been deployed accordingly.

Test Name	IS-0017 Ask for changes in SmartCLIDE
Actors	Quality Manager (SmartCLIDE User), IT Administrator (SmartCLIDE User)
Triggers	A user finds a bug or wants to demand a new feature.

Test Name		IS-0017 Ask for changes in SmartCLIDE
Preconditions		None.
Normal Flow	Description	1. The user goes online and visits the public repository of SmartCLIDE 2. The user navigates to the issue tracker 3. The user creates a new issue and states what is faulty or what change is wanted.
	Postconditions	A new issue is created and the maintainers of the repository are notified.

Test Name		IS-0024 Use SmartCLIDE's container repository
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		A user wants to upload a Docker image and share it.
Preconditions		A Docker image has to exist.
Normal Flow	Description	1. The user opens SmartCLIDE and navigates to the built-in container registry. 2. The user uploads a local docker image to the registry.
	Postconditions	The Docker image has been uploaded and is available for other users.
Alternative Flows and Exceptions		1. In addition to the normal flow: Another user pulls the recently uploaded image locally.

Test Name		IS-0026 Create secure services with authentication
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		A user wants to create a secure service or a set of secure services.
Preconditions		-
Normal Flow	Description	1. The user implements a service. 2. The user adds configuration to said service including an authentication method with the assistance of SmartCLIDE.
	Postconditions	The service is ready to use, a potential user has to provide valid credentials to use its functionality.
Alternative Flows and Exceptions		1. The user implements a set of services with several security measures such as authentication, https communication etc. with the assistance of SmartCLIDE. 2. The user deploys the services. 3. The services communicate with each other using secure protocols and provided credentials are always passed encrypted.

2.1.4 Test Run 3: Improve code quality of a service

Related Tests: *IS-0008 Test services from within SmartCLIDE* and *IS-0015 Accessing Git repositories*

Test Name		**IS-0018 Language support**
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User), IT Administrator (SmartCLIDE User)
Triggers		A user wants to create a new service or change an existing one.
Preconditions		If needed additional language support packages need to be installed.
Normal Flow	Description	1. The user creates a new file. 2. The user implements the desired solution, while SmartCLIDE provides the syntax highlighting.
	Postconditions	-
Alternative Flows and Exceptions		1. The user opens an existing file. 2. The syntax of the code is automatically highlighted correctly. 3. The user implements the changes.

Test Name		**IS-0027 Measuring and debugging SmartCLIDE**
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		A user wants to debug SmartCLIDE or look at usage metrics.
Preconditions		The logging has to be activated and set to the necessary level.
Normal Flow	Description	1. The user takes a look at the recent logs SmartCLIDE has written. The debug level may be changed at runtime. 2. The user opens SmartCLIDE and navigates to the usage data section. 3. SmartCLIDE provides the user with a set of metrics gathered from normal usage.
	Postconditions	-

2.1.5 Test Run 4: Assess performance of deployed service(s)

Related Tests: *IS-0004 Discover resources and services* and *IS-0005 Search for deployed services*

Test Name	**IS-0009 Conduct a cost analysis**
Actors	Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User), IT Administrator (SmartCLIDE User)

Test Name		IS-0009 Conduct a cost analysis
Triggers		The user wants to have an estimation of operating costs if the system under inspection would be deployed as a productive system.
Preconditions		An existing system consisting of one or more services that is ready for deployment.
Normal Flow	Description	1. The user uses the service registry of SmartCLIDE to filter the services needed for the system to be deployed and functional. 2. The user selects the target environment (such as cloud provider). 3. The user selects all of the needed services or the system as a whole. 4. The user checks whether the meta data of the system/ services is sufficient to conduct a cost analysis (maybe warnings or tool tips from SmartCLIDE to assist with that) 5. The user conducts the actual cost analysis.
	Postconditions	SmartCLIDE provides the user with an overview of costs the system would cause as a whole and also a breakdown of the costs for each service individually. Listed costs consist of items regarding RAM, storage, CPU, usage frequency etc.
Alternative Flows and Exceptions		1. A workflow for the desired functionality is already in place. 2. The user triggers a cost analysis via SmartCLIDE.

Test Name		IS-0025 Monitor and verify services
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		A user wants to monitor deployed services.
Preconditions		A deployment of a set of services exists.
Normal Flow	Description	1. The user opens SmartCLIDE and navigates to the deployed system. 2. SmartCLIDE provides the user with live information of the deployed services. 3. The user selects the information to be monitored.
	Postconditions	The desired information are gathered and displayed to the user.
Alternative Flows and Exceptions		1. In addition to the normal flow: The user defines thresholds for different metrics of the system (CPU workload, HDD capacity etc.). 2. SmartCLIDE notifies the user if thresholds are broken or failures occur.

2.2 Wellness Telecom Pilot Case validation

WTs use of SmartCLIDE focused on a Real-Time Communication Platform where the new project technologies are expected to be used to discover and deploy the Real-Time Communication Platform. In particular, the evaluations of the full prototype will include both the management and monitoring of a running instance of the Real-Time Communication Platform. The Real-Time Communication Platform is composed by several containerized instances that may be replicated for scalability purposes. The motivation for WT is to jointly develop tools so that the user can intuitively specify the needs of the application, deploy the applications, and adapt the infrastructure to changing needs automatically or under the supervision of WT personnel.

2.2.1 Validation Environment

The validation environment will make use of the main components associated with the Real-Time Communication Platform. In particular, these are:

- Application and infrastructure programming and control model
- Application development tools – IDE: Interactive Development Environment.
- Tools for application deployment – DRIP: Distributed Real-time Infrastructure Planner.
- Tools to enable automatic adaptability – ASAP: Autonomous Self-Adaptation Platform.

Figure 1 illustrates the relations between SmartCLIDE and WTs Pilot Case that focuses on the Real-Time Communication Platform. The validation testing of SmartCLIDE prototype will be used to carry out the following tasks:

- Discover and deploy the Real-Time Communication Platform
- Manage a running instance of the Real-Time Communication Platform
- Monitor a running instance of the Real-Time Communication Platform

The Real-Time Communication Platform is composed by several containerized instances that may be replicated for scalability purposes.

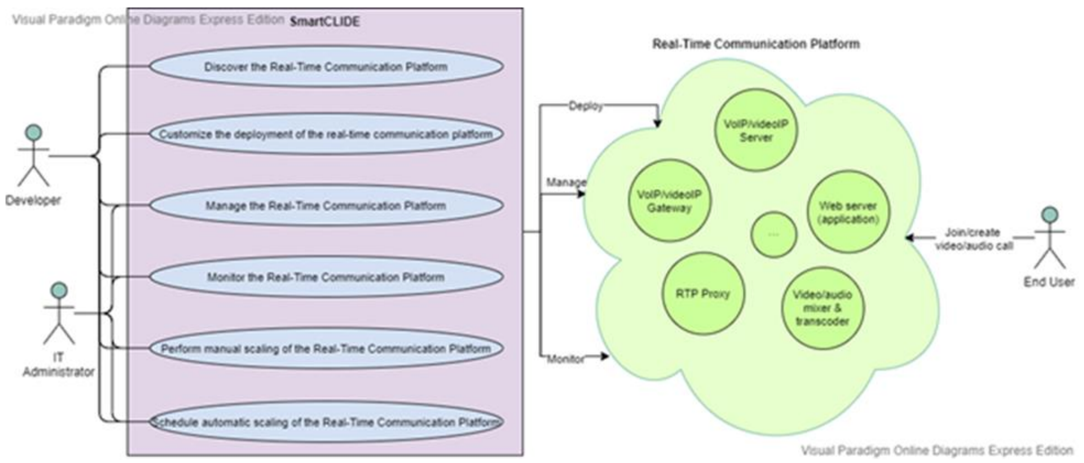

Visual Paradigm Online Diagrams Express Edition
SmartCLIDE
Discover the Real-Time Communication Platform
Customize the deployment of the real-time communication platform
Manage the Real-Time Communication Platform
Monitor the Real-Time Communication Platform
Perform manual scaling of the Real-Time Communication Platform
Schedule automatic scaling of the Real-Time Communication Platform
Developer
IT Administrator
Deploy
Manage
Monitor
Real-Time Communication Platform
VoIP/videoIP Server
VoIP/videoIP Gateway
Web server (application)
...
RTP Proxy
Video/audio mixer & transcoder
Join/create video/audio call
End User
Visual Paradigm Online Diagrams Express Edition

Figure 1: Relations between SmartCLIDE and Pilot Case

The WT system design makes it possible to easily integrate with the SmartCLIDE environment. SmartCLIDE interacts with the Unified Communication system adapting the reliability of the system given a QoS and QoE. The adaptability will depend on the demand of the functionalities used by the user. The dynamic condition will define the scalability of the system according to the metrics monitored by the system. The vertical scalability is assumed by Docker and the horizontal scalability is defined by the SmartCLIDE environment.

The SmartCLIDE system will interact with Pilot Case system is the following way:

IDE:

- Implement the WT infrastructure.
- Define the parameter QoS and QoE.
- Parameter for scaling the RTP proxy.

DRIP:

- Deploy the infrastructure according to the definition of the use of case.
- Deploy the containers and monitors defined by the IDE.

ASAP:

- Measure the general parameters: CPU, RAM, Tx Packet, etc.
- And the specific WT use case parameter: RTP Proxy ports.

The metrics collected by ASAP are used by DRIP and ASAP for scaling the containers. The general metrics and criteria are used by the system applicable to all pilot use cases, where the requirements are more focused in technical specification of the system, for instance: containers requirement, network requirement or pipeline requirement among other.

The following summarises the Test Runs that will be executed for validating the full prototype of the SmartCLIDE solution.

2.2.2 Test Run 1: Customised deployment

Test Name		**WT-0003 Create and deploy a service from the IDE**
Actors		Developer (SmartCLIDE User)
Triggers		A developer wants to create a new service for an existing system or a completely new system one using command line tools.
Preconditions		Necessary packages installed on the host to provide SmartCLIDE with the ability to use them. In addition to that, all necessary tools for a local Kubernetes instance are installed.
Normal Flow	Description	1. The developer uses SmartCLIDE to implement a new service. 2. The developer uses the built-in functionality to verify the written code with an installed linter(e.g. using Pylint for Python code) 3. The developer uses the built-in functionality to deploy the new service (whilst providing SmartCLIDE with a valid configuration) with kubectl in a local Kubernetes environment.
	Postconditions	The service has been successfully deployed.
Alternative Flows and Exceptions		1. The developer uses SmartCLIDE to implement a new service. 2. The developer uses the built-in functionality to verify the written code (e.g. using Pylint for Python code) 3. The developer uses the built-in functionality to deploy the new service with kubectl using a saved configuration within SmartCLIDE

Test Name		**WT-0010 Deploy a service from the CLI within SmartCLIDE**
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		A user wants to deploy an already implemented service via SmartCLIDE. The necessary tools (e.g. Kubectl, AWS, Helm, Docker) are installed.
Preconditions		A service ready to be deployed exists.
Normal Flow	Description	1. The user navigates to the built-in command line interface. 2. The user provides the system with a valid configuration or uses a saved one within SmartCLIDE 3. The user deploys the implemented service via the CLI
	Postconditions	The service has been deployed accordingly.

Test Name	**WT-0011 Create a system using low-code programming**
Actors	Developer (SmartCLIDE User), Quality Manager (SmartCLIDE

Test Name		**WT-0011 Create a system using low-code programming**
		User)
Triggers		A user with limited programming knowledge wants to create a service, which is deployable.
Preconditions		-
Normal Flow	Description	1. The user navigates to the modelling canvas of SmartCLIDE. 2. The user 'draws' a service using premade functional and decision blocks. 3. The user is able to change the configuration of the service via a graphical interface. 4. SmartCLIDE supports the user with tool tips and documentation at runtime (to determine whether the service would be functional). 5. The user provides the system with a valid configuration or uses a saved one within SmartCLIDE.
	Postconditions	The system has generated the needed source code and the service is ready for deployment.

Test Name		**WT-0012 Create a complex scenario from templates**
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		A user with limited programming knowledge wants to create a more complex system, consisting of multiple services, which is deployable.
Preconditions		-
Normal Flow	Description	1. The user navigates to the modelling canvas of SmartCLIDE. 2. The user chooses a system template (consisting of one or more services). 3. The system is graphically displayed on the canvas 4. The user models the system on the canvas according to its needs. 5. SmartCLIDE supports the user with tool tips and documentation at runtime (to determine whether the system would be functional).
	Postconditions	SmartCLIDE has generated the needed source code and the system is ready for deployment.

2.2.3 Test Run 2: Service management

Test Name	**WT-0004 Discover resources and services**
Actors	Developer (SmartCLIDE User), Quality Manager (SmartCLIDE

Test Name		**WT-0004 Discover resources and services**
		User)
Triggers		A SmartCLIDE user wants to discover existing services of a system.
Preconditions		A system that consists of one or more services is deployed/ in use or just known to SmartCLIDE (via the registry).
Normal Flow	Description	1. The user starts SmartCLIDE and navigates to its search engine 2. The user inputs necessary data into the search engine (e.g. credentials, type of service/ resource) and specifies that he/ she wants to search through the REST API 3. The user uses SmartCLIDE to search for services and resources
	Postconditions	SmartCLIDE provides the user with an overview of services and resources within the searched system
Alternative Flows and Exceptions		1. The user starts SmartCLIDE and navigates to its search engine 2. The user inputs necessary data (e.g. credentials, type of service/ resource) into the search engine and specifies a URL (or a rancher namespace) to be searched. 3. The user uses Smart CLIDE to search for services and resources. 4. The user starts SmartCLIDE and navigates to its search engine 5. The user inputs necessary data into the search engine (e.g. credentials) and specifies that he/ she wants to search through the REST API 6. The user uses SmartCLIDE to search for services and resources 7. After the initial search, the user switches to a configuration of another environment (e.g. another Kubernetes cluster)

Test Name		**WT-0005 Search for deployed services**
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		The user wants to look up already deployed services.
Preconditions		Several services have been implemented and deployed via SmartCLIDE.
Normal Flow	Description	1. The user navigates to the SmartCLIDE registry (e.g. via a tool bar) 2. The user filters the registry in order to specify the search results
	Postconditions	SmartCLIDE provides the user with a list of deployed services based on the inputs/ filters given by the user.

Test Name	WT-0005 Search for deployed services
Alternative Flows and Exceptions	1. The user uses the built-in command line tool to look up deployed services. 2. The user specifies additional arguments to the command line call to filter the search results.

2.2.4 Test Run 3: QoS monitoring

Test Name		WT-0021 Define and configure QoS
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		A user wants to define or evaluate QoS configurations regarding resources.
Preconditions		At least one ready implemented service exists.
Normal Flow	Description	1. The user defines the configuration of a service with the use of functional blocks provided by SmartCLIDE. 2. The user specifies additional dependencies and QoS constraints to other services.
	Postconditions	The added configuration is shown within the meta data of the services and is shown on the modelling canvas.
Alternative Flows and Exceptions		1. Given a set of services with dependencies and SLAs exists. 2. The user opens the modelling canvas and lets SmartCLIDE evaluate the given system regarding its resource requirements. 3. SmartCLIDE provides the user with information on what resources are needed to fulfil the requirements and maintain the desired QoS.

Test Name		WT-0023 Analyse a system and clean its data
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		A user wants to clean the data of a system.
Preconditions		The system is already deployed and SmartCLIDE has been analysing and classifying it with all its resources.
Normal Flow	Description	1. The user opens SmartCLIDE and navigates to the overview of the existing system. 2. SmartCLIDE shows a classification of the resources and services in place. 3. The user uses the SmartCLIDE functionality to clean the data from the system’s experience.
	Postconditions	The data sets are cleaned and may be used for training a predictive

Test Name		WT-0023 Analyse a system and clean its data
		data model.

Test Name		WT-0025 Monitor and verify services
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		A user wants to monitor deployed services.
Preconditions		A deployment of a set of services exists.
Normal Flow	Description	1. The user opens SmartCLIDE and navigates to the deployed system. 2. SmartCLIDE provides the user with live information of the deployed services. 3. The user selects the information to be monitored.
	Postconditions	The desired information are gathered and displayed to the user.
Alternative Flows and Exceptions		1. In addition to the normal flow: The user defines thresholds for different metrics of the system (CPU workload, HDD capacity etc.). 2. SmartCLIDE notifies the user if thresholds are broken or failures occur.

Test Name		WT-0027 Measuring and debugging
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		A user wants to debug SmartCLIDE or look at usage metrics.
Preconditions		The logging has to be activated and set to the necessary level.
Normal Flow	Description	1. The user takes a look at the recent logs SmartCLIDE has written. The debug level may be changed at runtime. 2. The user opens SmartCLIDE and navigates to the usage data section. 3. SmartCLIDE provides the user with a set of metrics gathered from normal usage. 4. The user opens SmartCLIDE and navigates to the deployed system.
	Postconditions	The desired information are gathered and displayed to the user.

2.2.5 Test Run 4: Scaling of the Communication Platform

Related Tests: *WT-0005 Search for deployed services* and *WT-0012 Create a complex scenario from templates*

Test Name		WT-0004 Discover resources and services
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		A user wants to discover existing services of a system.
Preconditions		A system that consists of one or more services is deployed / in use or just known to SmartCLIDE (via the registry).
Normal Flow	Description	1. The user starts SmartCLIDE and navigates to its search engine. 2. The user inputs necessary data into the search engine (e.g. credentials, type of service/ resource) and specifies that he/ she wants to search through the REST API. 3. The user uses SmartCLIDE to search for services and resources.
	Postconditions	SmartCLIDE provides the user with an overview of services and resources within the searched system
Alternative Flows and Exceptions		1. The user starts SmartCLIDE and navigates to its search engine 2. The user inputs necessary data (e.g. credentials, type of service/ resource) into the search engine and specifies a URL (or a rancher namespace) to be searched. 3. The user uses Smart CLIDE to search for services and resources. 1. The user starts SmartCLIDE and navigates to its search engine 2. The user inputs necessary data into the search engine (e.g. credentials) and specifies that he/ she wants to search through the REST API 3. The user uses SmartCLIDE to search for services and resources 4. After the initial search, the user switches to a configuration of another environment (e.g. another Kubernetes cluster)

Test Name		WT-0014 Visualising services and data flows
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User), IT Administrator (SmartCLIDE User)
Triggers		The user wants to have a graphical overview of an existing system or service.
Preconditions		One or more services and at least one system exist.
Normal Flow	Description	1. The user navigates to the desired service. 2. The user opens the graphical overview of the service.
	Postconditions	The service is displayed correctly including possible data flows to

Test Name		**WT-0014 Visualising services and data flows**
		and from said service.
Alternative Flows and Exceptions		1. The user navigates to the desired system. 2. The user opens the graphical overview of the system. 3. **Additional Postcondition 1**: The system with all its services and corresponding data flows is displayed correctly. 4. The user groups the services within the system. (e.g. by category, type of data etc.) 5. **Additional postcondition 2**: The services are grouped correctly, based on their meta data.

2.3 Unparallel Innovation Pilot Case validations

The final version of the SmartCLIDE solution is expected to support UNP’s IoT-Catalogue system by assisting users from the IoT-Catalogue community when they need to create new services and improve the IoT-Catalogue itself by indexing and classifying the services inside, thereby providing assistance to UNP to enrich the services descriptions. For users, IoT-Catalogue will provide an embedded IDE based on SmartCLIDE aimed at developing applications using software components described in the IoT-Catalogue itself. The IDE will help to avoid deep technical requirements related to the implementations. Also, this IDE will have a harmonized graphic layout matching the one from the IoT-Catalogue, providing a seamless and fully integrated experience to IoT-Catalogue users.

2.3.1 Validation Environment

The validation environment will focus on the validation of the functional aspects of SmartCLIDE tools, in order to ensure that they are able to process the required information. This implies that SmartCLIDE tools will be tested with IoT domain applications and services. The services to be used on this Use Case are indexed and modelled on the IoT-Catalogue. IoT-Catalogue provides a REST API that provides data about the indexed services and can be used by SmartCLIDE service indexing tools to analyse them to allow their usage for the development of IoT applications.

An example of some services that IoT-Catalogue can provide to SmartCLIDE is represented in Table 2. Services are grouped under three types:

- Data Source – services that provide data sets.
- Processing – services that perform any type of processing over the data.
- Visualization – services that provide mechanism to visualize specific dimensions of the data.

For each service the name is provided, the endpoint used to access the service, the type, and a brief description.

Table 2: IoT-Catalogue Services for use with SmartCLIDE

Name	URL	Type	Description
BigML	https://bigml.com/api	Processing	BigML.io is a Machine Learning REST API to easily build, run, and bring predictive models to your project. You can use BigML.io for basic supervised and unsupervised machine learning tasks and also to create sophisticated machine learning pipelines.
Google Cloud - Bigquery	https://cloud.google.com/bigquery/docs/reference/libraries-overview	Processing	BigQuery is Google Cloud's fully managed, petabyte-scale, and cost-effective analytics data warehouse that lets you run analytics over vast amounts of data in near real time.
Amazon Forecast	https://docs.aws.amazon.com/forecast/index.html	Processing	Amazon Forecast is a fully managed deep learning service for time-series forecasting.
OpenVisionAPI	https://openvisionapi.com/#demo	Processing	Open-source computer vision API based on open source models
World Air Quality Index project	https://aqicn.org/json-api/doc/#api-_	Data Source	Provide information about Air Quality
OpenWeather	https://openweathermap.org/api	Data Source	Provide information about Air Pollution, geocoding and weather information
Here API Traffic	https://developer.here.com/documentation/traffic/dev_guide/topics/what-is.html	Data Source	Provide traffic information about a place
Live Traffic camera	https://opendata.transport.nsw.gov.au/dataset/live-traffic-cameras	Data Source	The API gives Image URL, GPS coordinates, and view description of traffic cameras in GeoJSON format.
Thingspeak	https://thingspeak.com/	Visualisation	Visualiser where user can send the information through api

Name	URL	Type	Description
Weatherunderground	https://www.wunderground.com	Data Source/Visualisation	Has information which comes from weather station deployment across the world.
Thingsboard	https://thingsboard.io	Data Source/Processing/Visualisation	Open-source IoT Platform - Device management, data collection, processing and visualization for your IoT solution

Those services can be combined in different ways to implement specific IoT Applications. For example an application can be developed to identify if there are any relation between traffic car, air quality, air pollution and temperature in a certain place. In this case it can be obtained temperature using the Weather Underground API/OpenWeather API, air quality data can be obtained from the World Air Quality Index project, Air pollution can be collected from the OpenWeather API and information about traffic can be retrieved from HERE API Traffic.

Those data sets can be correlated by resorting to Processing services with the ability to work on a model. Examples of such services are the BigML platform or the BigQuery presented in Google Cloud. It is also possible to use the available data by feeding a different type of model and do a forecast with the gathered data using the Amazon Forecast to, for example, predict how the air quality will evolve in the near future. To show the results, Visualization services like the Thingspeak can be used.

Another possible IoT application is the live detection of traffic jams by using camera livestreams. This application can be based on service like the "Live Traffic camera". Live feed from this API can be analysed through the OpenVisionAPI, which will return the density of detected objects and that information can be used to conclude if there are traffic jams. Visualisation services like the Thingsboard can be used to show these results.

The following summarises the Test Runs that will be executed for validating the full prototype of the SmartCLIDE solution.

2.3.2 Test Run 1: Creation of services

Test Name		UI-0001 Creation of a service from a template
Actors		Developer
Triggers		A developer wants to create a new service for an existing system or a completely new one.
Preconditions		If service templates are provided as an additional plugin, it has to be installed.
Normal Flow	Description	1. The developer has a service for a certain purpose in mind. 2. The developer navigates to the template repository.

Test Name		UI-0001 Creation of a service from a template
		3. The developer chooses an adequate template of a sortable list (e.g. by filtering meta data).
	Postconditions	The system provides the developer with a code skeleton of the chosen service (in a new or existing tag/ file).

Test Name		UI-0002 Create services with data abstraction levels
Actors		Developer
Triggers		A developer wants to implement a set of services (e.g. a database system) and abstract data from said services.
Preconditions		-
Normal Flow	Description	1. The developer composes a set of services. 2. The developer defines the abstraction of data (e.g. data flows, data formats) from the implemented services with the help of the system (e.g. tool tips, auto-completions).
	Postconditions	A set of services with the desired data abstraction levels (data flows, input and output formats) has been created.

Test Name		UI-0006 Non-expert user creates a new service with assistance
Actors		Non-trained user
Triggers		A non-trained user wants to create a new service or add a service to an existing system.
Preconditions		-
Normal Flow	Description	1. The user has a service for a certain purpose in mind. 2. The user navigates to the template repository. 3. The user chooses an adequate template of a sortable list (e.g. by filtering meta data). SmartCLIDE provides short documentation to each template (description, purpose etc.) 4. The user adds the necessary specifics to the template code, SmartCLIDE provides tips and possible solutions via the auto-complete feature.
	Postconditions	A ready to deploy service is created in the process.

Test Name	UI-0008 Test services from within SmartCLIDE
Actors	Developer
Triggers	The user wants to test existing services.

Test Name		**UI-0008 Test services from within SmartCLIDE**
Preconditions		One or more services and corresponding tests already exist and are indexed by SmartCLIDE. The tools required for the execution of the tests are provided in the development environment.
Normal Flow	Description	1. The user uses the tools provided by SmartCLIDE to execute the written tests. 2. The test results are shown.
	Postconditions	SmartCLIDE marks the service(s) as tested.

Test Name		**UI-0011 Create a system using low-code programming**
Actors		Non-trained user
Triggers		A user with limited programming knowledge wants to create a workflow, which is deployable.
Preconditions		None.
Normal Flow	Description	1. The user navigates to the SmartCLIDE's BPMN editor 2. The user draws a workflow using premade functional and decision blocks 3. The user changes the configuration of the workflow through the graphical interface 4. SmartCLIDE supports the user with tool tips and documentation at runtime (to assess the workflow's functionality
	Postconditions	The IDE has generated the needed source code and the workflow is ready for deployment.

Test Name		**UI-0012 Create a complex scenario from templates**
Actors		Non-trained user
Triggers		A user with limited programming knowledge wants to create a more complex system, consisting of multiple services, which is deployable.
Preconditions		-
Normal Flow	Description	1. The user navigates to the SmartCLIDE's BPMN editor. 2. The user chooses a system template. 3. The system is graphically displayed on the canvas. 4. The user models the system on SmartCLIDE's BPMN editor according to its needs. Functionalities of existing services are consumed on the flow. 5. SmartCLIDE supports the user with tool tips and documentation at runtime (to determine whether the system would be functional).

Test Name		UI-0012 Create a complex scenario from templates
	Postconditions	SmartCLIDE has generated the needed source code and the system is ready for deployment.

Test Name		UI-0013 Customize colour scheme and graphical elements
Actors		Developer, IT Administrator
Triggers		A user dislikes the default colour scheme or the display of certain elements.
Preconditions		-
Normal Flow	Description	1. The user navigates to the SmartCLIDE settings. 2. The user modifies the overall colour scheme of SmartCLIDE 3. The user changes the display of some graphical elements (such as icons for services for example).
	Postconditions	The colour scheme and graphical elements are now displayed according to the user's inputs.

Test Name		UI-0014 Visualising workflows
Actors		Developer, IT Administrator
Triggers		The user wants to have a graphical overview of an existing system.
Preconditions		One or more services and at least one workflow exist.
Normal Flow	Description	1. The user navigates to the desired workflow and selects the option to edit it. 2. The SmartCLIDE's BPMN editor opens providing the need tools for visualising and editing the workflows.
	Postconditions	The workflow is correctly displayed including possible data flows to and from the said system.

Test Name		UI-0015 Accessing Git repositories
Actors		Developer
Triggers		The developer accesses a Git repository to create or edit a workflow.
Preconditions		A workflow is stored in a Git repository
Normal Flow	Description	1. The user opens the SmartCLIDE's BPMN editor and selects a workflow to be imported. 2. Workflow diagram is loaded and shown to the User 3. User is able to edit the diagram

Test Name		UI-0015 Accessing Git repositories
	Postconditions	A fully editable workflow is loaded and presented to the user

Test Name		UI-0016 Change something on SmartCLIDE itself
Actors		Developer
Triggers		A developer wants to change or add functionality of SmartCLIDE or fix an issue.
Preconditions		The SmartCLIDE code basis needs to be publicly accessible. The repository includes automated (acceptance) tests
Normal Flow	Description	1. The user clones the SmartCLIDE repository. 2. The user creates a new local branch. 3. The user makes the desired changes. 4. The user commits the changes 5. The user pushes the changes and creates a merge request for the maintainers to review.
	Postconditions	A merge request has been created and is available for the maintainers to review.

Test Name		UI-0017 Ask for changes in SmartCLIDE
Actors		Developer, IT Administrator
Triggers		A user finds a bug or wants to demand a new feature.
Preconditions		-
Normal Flow	Description	1. The user goes online and visits the public repository of SmartCLIDE 2. The user navigates to the issue tracker 3. The user creates a new issue and states what is faulty or what change is wanted.
	Postconditions	A new issue is created and the maintainers of the repository are notified.

Test Name		UI-0018 Language support
Actors		Developer, Quality Manager, IT Administrator
Triggers		A user wants to create a new service or change an existing one.
Preconditions		The Python and JavaScript languages are supported by default. If needed additional language support packages need to be installed.
Normal Flow	Description	1. The user creates a new file or opens an existing one.

Test Name		UI-0018 Language support
		2. The user implements the desired solution, while SmartCLIDE provides the syntax highlighting.
	Postconditions	SmartCLIDE shows support for the selected language by showing correct syntax highlighting.

Test Name		UI-0019 Access SmartCLIDE via a browser
Actors		Developer, Quality Manager, IT Administrator
Triggers		A user wants to access SmartCLIDE from within a browser, since it might not be installed on the local machine.
Preconditions		A SmartCLIDE container or virtual machine has to be deployed somewhere reachable (local docker container, remote access to virtual machine etc.)
Normal Flow	Description	1. The user opens a browser. 2. The user navigates to the correct URL and opens SmartCLIDE.
	Postconditions	All of SmartCLIDE's functionality is available via the web interface.

Test Name		UI-0020 Specify the licence of a service
Actors		Developer, Quality Manager
Triggers		A user wants to specify the license of a service without knowing, which one would be appropriate.
Preconditions		An implemented service exists without any license specified.
Normal Flow	Description	1. The user opens the source code of the service. 2. The user navigates to the license mechanism of SmartCLIDE. 3. The user filters the possible licenses by providing information regarding the service (purpose, commercial or not). 4. The user chooses a license based on the suggestion of SmartCLIDE.
	Postconditions	The license is inserted at the top of the service's source code.

Test Name	UI-0022 Usage of a BPMN editor
Actors	Developer, Quality Manager, Business Analyst
Triggers	A user wants to create a process model in a web browser.
Preconditions	Possible third-party BPMN editor

Test Name		**UI-0022 Usage of a BPMN editor**
Normal Flow	Description	1. The user opens SmartCLIDE in a web browser. 2. The user models a business process using a BPMN editor..
	Postconditions	The business process is displayed correctly.

Test Name		**UI-0026 Create secure services with authentication**
Actors		Developer, Quality Manager
Triggers		A user wants to create a secure service or a set of secure services.
Preconditions		The SmartCLIDE code basis needs to be publicly accessible. The repository includes automated (acceptance) tests
Normal Flow	Description	1. The user implements a service. 2. The user adds support for authentication with the assistance of SmartCLIDE. 3. The user deploys the service.
	Postconditions	The service is ready to use, a potential user has to provide valid credentials to use its functionality.

Test Name		**UI-0030 Configure an existing service**
Actors		Developer, Quality Manager
Triggers		The user wants to use an existing service in the workflow being designed.
Preconditions		At least one configurable implemented service exists.
Normal Flow	Description	1. The user selects an existing service which requires some configurations. 2. The user specifies some configurations of the service. 3. The service is imported to the workflow with the specified configurations.
	Postconditions	The service behaviour matches the specified configurations.

Test Name		**UI-0007 Decompose complex systems into smaller pieces**
Actors		Developer, Quality Manager
Triggers		A user wants to lower the complexity of a system / problem by separating it into smaller pieces.
Preconditions		None. The need to decompose a system occurred on the user's side and does not necessarily need anything from the system in the first place.

Test Name		UI-0007 Decompose complex systems into smaller pieces
Normal Flow	Description	1. The user imports an existing system, which is known by SmartCLIDE into the workflow tool. 2. The user uses SmartCLIDE's workflow tools to remodel the system's parts individually. 3. The user is able to change metadata for the individual pieces. 4. The user changes flows (e.g. of data, input, output, dependencies, general process) between the pieces with the help of the workflow tool and is able to create new dependencies or flows between the existing pieces.
	Postconditions	A decomposed overview of the system/ problem exists within SmartCLIDE

2.3.3 Test Run 2: Classification of services

Related Tests: *UI-0015 Accessing Git repositories, UI-0016 Change something on SmartCLIDE itself, UI-0017 Ask for changes in SmartCLIDE, UI-0018 Language support, UI-0019 Access SmartCLIDE via a browser* and *UI-0020 Specify the licence of a service.*

Test Name		UI-0028 Indexing and classification of services from an external repository
Actors		SmartCLIDE backend, External Services Repository
Triggers		SmartCLIDE backend wants to periodically index and classify existing services from an external repository.
Preconditions		An external repository that contains one or more services for further processing.
Normal Flow	Description	1. SmartCLIDE accesses a repository in a pre-configured address. 2. SmartCLIDE obtains a list of services. 3. SmartCLIDE indexes and classifies the received list.
	Postconditions	An indexed list of services from an external repository classified by relevant categories.

2.3.4 Test Run 3: Estimate deployment costs

Related Tests: *UI-0013 Customize colour scheme and graphical elements, UI-0014 Visualising workflows, UI-0015 Accessing Git repositories, UI-0016 Change something on SmartCLIDE itself, UI-0017 Ask for changes in SmartCLIDE, UI-0018 Language support, UI-0019 Access SmartCLIDE via a browser* and *UI-0020 Specify the licence of a service.*

Test Name		**UI-0009 Conduct a cost analysis**
Actors		Developer, Quality Manager, IT Administrator
Triggers		The user wants to have an estimation of operating costs if the system under inspection would be deployed in a production environment.
Preconditions		An existing system consisting of one or more services that is ready for deployment.
Normal Flow	Description	1. The user uses the service registry of SmartCLIDE to filter the services needed for the system to be deployed. 2. The user selects the target environment (such as cloud provider). 3. The user selects all services they want to include in the costs calculation or the system as a whole. 4. The user checks whether the existing metadata about the system/ services is sufficient to conduct a cost analysis (maybe warnings or tool tips from SmartCLIDE to assist with that). 5. The user conducts the actual cost analysis.
	Postconditions	SmartCLIDE provides the user with an overview of costs the system would cause as a whole and also a breakdown of the costs for each service individually. Listed costs consist of items regarding RAM, storage, CPU, usage frequency etc.

Test Name		**UI-0021 Define and configure QoS**
Actors		Developer, Quality Manager
Triggers		A user wants to define or evaluate QoS configurations regarding resources.
Preconditions		At least one already implemented service exists.
Normal Flow	Description	1. The user defines the configuration of a service with the use of functional blocks provided by SmartCLIDE. 2. The user specifies additional dependencies and QoS constraints to other services. 3. User starts the calculation of new deployment costs.
	Postconditions	The added configuration is shown within the meta data of the services. User can assess the impact of the QoS constraints in the deployment costs.

2.4 CONTACT Software Test Runs

The system used for SmartCLIDE validations will mainly be CONTACT Elements Platform and CONTACT Elements for IoT, since this system consists of a lot of microservices working together and also uses Eclipse technology. In addition, it is important to also validate metrics regarding process enhancements made possible by SmartCLIDE. This will include typical indicators such as time to delivery or average time being used to work on an issue. Key interests for SmartCLIDE is being able to more easily involve customers in the definition of new features, enabling their automatic validation, smart classification of features (i.e. ELEMENT's building blocks), and composition of services to easily enhance existing services or create new services.

2.4.1 Validation Environment

The validation environment that will be used for testing the full prototype of the SmartCLIDE solution will include GitLab (used as version control system), GitLab CI, Docker, Kubernetes, Helm and SonarQube.

All of our packages and applications are managed in an on premise instance of GitLab, also using GitLab CI to build, test and analyze the software. The separate CI jobs are executed on Docker and Windows CI runners using Docker images built beforehand (mostly based on Ubuntu). The Windows runners are all configured the same to allow comparability and run Windows Server 2016 as the operating system with additional software installed that provide all the tools necessary for the building and testing process. The code is analyzed by the sonar-scanner in later jobs of the CI pipeline and the results are uploaded to the local SonarQube instance. All branches of all repositories also deploy the current state of the application to an on-premise Kubernetes cluster mostly using Helm charts.

SmartCLIDE itself will probably either installed on the developer's machines separately or deployed on the cluster centrally and used by the users. For the early validation, SmartCLIDE will mainly be used by QA and other developer teams that are yet to be decided. Testing of the use and test cases may either be done on developer machines with locally running Docker images or deploying images and services to a dedicated namespace on the company-wide used Kubernetes cluster.

The following summarises the Test Runs that will be executed for validating the full prototype of the SmartCLIDE solution.

2.4.2 Test Run 1: Collaborative development

Test Name		**CO-0001 Creation of a service from a template**
Actors		Developer (SmartCLIDE User)
Triggers		A developer wants to create a new service for an existing system or a completely new one.
Preconditions		None. It should always be possible to create a new service from a template. If service templates are provided as an additional plugin, it has to be installed.
Normal Flow	Description	1. The developer has a service for a certain purpose in mind. 2. The developer navigates to the template repository

Test Name		CO-0001 Creation of a service from a template
		3. The developer chooses an adequate template of a sortable list (e.g. by filtering meta data)
	Postconditions	The system provides the developer with a code skeleton of the chosen service (in a new or existing tag/ file)

Test Name		CO-0003 Create and deploy a service from the IDE
Actors		Developer (SmartCLIDE User)
Triggers		A developer wants to create a new service for an existing system or a completely new system one using command line tools.
Preconditions		Necessary packages installed on the host in order to provide SmartCLIDE with the ability to use them. In addition to that, all necessary tools for a local Kubernetes instance are installed.
Normal Flow	Description	1. The developer uses SmartCLIDE to implement a new service. 2. The developer uses the built-in functionality to verify the written code with an installed linter(e.g. using Pylint for Python code) 3. The developer uses the built-in functionality to deploy the new service (whilst providing SmartCLIDE with a valid configuration) with kubectl in a local Kubernetes environment.
	Postconditions	The service has been successfully deployed.
Alternative Flows and Exceptions		1. The developer uses SmartCLIDE to implement a new service. 2. The developer uses the built-in functionality to verify the written code (e.g. using Pylint for Python code) 3. The developer uses the built-in functionality to deploy the new service with kubectl using a saved configuration within SmartCLIDE

Test Name		CO-0004 Discover resources and services
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		A SmartCLIDE user wants to discover existing services of a system.
Preconditions		A system that consists of one or more services is deployed/ in use or just known to SmartCLIDE (via the registry).
Normal Flow	Description	1. The user starts SmartCLIDE and navigates to its search engine 2. The user inputs necessary data into the search engine (e.g. credentials, type of service/ resource) and specifies

Test Name		CO-0004 Discover resources and services
		that he/ she wants to search through the REST API 3. The user uses SmartCLIDE to search for services and resources
	Postconditions	SmartCLIDE provides the user with an overview of services and resources within the searched system

Test Name		CO-0006 A non-expert user creates a new service with assistance
Actors		Non-trained user (SmartCLIDE user)
Triggers		A non-trained user wants to create a new service or add a service to an existing system.
Preconditions		None.
Normal Flow	Description	1. The user has a service for a certain purpose in mind. 2. The user navigates to the template repository. 3. The user chooses an adequate template of a sortable list (e.g. by filtering meta data). SmartCLIDE provides short documentation to each template (description, purpose etc.) 4. The user adds the necessary specifics to the template code, SmartCLIDE provides tips and possible solutions via the auto-complete feature.
	Postconditions	A ready to deploy service is created from the process.
Alternative Flows and Exceptions		1. The user navigates to the SmartCLIDE registry of deployed services. 2. The user selects the system to which a service should be added. 3. SmartCLIDE provides the user with an overview of the system and its dependencies (maybe even some sort of documentation). 4. The user navigates to the template repository. 5. The user chooses an adequate template of a sortable list (e.g. by filtering meta data). SmartCLIDE provides short documentation to each template (description, purpose etc.) 6. The user adds the necessary specifics to the template code, SmartCLIDE provides tips and possible solutions via the auto-complete feature. 7. The user deploys the created service to the existing system. 8. **Additional postcondition**: The deployed service finds its way into SmartCLIDE's service registry.

Test Name		**CO-0008 Test services from within SmartCLIDE**
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		The user wants to test already existing services.
Preconditions		One or more services and corresponding tests already exist. The needed command line tool to execute the tests is installed on the host.
Normal Flow	Description	1. The user uses the built-in command line of SmartCLIDE to execute the written tests. 2. The command line prints the results.
	Postconditions	SmartCLIDE marks the service(s) as tested or adds the test coverage to the meta data of the service.

Test Name		**CO-0009 Conduct a cost analysis**
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User), IT Administrator (SmartCLIDE User)
Triggers		The user wants to have an estimation of operating costs if the system under inspection would be deployed as a productive system.
Preconditions		An existing system consisting of one or more services that is ready for deployment.
Normal Flow	Description	1. The user uses the service registry of SmartCLIDE to filter the services needed for the system to be deployed and functional. 2. The user selects the target environment (such as cloud provider). 3. The user selects all of the needed services or the system as a whole. 4. The user checks whether the meta data of the system/ services is sufficient to conduct a cost analysis (maybe warnings or tool tips from SmartCLIDE to assist with that) 5. The user conducts the actual cost analysis.
	Postconditions	SmartCLIDE provides the user with an overview of costs the system would cause as a whole and also a breakdown of the costs for each service individually. Listed costs consist of items regarding RAM, storage, CPU, usage frequency etc.
Alternative Flows and Exceptions		1. A workflow for the desired functionality is already in place. 2. The user triggers a cost analysis via SmartCLIDE.

Test Name	**CO-0011 Create a system using low-code programming**
Actors	Business Stakeholder (SmartCLIDE User)

Test Name		CO-0011 Create a system using low-code programming
Triggers		A user with limited programming knowledge wants to create a service, which is deployable.
Preconditions		None.
Normal Flow	Description	1. The user navigates to the modelling canvas of SmartCLIDE 2. The user 'draws' a service using premade functional and decision blocks. 3. The user is able to change the configuration of the service via a graphical interface.
	Postconditions	The drawn service is saved by the system.

Test Name		CO-0015 Accessing Git repositories
Actors		Developer (SmartCLIDE User)
Triggers		The developer wants to push changes made to an existing service or a new file to a Git repository.
Preconditions		The necessary tools (e.g. Git bash) are installed on the host, the source code has been cloned.
Normal Flow	Description	1. The user makes changes to an existing service. 2. The user saves the file. 3. The user commits the changes via the built-in CLI. 4. The user pushes the commits to a remote repository via the built-in CLI.
	Postconditions	The commits have been successfully pushed to the defined repository.
Alternative Flows and Exceptions		1. The user makes changes to an existing service. 2. The user saves the file. 3. The user uses the context menu to commit the changes. 4. A window pops up, in which the user enters the commit message. 5. The user uses the context menu to push the commits to a remote repository. 6. A window pops up if the repository is private and no credentials are available.

2.4.3 Test Run 2: Performance analysis

Related Tests: *CO-0004 Discover resources and services.*

Test Name	CO-0005 Search for deployed services
Actors	Developer (SmartCLIDE User), Quality Manager (SmartCLIDE

Test Name		**CO-0005 Search for deployed services**
		User)
Triggers		The user wants to look up already deployed services.
Preconditions		Several services have been implemented and deployed via SmartCLIDE.
Normal Flow	Description	1. The user navigates to the SmartCLIDE registry (e.g. via a tool bar) 2. The user filters the registry in order to specify the search results
	Postconditions	SmartCLIDE provides the user with a list of deployed services based on the inputs/ filters given by the user.
Alternative Flows and Exceptions		1. The user uses the built-in command line tool to look up deployed services. 2. The user specifies additional arguments to the command line call to filter the search results.

Test Name		**CO-0025 Monitor and verify services**
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		A user wants to monitor deployed services.
Preconditions		A deployment of a set of services exists.
Normal Flow	Description	1. The user opens SmartCLIDE and navigates to the deployed system. 2. SmartCLIDE provides the user with live information of the deployed services. 3. The user selects the information to be monitored.
	Postconditions	The desired information are gathered and displayed to the user.
Alternative Flows and Exceptions		1. In addition to the normal flow: The user defines thresholds for different metrics of the system (CPU workload, HDD capacity etc.). 2. SmartCLIDE notifies the user if thresholds are broken or failures occur.

2.4.4 Test Run 3: Improve code quality of service

Related Tests: *CO-0001 Creation of a service from a template*, *CO-0003 Create and deploy a service from the IDE* and *CO-0004 Discover resources and services.*

Test Name	**CO-0010 Deploy a service from the CLI within SmartCLIDE**
Actors	Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)

Test Name		**CO-0010 Deploy a service from the CLI within SmartCLIDE**
Triggers		A user wants to deploy an already implemented service via SmartCLIDE. The necessary tools (e.g. Kubectl, AWS, Helm, Docker) are installed.
Preconditions		A service ready to be deployed exists.
Normal Flow	Description	1. The user navigates to the built-in command line interface. 2. The user provides the system with a valid configuration or uses a saved one within SmartCLIDE. 3. The user deploys the implemented service via the CLI.
	Postconditions	The service has been deployed accordingly.

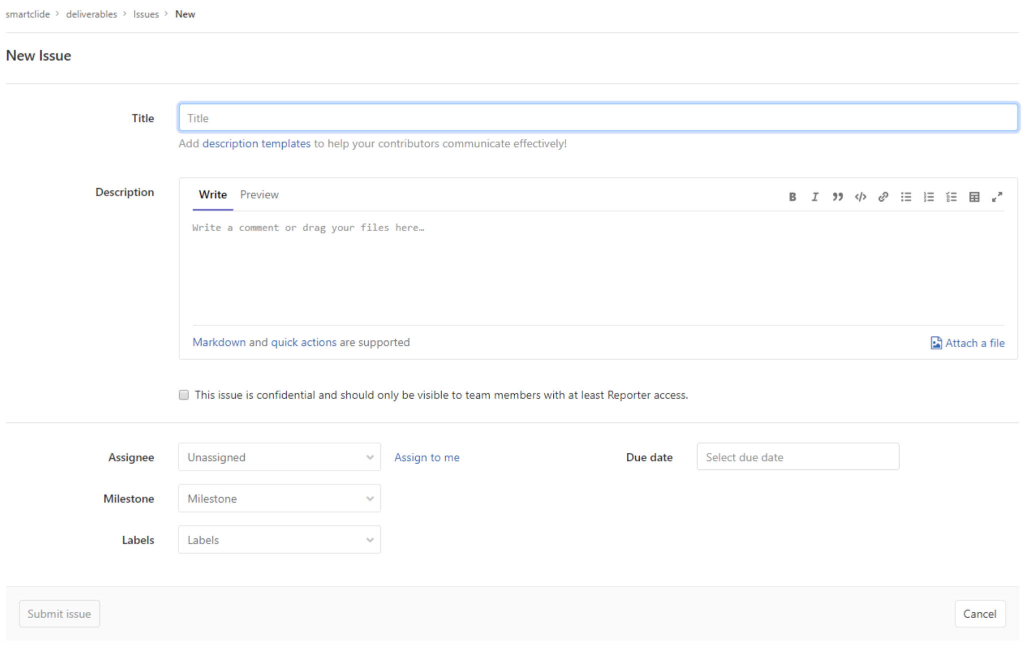
Test Name		**CO-0012 Create a complex scenario from templates**
Actors		Business Stakeholder (SmartCLIDE User)
Triggers		A user with limited programming knowledge wants to create a more complex system, consisting of multiple services, which is deployable.
Preconditions		A valid configuration to deploy a system to a cloud service or cluster exists. (This would be needed to fill certain variables to make the system actually deployable)
Normal Flow	Description	1. The user navigates to the modelling canvas of SmartCLIDE. 2. The user chooses a system template (consisting of one or more services). 3. The system is graphically displayed on the canvas 4. The user models the system on the canvas according to its needs. 5. SmartCLIDE supports the user with tool tips and documentation at runtime (to determine whether the system would be functional).
	Postconditions	SmartCLIDE has generated the needed source code and the system is ready for deployment.

Test Name		**CO-0024 Use SmartCLIDE's container repository**
Actors		Developer (SmartCLIDE User), Quality Manager (SmartCLIDE User)
Triggers		A user wants to upload a Docker image and share it.
Preconditions		A Docker image has to exist.
Normal Flow	Description	1. The user opens SmartCLIDE and navigates to the built-in container registry. 2. The user uploads a local Docker image to the registry.

Test Name		CO-0024 Use SmartCLIDE's container repository
	Postconditions	The Docker image has been uploaded and is available for other users.
Alternative Flows and Exceptions		1. In addition to the normal flow: Another user pulls the recently uploaded image locally.

3 Bug Tracking

During the validation procedures, Pilot Case partners will utilise GitHub for reporting bugs and issues with the full prototype delivery of the SmartCLIDE technologies. Every Pilot Case team member is able to register new issues in GitHub during each validation cycle (see Figure 2).


smartclide › deliverables › Issues › New

New Issue

Title: Title

Add description templates to help your contributors communicate effectively!

Description: Write Preview

Write a comment or drag your files here…

Markdown and quick actions are supported — Attach a file

This issue is confidential and should only be visible to team members with at least Reporter access.

Assignee: Unassigned — Assign to me — Due date: Select due date

Milestone: Milestone

Labels: Labels

Submit issue — Cancel

Figure 2: Issue Creation Form in GitHub

The category of issues to be reported by Pilot Case partners are defined in alignment with the project's agile software development methodology. In particular, each Pilot Case team member will be able to create the following types of issues:

- **User Stories** – are used for the definition of needed functional or technical features discovered during the validation testing. User stories are requested to the research and development team from the perspective of an end user of the SmartCLIDE IDE represented capabilities needed to achieve expected outcomes.
- **Epics** – represent large bodies of work that can typically be broken down into a number of smaller elements (i.e. User Stories). They would be expected to be less common from validation testing and they represent substantial increases in technical or functional capabilities to achieve outcomes expected by SmartCLIDE IDE end users.

- **Bugs** – represent errors found in the use of the SmartCLIDE IDE or other elements (e.g. installation) which should be corrected as part of the final prototype development at the end of the project.

Every issue reported shall include the following information:

- Originating Pilot Case
- Pilot Case team member reporting the issue
- Specific component or set of components that were in use during the validation testing when the issue was identified
- The Pilot Case validation test run being executed (if applicable) when the issue was identified
- Capability, benefit, or innovation expected from the SmartCLIDE IDE that is either unavailable, obfuscated or limited in scope due to the issue

By reporting issues in GitHub, the SmartCLIDE Pilot Case partners and research and development team members will ensure that:

- Problems at any stage of the validation process are documented and can be corrected and used for process improvement;
- Reported issues and their associated corrective actions are implemented in accordance with Pilot Case partner approved solutions;
- Feedback is provided to the technical teams and the Pilot Case team members of issue status, and
- Data is provided for measuring and predicting the degree to which the SmartCLIDE IDE is providing the right features and capabilities to the representative end users.

Monitoring of issues with respect to any new issues introduced, progress in resolving open issues, and any requested or implied design or development implications will be discussed at each of the regular research and development team conference calls.

4 Performance Indicators and Targets

The SmartCLIDE workplan distinguishes between validation testing of the early and full prototypes (WP4) and assessment (WP5) of the final SmartCLIDE IDE. The latter addresses the effectiveness of the new platform in providing measurable business, operational and organisational benefits for each of the industrial Pilot Case partners. The tasks under WP4 validate that project technologies provide the features and capabilities that are needed (i.e. the right product was developed), while the tasks under WP5 set out to quantify the impact of those features and capabilities by specifying and implementing Assessment Scenarios designed to measure the Key Performance Indicators (KPI) for each Pilot Case.

In carrying out the validations under WP4 as described in Section 2, it is important for Pilot Case partners to take into consideration the overall KPIs established for the project and to identify in the validation process any issues concerning features and functions that might limit or prevent the project from achieving the targets established for each KPI. The KPIs, targets to be achieved, and the focus area of each Pilot Case for the assessments to be carried out under WP5 are summarised in Table 3.

Table 3: Pilot Case KPIs, Targets, Assessment Methods and Assessment focus areas

No.	Performance Indicator	Sufficient	Good	Excellent	Assessment Method	Pilot Case Focus (P=Primary, S=Secondary)			
						INTRA	WT	UNP	CNTCT
1	Software components reutilisation rate	35%[1]	40%	65%	Analysing historical projects records and comparing the results to the project(s) developed with SmartCLIDE.	P	S	P	S
2	Reduction in number of errors reported (i.e. by type and stage) as major and minor issues	10%	30%	50%	Side-by-side developments with and without SmartCLIDE.	S	S	S	P
3	Reduction in time for resolving errors	10%	30%	50%	Side-by-side developments with and without SmartCLIDE.	P	S	S	P
4	Reduction in time taken to develop a new application by non-technical staff (compared to the traditional learning curve at the company)	5%	20%	30%	Comparison of application development by non-technical staff with and without SmartCLIDE – based on experiments or against historical data.	P	P	P	S
5	Reduction in time to deploy a new application or significant feature requested by an end user	10%	25%	50%	Side-by-side developments with and without SmartCLIDE, or compared against historical projects data.	P	P	P	S
6	Reduction in lifecycle costs (i.e. specification and planning, implementation, testing, and deployment)	5%	15%	30%	Side-by-side developments with and without SmartCLIDE.	P	P	S	S
7	Reduction in number of incidents where developed system was not aligned with user requirements	20%	50%	90%	Side-by-side developments with and without SmartCLIDE.	S	P	S	S
8	Reduction in Cloud services costs due to optimisation and efficient use of Cloud resources	5%	15%	30%	Compare two similar applications developed with and without SmartCLIDE.	S	P	P	S
9	Increase in the number of security vulnerabilities detected	10%	15%	25%	Side-by-side developments with and without SmartCLIDE, or compared against historical projects data.	S	P	S	P
10	Improved transparency, readability and comprehensibility of software (i.e. by using coding-by-demonstration principle)	Average score of "Some"	Average score of "Significant"	Average score of "Substantial"	Performing a micro-survey of internal software development responsibles and averaging their assessment of improvements using a scale of None, Some, Significant and Substantial.	S	S	P	S

[1] Current estimated component reutilisation rates noted are INTRA: 30%; WT: 35%; UNP: 25%; CONTACT: 25%

5 Conclusion

This deliverable has described the validation procedures and specific validation tests that will be utilised by the Pilot Case partners to validate the full prototype of the SmartCLIDE solution has provided the needed capabilities for effective Cloud-based development of industrial applications. The validation environments used by each Pilot Case partner are representative of industrial software development environments and commercial applications and through the described Bug Tracking facilities, will provide important feedback and guidance to the research and development teams as the final prototype developments are completed by the end of the project. Overall performance indicators and targets that will form the basis for the Assessment Scenarios under Workpackage 5 are also noted for due consideration during the validation testing to identify any bottlenecks or other issues that might later affect the industrial assessment where the business, operational and other impacts delivered by the project technologies are quantified by each of the industrial Pilot Case partners.