D9.10 – Quality Assurance Plan

Project Acronym	NIMBLE
Project Title	Collaboration Network for Industry, Manufacturing, Business and Logistics in Europe
Project Number	723810 (H2020)
Work Package	WP9: Project Management
Responsible author	Wernher Behrendt (Salzburg Research)
Dissemination Level	Confidential (consortium & Commission services)
Contractual Delivery Date	30.11.2016
Actual Delivery Date	18.01.2017
Version	V.4.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 723810

Table of contents

Table of contents **2**
Document Information **3**
NIMBLE in a Nutshell **4**
Executive Summary **5**
1 Summary of Project Objectives **6**
1.1 Milestones of the Project 8
2 Summary of Project Risks and Mitigation Measures **9**
3 Summary of QA Procedures **12**
3.1 QA for Deliverables 12
3.2 QA for Research Results / Papers 13
3.3 QA for Software 13
3.3.1 Requirements Definition 13
3.3.2 Software Design 14
3.3.3 Source code management 15
3.3.4 Configuration management 15
3.3.5 Documentation management 16
3.3.6 Testing and validation 16
3.3.7 Release management 17
3.3.8 Issues management 18
3.3.9 Software Design and Development Methods 19
3.3.10 QA for Security 20
4 Project Infrastructure **22**
4.1 Communication 22
4.2 Collaboration 22
4.3 Development & Testing 23
4.4 Administration 23
4.4.1 NIMBLE Office 23
4.4.2 Financial Management and Resource Controlling 24
5 Appendix – QA Template for Reviews of Deliverables **25**

Document Information

Project	NIMBLE (H2020-723810)
Identifier	NIMBLE-D9.10
Author(s):	Wernher Behrendt (Salzburg Research)
Document title:	NIMBLE Quality Assurance Plan
Source Filename:	NIMBLE_D9_10_Nimble_QA_Plan_20170118.docx
Dissemination level	Confidential (consortium & Commission services)

Document context information

Work package/Task	Task 9.3
Responsible person and project partner:	Wernher Behrendt (Salzburg Research)

Quality Assurance / Review

Name / QA / Release / Comment	Violeta Damjanovic-Behrendt, Benny Mandler (IBM) Ready for release

Citation information

Official citation	NIMBLE Consortium (2016), NIMBLE Project QA Plan

Document History

V	Name	Date	Remark
1.0	W. Behrendt	31.10. 2016	Initial QA Plan – Draft
2.0	QA Team	10.01. 2017	Comments on QA Plan
3.0	W. Behrendt	14.01. 2017	Final contributions included (Software QA)
4.0	W.Behrendt	18.01. 2017	Final proof-read, edits and submission

Copyright Notice

The views expressed in this document are those of the authors and do not necessarily reflect the policies of the European Commission.

NIMBLE in a Nutshell

NIMBLE is the collaboration Network for Industry, Manufacturing, Business and Logistics in Europe. It will develop the infrastructure for a cloud-based, Industrie 4.0, Internet-of-things-enabled B2B platform on which European manufacturing firms can register, publish machine-readable catalogs for products and services, search for suitable supply chain partners, negotiate contracts and supply logistics. Participating companies can establish private and secure B2B and M2M information exchange channels to optimise business work flows. The infrastructure will be developed as open source software under an Apache-type, permissive license. The governance model is a federation of platforms for multi-sided trade, with mandatory interoperation functions and optional added-value business functions that can be provided by third parties. This will foster the growth of a net-centric business ecosystem for sustainable innovation and fair competition as envisaged by the Digital Agenda 2020. Prospective NIMBLE providers can take the open source infrastructure and bundle it with sectorial, regional or functional added value services and launch a new platform in the federation. The project started in October 2016 and will last for 36 months.

Executive Summary

This is the quality assurance plan D9.10 of the NIMBLE project summarizing and detailing relevant sections of the DoA.

It serves also as a handbook for researchers and managers in the NIMBLE project.

1 Summary of Project Objectives

To deliver the envisaged NIMBLE platform, we have to achieve five high-level objectives:

- Develop the NIMBLE collaboration infrastructure with core services;
- Ensure ease-of-entry and ease-of-use of the platform;
- Grow the use of the platform - this is a top-priority - see our impact strategy!
- Enhance NIMBLE functionality from the core services and ensure that firms master it on their own;
- Ensure trust in the platform – this requires synthesis of security, privacy, reputation and information quality.

The following tables show a more detailed view of the objectives:

1. Develop the collaboration infrastructure with core services → to be developed by month 10	
1.1	Establish with stakeholders, the requirements for core services of the platform. Establish with the stakeholders early success criteria. *(Please see the impact section for metrics of early uptake)*
1.2	Design the top-level architecture and modules. We will be guided by the Industrial Internet Reference Architecture (IIRA) and the Reference Architecture Model Industrie 4.0 (RAMI)
1.3	Use permissive open source software wherever possible. We have analysed Apache and FIWARE and will base the work in NIMBLE on existing tools that have a combined value of over 100m €[1].
1.4	Deploy the basic infrastructure with core services, to industrial use case partners in the consortium
1.5	Learn from early validation (release early, release often approach)
2. Ensure Ease of Entry and Ease of Use → to be demonstrated in month 15	
2.1	A company can publish its product catalogue in bulk or via semantic product descriptions, at different levels of granularity and completeness.
2.2	Two companies can establish private, encrypted information channels for a business collaboration. In NIMBLE phase two, arbitrary supply chains can be established between any number of firms.
2.3	Services for matchmaking between producers and consumers (B2B, B2C) are available to establish business collaborations fast.
2.4	Joint planning and real-time adjustment of contractual commitments to gain mutual benefits from shared information leading to optimized re-planning, instead of sticking to outdated default plans.
2.5	Data collection, management and analytics: Firms can choose data collecting, management and analytics services freely, from different providers that are all available on the same platform.

[1] According to the estimation method used by openhub: e.g. https://www.openhub.net/p/cloudfoundry/estimated_cost

3. Growing the use of the platform → 100+ early adopters testing the platform services by month 24	
3.1	Each of the use cases demonstrates benefits for businesses of the sector (white goods, textiles, furniture, ready-built homes) leading to a "me too" effect
3.2	Start early adopter scheme, recruiting external industrial users of the platform
3.3	Provide a core software tool set to initiate the software supply side of the platform
3.4	Continually improve business integration pushing further down barriers to entering the platform
3.5	Ensure "balanced multi-sidedness" of the platform by specialising the integration tools to the needs of different sectors (manufacturing, retail, logistics, etc.).
4. Mastering the platform and achieving higher maturity levels → advanced services by month 30	
4.1	Develop a methodology for establishing B2B collaboration with the help of asset virtualization mechanisms, going from enterprise level to operations management down to shop-floor control
4.2	Adapt collaboration methodologies for SMEs that are on varying digitalization levels on ISA-95 hierarchy or that have no digitalization at all, yet.
4.3	Establish rules of governance - ideally, these will emerge as platform-etiquette
4.4	Establish KPIs and benchmark methods – the platform will automate many of these methods
4.5	Based on benchmarks, establish measures of the economic value of the platform and its services
4.6	Develop novel features of the platform and encourage others through the open source approach, to join our development efforts and to add further common value
5. Ensuring Trust, Security, Privacy, Reputation and Information Quality → from day one !	
5.1	The platform will support user-adjustable levels of security and privacy in order to maintain customer trust in balance with ease of use
5.2	The platform will be designed modular and resilient so that security breaches can never "sink the whole ship"
5.3	Data storage must be entirely at the owner's control - from cloud to storage on personal devices
5.4	Strength of encryption will be controlled by the business actors who communicate
5.5	The platform will be develop as a federated ecosystem. The principle of subsidiarity will ensure *co-opetition* and service innovation. This "many-sidedness" will contribute to its overall stability.
5.6	Grow trust on the platform by a) fair gain distribution among the platform sides; b) maintaining strict interoperability; c) providing privacy in B2B communication and data exchange
5.7	Information quality will be a fundamental value to be maximised in the platform. Wherever possible this will be achieved by automation and semantic modelling.

The objectives will be revisited at quarterly meetings to ensure work progress is in line.

1.1 Milestones of the Project

Number	Name	Lead Beneficiary	Delivery Date (Annex I)
1	Project Start	SRFG	01 Nov 2016
2	Project Websites (external and internal) created	SRFG	01 Jan 2017
3	Use case requirements established	UNI-HB	01 Apr 2017
4	Platform technology baseline established	SRDC	01 May 2017
5	First core platform prototype finalized	IBM ISRAEL	01 Aug 2017
6	Security and privacy for core business services established	SRFG	01 Nov 2017
7	First use case is processed via platform services	LTU	01 Jan 2018
8	First external firms can join the platform and establish supply chains	INNOVA SRL	01 May 2018
9	100+ external adopters	INNOVA SRL	01 Oct 2018
10	Advanced platform services are functional	HOLONIX	01 Apr 2019
11	1500+ external adopters	INNOVA SRL	01 Jul 2019
12	Project Termination	SRFG	01 Oct 2019

As can be seen from the milestones, we expect to have a core NIMBLE system up and running before the end of the first project year.

The milestones are in line with the project work plan and with the project objectives and are of course, also revisited in quarterly progress reviews.

2 Summary of Project Risks and Mitigation Measures

Foreseen critical risks are reported in the "Critical Risks" section of SyGMa, the Commission's system for continuous reporting:

https://ec.europa.eu/research/participants/grants-app/reporting/DLV-723810

For the sake of redundancy, these are:

RISK	Probability	Impact	Mitigation approach
IBM as research partner and cloud provider leaves the consortium	Low	High	IBM ISRAEL offers a great variety of solutions, and IBM ISRAEL has strong experience with cloud centric solutions. IBM assessed the innovation potential of NIMBLE and committed to the project. The credibility of the company is beyond doubt. In the very unlikely case of IBM leaving the consortium they would give the consortium a grace period of at least 6 months in which a substitute could be found. If no substitute can be found within the grace period then the continuation of the project has to be re-assessed.
Failure in acquiring the planned number of early adopters for the NIMBLE platform	Probability: Medium	Impact: High	This is a critical risk because the consortium commits to the presented approach and the proposed transition from dissemination-driven recruitment to self-sustained growth of the platform is an unknown for European projects in this programme line. This is why we designed the AMBASSADOR and SEED initiatives as described in Impact and as executed in WP8. This will be led by Innova, who have a track record for the valorisation of R&D results. A mitigating aspect of not achieving the high adoption rates lies in the design of the Work Plan: the core services of NIMBLE will remain available as permissive open source for uptake by developer communities and other projects. Salzburg Research as coordinator of the successful IKS project has experience with mobilising stakeholder communities in such a project.
Results that cannot be adopted by small and micro SMEs easily	Probability: Low	Impact: Medium	We dedicate resources in WP2 and WP3 to design and develop intuitive user interfaces in addition to simplified programming level interfaces targeting ICT providers. We also have a separate work package (WP4) for collecting feedbacks from end users about the core platform features

Probability: Low Impact: Medium			that would enable us to revise them addressing SMEs' requirements.
Failure in managing heterogeneous data produced in use cases	Probability: Low	Impact: Medium	In order to deal with the variety, size and velocity of data, we will use proven solutions provided by open source communities and project partners: Metadata registry used in e-health and security domains for data modelling; open source established solutions for streaming, persistence, analytics; lifecycle data management framework already used in various industry sectors such as food, and machinery.
Use case requirements are not well understood and platform features do not meet the actual requirements of end users	Probability: Low	Impact: Medium	In addition to requirement specification (WP1) and user interface design (T2.5), NIMBLE technology providers will have face-to-face meetings with end users in order to eliminate misunderstandings on the needs of end users. Furthermore, WP4 will give the chance to end users to provide their feedback guiding the platform development in line with their needs.
Inadequate level of trust and security	Probability: Low	Impact: High	NIMBLE will follow the security by design principle in order to ensure the technical security of core platform components and services provided through the platform. NIMBLE will also ensure trust of platform participants by employing fair gain distribution and reputation management measures. In particular, the holistic approach including information quality has high innovation potential and the partners involved have the necessary experience to create a solution that goes beyond the state of the art.
Failure in measuring and achieving the expected impacts	Probability: Medium	Impact: High	The evaluation strategy introduced in the Methodology Section will be elaborated in WP1 by definition of qualified and quantified KPIs on business and operational levels. The product lifecycle data management and lifecycle estimation tools will enable us to collect data required for calculating the KPI and measure them accurately.
Delays on the time of task deliverables that are prerequisites of other tasks	Probability: Medium	Impact: Medium	NIMBLE partners are involved across different WPs and effective knowledge transfer will be established in order to resolve dependencies and integrate WP results. Furthermore, NIMBLE will follow iterative prototype cycles to manage these dependencies efficiently and a detailed work plan with Deliverables and Milestones allowing for frequent monitoring of progress. Strict govern-

			ance structure and strict procedures are to be agreed upon and documented in the Consortium Agreement based on MCARD-2020.
A key technology partner leaves consortium	Probability: Low	Impact: Medium	Even if a partner leaves who has critical responsibilities in the development process, the NIMBLE platform will be designed in such a way that any component of the platform will be provided by new partners to be found. The special case of IBM is discussed separately, at the top of the risk list.
Conflicts in the consortium	Probability: Medium	Impact: Low	A solid Consortium Agreement is established with well-defined roles and tasks, use of MCARD-2020 Consortium Agreement to provide clarity.

All risks are continuously monitored by the project co-ordinator.

3 Summary of QA Procedures

3.1 QA for Deliverables

We develop all deliverables initially online, using the Confluence collaboration platform established by the co-ordinator.

We expect a first content description typically three months after the relevant task has started. Six weeks before the deadline, a first draft of the deliverable must be issued to the consortium. Named QA reviewers have to comment on that draft. Three weeks before the deadline, an improved close-to-final draft has to be issued. The final draft must reach the coordinator one week before the deadline. The coordinator performs a final QA and does final edits if necessary and then submits the deliverable as set out in the deliverable plan.

(1) First draft released to consortium for comment	6 weeks before deadline
(2) Close-to-final draft to named QA group (min. two)	3 weeks before deadline
(3) Final draft to coordinator for final QA and submission	1 week before deadline

The initial document structure should be developed on the project's Collaboration Platform, together with relevant background material that can also be uploaded (see section 4 of this document for a description of the platform).

The **first draft** (1) can still be a collaborative document on the platform or a Google doc or a Word document. Approximately two-thirds of the final text should already be in place, maximum one-third of the sections may still be bullet points still to be developed into a coherent narrative.

The **close-to-final draft** (2) must contain the full text, it may still lack some references and it need not be copy-edited yet. The internal QA reviewers must be able to act as independent peer reviewers on the basis of this draft. The review should be done within one working week If the QA review suggests that significant re-work is still required then at this stage the project board may decide to request a delay of the deliverable by one month. A recommendation to do so should come from the QA review panel of the deliverable and has to be reported to the PO for approval.

The **Final Draft** (3) of each deliverable must be sent to the co-ordinator two weeks before the deadline. The co-ordinator is always responsible for a final QA and for upload (submission) to the Participant Portal. If the co-ordinator is not satisfied with the quality of the deliverable at this stage, the project board will convene to decide whether the deliverable should be submitted in its present state or whether a one-month delay should be reported in order to improve the deliverable.

The template for QA review is in the appendix of this QA guide and it is also available online on the Project Collaboration Platform:

https://secure.salzburgresearch.at/wiki/display/NIMBLE/NIMBLE+QA+Guidelines

3.2 QA for Research Results / Papers

The reality of research life is that deadlines for papers are tight and as a result, lengthy internal approval procedures either stifle the production of research papers or get ignored. Therefore, we keep our procedures light:

a) Researchers are requested to announce plans to write papers as early as possible
b) Papers should be uploaded to the collaboration platform before submission, for information and to give industrial partners an opportunity to check w.r.t. IPR protection
c) Researchers are encouraged to produce papers jointly with others
d) Participation of industry partners is particularly encouraged
e) When specific industry related topics are discussed, relevant industrial partners should be involved in the description, to ensure IP protection as well as correctness
f) Papers need to include standard acknowledgement of EC H2020 funding and a reference to the NIMBLE project
g) For all research achievements there should be adequate documentation in the public domain. The issuing of open access technical reports and white papers is encouraged for this purpose.
h) All technical publications should carry a note describing the kind of reviewing which the publication has undergone.
i) A disclaimer of no warranties for third parties following any of the described procedures is mandatory.

3.3 QA for Software

The software quality assurance process has to cover requirements definition, software design, source code management, configuration management, testing and release management. Additionally, since part of the development in NIMBLE is research-oriented, we need to develop a good understanding of the maturity of modules that are being assembled to make up the NIMBLE platform. This will be achieved by emphasizing the importance of design and code reviews.

3.3.1 Requirements Definition

Research projects tend to develop an imbalance between requirements and actual running software. The problem can be described as follows: research partners have a bias towards recognising requirements that are closely related to their research focus, and tend to have blind spots for issues lying outside their research focus. Industrial user partners sometimes project their wish list of silver bullets into the research prototype to be developed. Industrial software partners tend to either solve the new problems with existing tools or are trying to develop new products with modest regard for the core needs of the project.

The net effect of these tendencies is often an excessively large requirements document that bears little resemblance to the actual designs and the actual working software. We regard this as a crucial difference between standard software development and software development in a large-scale research project.

Therefore, our focus is on maintaining at all times, a good balance between requirements gathered, designs created and actual working code.

The clearest path towards meaningful requirements is to firstly, meet up with the owners of the use case scenarios and see them at work, in their respective environments. Workshops with all use case scenario owners are planned, involving also technical partners.

The challenge then is to derive horizontal (valid for all or many user communities) and vertical requirements (valid for specific user scenarios, but not necessarily for others).

NIMBLE faces an additional challenge because it also needs to bear in mind the requirements that come from managing a large B2B collaboration platform, which is a business in its own right. We identified this as an important driver for new requirements and will contact existing providers of B2B platforms for collaboration, even to offer NIMBLE technology for inclusion into existing platforms ("Nimble inside" approach).

The requirements space is thus bounded by the expectations of stakeholder communities:

- project reviewers on behalf of the European Commission with regard to the proposed action
- use case owners and their supply chain partners
- third-party software providers hoping to benefit from the NIMBLE eco-system
- early adopter businesses subscribing to a NIMBLE platform
- providers of a NIMBLE platform running the platform as a business for profit
- providers of a NIMBLE platform offering the platform as a regional infrastructure
- providers of other B2B platforms wishing to adopt relevant NIMBLE technology

Our QA approach towards this space is to have an explicit attribution of requirements to stakeholder groups.

3.3.2 Software Design

Our software design methodology uses top-down and bottom-up techniques to achieve the previously mentioned balance between requirements, designs and running software.

The technical group in NIMBLE frames the overall problem space of developing a large-scale business collaboration platform. This is driven top-down, by the designs presented in the project proposal and in the DoA.

It is still debatable whether software engineering is already at the level of other industrial engineering disciplines, in terms of reproducible quality through well-defined methods. It is even more debatable whether research consortia can achieve such levels of reproducible quality despite limiting factors such as temporary collaboration for the duration of funding, the distributed and heterogeneous setup of partners, unclear requirements due to the research character of the action, etc.

However, it is clear that any unstructured work environment can profit from the introduction of structure and here are our measures:

- The design process has started top-down, by framing the problem space of the B2B platform. This process is driven by the technical partners.
- This top-down process is met by a bottom-up process gathering the specific requirements of each of the use cases, which is driven by the main problem owners.
- There is a "middle-out" process addressing the generalizable aspects of user requirements, in particular the issues of collaboration in supply chains and of business models and how they can be supported and adapted. This process is managed by Lulea University (LTU).
- A specific issue concerns the use of advanced technologies ranging from explicit semantic models to AI based, unsupervised learning methods and the potential of block chain technology. For these, our approach is to design for functional modularity so

that business functions can be supported either by an advanced technology or by more traditional means. In the interest of wide acceptance, we need to keep user exposure to technological risks at a minimum – this is a balancing act for a research project. We support it managerially, by separating "core business services" (WP3) from "advanced services" (WP5).

- We are holding regular virtual meetings for WP2 (design) and WP3 (development) to discuss the current design and its relationship with requirements coming from WP1.
- Additionally, we use mock-up techniques to present designs early, to the potential users and to prompt them for feedback.

3.3.3 Source code management

The source code of the core business services developed in NIMBLE is licensed under the Apache License 2.0, a permissive open source license. We therefore selected github.com as our major source code repository. It supports all versioning control features of the *git*[2] source code management system including a web-based user interface and an issue tracking system. Due to the open source character it also allows easy integration with many other software development tools.

The "gitflow workflow"[3] will be used, which defines a strict branching model designed around the project release. Development and bug/issue fixing is done in separate branches. Merging into the 'master' branch is done via "pull requests", which must include a code review by at least one of the core developers, after all unit tests have been passed.

The public NIMBLE git repositories will be hosted under the address https://github.com/nimble-platform.

In addition, the git repositories provided by the Bluemix DevOps Services are available for the source code management and delivery of non-public code developed within the NIMBLE consortium.

3.3.4 Configuration management

Central configuration in distributed systems is necessary in order to lower the administrative overhead for configuring each individual service. Especially in the *microservices* approach a configuration management plays an important role, because services are loosely coupled with each other.

We are using Spring Cloud Config[4] as central configuration service. It reads settings from a single git repository and provides a RESTful interface for fetching individual configurations. Storing configurations files in a version control system (e.g. *git* or *svn*) comes with many benefits, such as version control, track of changes or code review functionalities. Each service fetches appropriate settings from the configuration service during start up and is being notified in case of changes.

[2] http://git-scm.com/

[3] https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

[4] https://cloud.spring.io/spring-cloud-config/

3.3.5 Documentation management

We will give special importance to the documentation in varying levels and modes as one of the major determinants of software quality, which would in turn lead to ease-of-use and high adoption of project results.

We will ensure clear categorization of the documentation so that users will be able to locate the most appropriate documentation for themselves. There will documentation for different target groups including industry firms desiring to benefit from the core and advanced services of NIMBLE; software developers desiring to develop their own modules using NIMBLE open source modules; and platform multipliers desiring to deploy a new instance of NIMBLE.

In addition to short and useful “Get Started” tutorials, we will prepare advanced tutorials with detailed information for performing specific tasks such as API consumption and platform multiplication.

At the finest level, we have documentation about the source code within the source code itself. Specific to the Java programming language, which is the main language we will be using in NIMBLE, we use Java Docs[5] as a self-explanatory documentation for individual source code modules. It is also possible to export this embedded documentation as a package that can be explored online. Furthermore, in order to ensure that the produced source code has a certain quality level, we will utilize continuous source code quality tools, like SonarQube[6], in conjunction with our continuous integration mechanism.

Software development is a living thing. In this respect, we will ensure soundness of documentation content by keeping it up-to-date and relevant in accordance with the latest advancements; and completeness of documentation by producing documentation for new software modules. Latest documentation will be publicly available on the NIMBLE web-site and source management platform, i.e. GitHub.

3.3.6 Testing and validation

The essential parts of software quality evaluation are the quality model, the method of evaluation, software measurement, and the supporting tools. To develop good software, quality requirements are specified, the software quality assurance process is planned, implemented and controlled, and both intermediate products and end products are evaluated.

NIMBLE software testing activities will use the Software Product Quality Evaluation Reference Model (SQuaRE) which describes the process, activities and tasks performed during the quality evaluation of a software product. This reference model is defined by the standard ISO/IEC 25040[7] that contains general requirements for specification and evaluation of software quality. As a practical outcome, it introduces a set of Evaluation Modules (EM) some of which we will use for categorizing our testing activities as follows:

- **EM1 – Unit tests:** These are standalone tests that verify if the components function(s) under test meet the requirements. For the evaluation, unit tests will be imple-

[5] http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html

[6] https://www.sonarqube.org/

[7] http://www.iso.org/iso/catalogue_detail.htm?csnumber=35766

mented by the developers along with the creation of the code. With these unit tests, we can ensure that all the different pathways of the components are exercised.

- **EM2 – Integration tests:** Integrations tests are intended to test Web Services or RESTful services of NIMBLE components through the HTTP protocol. These tests ensure that the NIMBLE microservices expose their functionalities to the outside world and process the incoming requests successfully. Furthermore, by developing such tests, we will be sure that NIMBLE components can communicate seamlessly through their HTTP based services.
- **EM3 – Fault tolerance analysis**: Constructing a dependable and fault-tolerant system is inherently difficult. Not only should the system work under normal circumstances, but must also continue operation and provide potentially degraded service under hostile circumstances. This evaluation module will evaluate the degree of fault tolerance of each NIMBLE components, by testing the system through erroneous input cases.
- **EM4 - Execution Time Measurement:** The time of execution that the components actions take can have a great impact on the user experience. So, this module evaluates if the components perform in a way that provide the functionalities fast enough to the end-user, and the user has the feeling to work fine with the tool without waiting time. For this, execution time of the selected critical use cases for our end users will be measured.
- **EM5 - User Interface Inspection:** This evaluation module has the purpose of assessing the usability of the user interface.

Our continuous integration mechanism will ensure that the first three evaluation modules will be executed automatically when an update is done on the NIMBLE source code. Logs about EM4, execution time measurement of services, will be collected via the Netflix Hystrix[8], which is a specific module gathering measurements about different metrics related to health condition of software services. As additional quality and performance indicators, we will keep track of parameters like failure rate of services, number of requests per second, etc. will be defined and monitored.

3.3.7 Release management

For initial development, there will be Production (release) / Master / development branches:

- Development branches are used to develop individual software modules;
- The master branch is used to integrate the individual updates with the rest of the source base and get the complete stack into a stable state;
- the production branch keeps the latest released and working version.

Third party libraries are checked for their licenses so that they would not violate the permissiveness of the source code.

Each release will refer to the list of implemented features; list of fixed bugs, resolved issues showing the updates as of the previous release; checksum (MD5) and a signature (ASC) to check the validity of the released files.

Each release will have a name indicating the released module(s) and uniquely identified with a version number.

[8] https://github.com/Netflix/Hystrix

Released source code will be published online and binaries of the source are also associated with the source code files.

Regarding the granularity of releases (releasing individual components vs. releasing the stack as a whole): We can choose an intermediate granularity level such that we would release the NIMBLE core as a whole, but software modules implementing advanced functionalities could be released separately.

Once the NIMBLE platform is running and in use, the managing of releases built with microservices begins to play a vital role, because single services are being released independently of each other. Despite these independent releases, we will enforce homogeneous release workflows for core business services in order to ensure consistent quality measurements.

New versions of single microservices have to pass a defined continuous delivery pipeline before being released, using the canary release technique: after deploying a new version only a small part of the overall traffic is routed to the new version (see Figure 1).

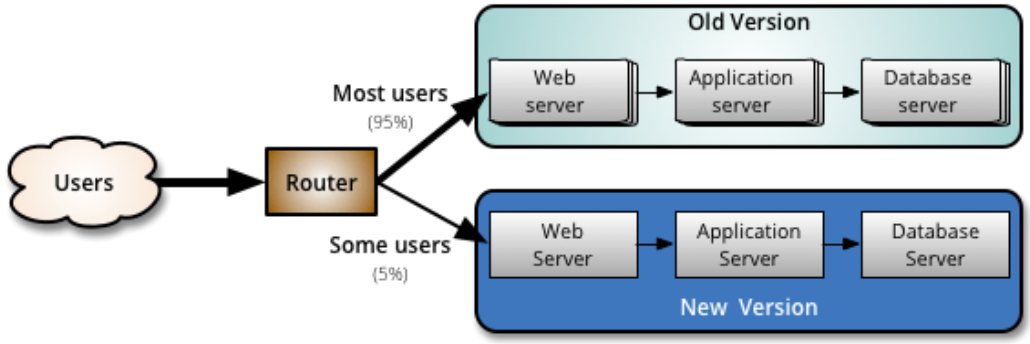

Figure 1: Canary releases for deploying new version of Microservices.
Source: https://martinfowler.com/bliki/CanaryRelease.html (Jan. 2017)

This technique allows stakeholders to rollback faulty releases without affecting the majority of users. The ratio of requests routed to the new version is continuously increased until it has reached 100%. Continuous integration in combination with canary releases allows developers to safely follow the release early, release often paradigm.

3.3.8 Issues management

For issue management, we will use the built-in capabilities of the GitHub platform. GitHub allows keeping track of issues opened both by the NIMBLE developers as well as external users outside the NIMBLE consortium. It is possible to assign issues to specific developers and annotate/filter them with standard labels like “bug”, “enhancement”; or with custom labels based on specific needs. Milestones can be defined to group issues with respect to module, feature, or time period.

GitHub also provides dashboard views in order to see the statistics about the issues including opened/closed “Pull requests”, opened/closed issues within a particular range; also overview pages where issues assigned to/created by a particular user can be seen.

3.3.9 Software Design and Development Methods

Usability and User Experience

In general a user-centered design shall be pursued. It is an iterative design process for products and systems centered on the users, their tasks and environment. When properly applied, it reduces the development time and costs, increases the quality of the product, avoids unnecessary or unwanted features, detects problems at an early stage and balances usability and user experience factors.

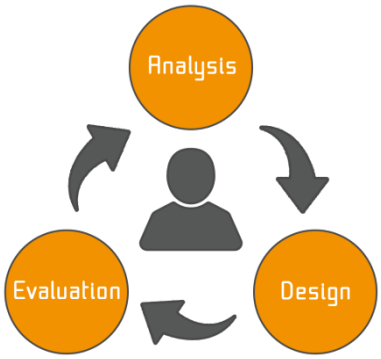

Figure 2 – The User-centered design cycle

The three-stage process consists of:

- Analysis

  Analysis is essential since expressing and communicating needs is difficult. Various analysis tools exist, such as information and literature research, task analysis, cultural probing, personas and scenarios.

- Design

  The design stage includes merging analysis results and defining what is needed for the implementation and how to measure the success. Therefore, conceptual models, storyboards and use cases are key elements. Continuous prototyping helps to create and evaluate requirements. Prototypes are categorized into low (no detail, quick overall feedback), medium (more detail, design feedback) and high fidelity (close to final product, usability feedback) depending on their degree of detail.

- Evaluation

  Evaluation is needed in order to rate previous decisions, identify usability and user experience issues and create new ideas. The two main evaluation categories are expert (experts rate and improve usability factors based on heuristics) and user tests (qualitative or quantitative data is derived from various evaluation methods; e.g.: interview, questionnaire, focus group, observation, benchmarking, thinking aloud). For user tests it is important to conduct pilot tests, create detailed test plans and record all data (e.g. by transcript, audio/video recording, software logging). The evaluation results serve as a basis for the next iteration. Depending on the severity of the detected issues, one may draft new prototypes, go back to the design process or even expand the analysis.

Relevant guidelines and frameworks that can be used in order to align with user interface and user experience design principles are listed below:

- ISO 9241
  A multi-part ISO standard covering ergonomics of human-computer interaction
- Human Interface Guidelines
  Interface guidelines for various major platforms
- Nielsen's Usability Heuristics
  General principles for interaction design to be used for heuristic evaluation
- Bootstrap
  HTML, CSS and JS framework for responsive web and mobile interfaces

Agile Project Management

In software projects requirements are constantly evolving over time. Therefore, it is necessary to have an agile project management method in place in order to continuously adapt to changing requirements. Scrum provides appropriate tools for agile project management in software projects and defines three roles with different responsibilities: Project Owners, Scrum Master and Developer Team. Project Owners (i.e. the product manager and use case partners) are aware of top level requirements and review each version periodically for any misalignments (periods are called Sprints). Improvements are communicated to the Scrum Master, which in turn maps changes to separate issues and prioritises them for the next Sprint. At the beginning of each period the Developer Team and the Scrum Master collaboratively plan issues to be solved in the upcoming Sprint, which results in a new version reviewed by Product Owners. This methodology achieves a good match between requirements and implemented functionalities of the software.

A major challenge in Nimble is the distributed setting of the project. Developers are spread across partners, which are located in different European countries. Special measures have to be applied in order to execute distributed Scrum (e.g. the length of each Sprint must be adapted and periodic meetings held remotely).

3.3.10 QA for Security

The NIMBLE QA plan includes the testing of the platform/applications/services security features, e.g. how easy is to hack the platform, to overload it, to block it by DoS attacks, etc. In NIMBLE, we will use the STRIDE threat modelling approach [1] to model scenarios to challenge the system against various attacks. These scenarios will be continuously in use for the validation of the NIMBLE platform security features. STRIDE stands for **S**poofing Identity, **T**ampering with Data, **R**epudiation, **I**nformation Disclosure, **D**enial of Service, Privilege **E**scalation (or Elevation of Privilege), and its methods provide defined sets of situations/strategies to identify each of these six major threats, plus the corresponding mitigation techniques. The threat mitigations (fixes) should be additionally tested and validated against the threats, too.

The identified bugs for fixing will be tracked in written form (e.g. bug ID, threat description, risk management technique, risk acceptance, tester, status of fixes, reasons of not having it fixed, etc.). The number of open defects, their severity level, etc. will help us to collect useful security metrics that will further guide the platform improvements. A vulnerability scoring system, such as CVSS (Common Vulnerability Scoring System) [2], CCSS (Common Configuration Scoring System) [3] or OVAL (Open Vulnerability and Assessment Language) [4] will be selected and used too, to i.e. communicate the characteristics and severity of software vulnerabilities.

In addition, the NIMBLE platform will be periodically tested against the defined policy compliance framework. The testing schedules will be established too.

[1] STRIDE: https://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx

[2] CVSS: https://www.first.org/cvss

[3] CCSS: csrc.nist.gov/publications/nistir/ir7502/nistir-7502_CCSS.pdf

[4] OVAL: http://oval.mitre.org/

4 Project Infrastructure

The project is co-ordinated by Salzburg Research. The company is certified according to ISO 9001 with the last re-certification in 2016.

4.1 Communication

Official communication happens via primarily via email (plus posted letters for legally binding documents) and the following email groups are available for all project staff:

Mailing list	Topic
nimble-work@salzburgresearch.at	For mails addressed to the full RTD team
nimble-wp1@salzburgresearch.at	Requirements
nimble-wp2@salzburgresearch.at	Specifications
nimble-wp3@salzburgresearch.at	Core Service Development
nimble-wp4@salzburgresearch.at	Experimentation and Initial Validation
nimble-wp5@salzburgresearch.at	Advanced Services
nimble-wp6@salzburgresearch.at	Trust & Security
nimble-wp7@salzburgresearch.at	Validation
nimble-wp8@salzburgresearch.at	Dissemination

We keep a directory of partners' project staff on the collaboration platform, also showing who is subscribed to which mailing list.

For conferencing, three systems are in use: Skype, Teamviewer supplied by the coordinator and IBM's teleconferencing suite.

4.2 Collaboration

Software development has already been described separately. For joint development of ideas, papers and deliverables, there are three approaches used:

- Google docs
- Word documents exchanged via email or dropbox
- The Confluence wiki-type collaboration platform

Of these, the Confluence platform is the official and central infrastructure provided by the co-ordinator. The platform is fully maintained and backed up daily. In addition, the co-ordinator keeps an internal project repository on a samba-share, but this is only available to Salzburg Research staff.

The collaboration platform is available to all registered staff of NIMBLE and has the following structure:

NIMBLE-Subspace	Used for
Consortium Meetings	Agenda, minutes, decisions, actions of con-

	sortium meetings, including attendance lists
Core Documents	Contract, Work plan, consortium agreement
Decision log	Executive decisions / agreements
Deliverables	Final deliverables as submitted
File lists	Overview page of uploaded files for NIMBLE
WP1 – WP8 subspaces	Further structured into tasks, each WP also has a meetings sub-space to record agreements and actions.
WP9 (Management)	This also contains the latest 6-monthly action plans for all work packages and tasks
Internal meetings (with subspaces for each partner)	We offer NIMBLE research groups the possibility to record internal meetings if they wish to share. This makes sense for larger teams, e.g. for the coordinator's team.

4.3 Development & Testing

In addition to the standard tools available on Github and the cloud operations-related tools of the Bluemix environment we will also make use of TeBES, the *Test Bed Environment for Standard based Interoperability.* This is a modular on-line test bed for testing system interoperability and standards conformance. Beyond specification conformance, the system is able to implement complex testing plans as well as interoperability testing across multiple step collaboration. In addition, another testing system, born from the standardised CEN GITB architecture, is fully configurable to implement complex testing plans described through standard based descriptors (TAML language and others). In NIMBLE, this tool (system) will be updated to be delivered as a set of open source services fuelled through testing plans automatically generated from business model patterns to facilitate collaboration setting up.

4.4 Administration

4.4.1 NIMBLE Office

The NIMBLE project office has a mail-alias nimble-office@salzburgresearch.at with 4 project staff receiving the mails, including the project administrator and the project coordinator.

nimble-office@salzburgresearch.at	ursula.atzlinger@salzburgresearch.at
	wernher.behrendt@salzburgresearch.at
	violeta.damjanovic-behrendt@salzburgresearch.at
	georg.guentner@salzburgresearch.at

There are two more mailing lists for administrative purposes:

nimble-admin@salzburgresearch.at	For communication among EU-project administrators
nimble-legal@salzburgresearch.at	For communication concerning legal issues (was used e.g. during negotiation of the consortium agreement).

4.4.2 Financial Management and Resource Controlling

A payment schedule has been devised in order to ensure timely payment of grant monies in line with project achievements.

For controlling purposes, the co-ordinator uses a spread sheet to monitor reported efforts vs. planned efforts on a per-partner, per-task basis. Reporting by partners is done quarterly to ensure timely interventions if needed.

5 Appendix – QA Template for Reviews of Deliverables

Review of NIMBLE Deliverable Dx.y <Title of Deliverable> **Reviewer**: <Name>, <Affiliation>

QA Main feature	*QA Sub-feature*	*excel cel-lent*	*good*	*weak ness*	*very poor*
DoW Compliance					
	Relevance of the work to project goals				X
	Impact of results on project progress				X
Research Quality					
	Methodology used in the research				X
	Quality of claimed results				X
Quality of Presen-tation					
	Readability, e.g. appropriate wordings				X
	Understandability, e.g. logic of the presentation, ordering, structure				X
	Use of illustrations, e.g. diagrams, ta-bles, figures etc.				X

Overall Recommendation (one of the following outcomes is possible)
Deliverable can be submitted
Some small changes should be made, then the deliverable can be submitted
Significant changes need to be made, I will check the deliverable again

Overall assessment and main recommendation for the authors of the deliverable
<Short description of main review points …>

Date of Submission for Review dd/mm/yyyy: Date of Review: dd/mm/yyyy

Specific comments to the authors / researchers responsible for the deliverable
This section should be as detailed as necessary to guide the authors to address important issues
<Reviewer's signature>