Collaborative Network for Industry, Manufacturing, Business and Logistics in Europe

D3.8

Tool for Collaboration Setup and Interoperability Testing

Project Acronym	NIMBLE
Project Title	Collaboration Network for Industry, Manufacturing, Business and Logistics in Europe
Project Number	723810
Work Package	WP3 Core Business Services for the NIMBLE Platform
Lead Beneficiary	ENEA
Editor	Gianluca D'Agosta, Nicola Gessa, Piero De Sabbata ENEA
Reviewers	Benjamin Mandler, Yildiray Kabak
Contributors	
Dissemination Level	PU
Contractual Delivery Date	
Actual Delivery Date	31/12/2018
Version	V2.0

Abstract

This document explains how NIMBLE can support firms in adopting open XML standards for business information exchange by offering a digital business document test service.

Since the '70s, the exchange of business information using reliable standard solutions, such as EDI, has been one of the most relevant elements for creating complex networks of enterprises interacting along the same supply chain. Unfortunately it was a heavy mechanism that very few supply chains were able to adopt, for example the automotive industry.
Since the end of the last century, the evolution of other simpler languages, such as the eXtended Markup Language (XML), has boosted the flowering of new applicative standards supporting several aspects of business processes such as OASIS UBL and CEN eBIZ/Moda-ML.
However, the adoption of such standards still requires effort and the assumption of costs that become a barrier for many small firms. It is experienced that the initial collaboration setup and debugging phases are the most risky and costly for companies.

Large firms, but not only, are increasingly asking for business documents to be automatically managed in system-to-system scenarios.
Because the exchange of business documents based on standards is one of the ways to support customised processes, this problem will affect also the NIMBLE platform, in which large and small companies coexist and collaborate.
The Test Bed Platform (TeBES), described in the following chapters, aims at contributing to make NIMBLE easier and more effective for system-to-system collaborations by reducing the risk of semantic misalignment of complex data formats and shortening the time to setup digital collaborations.

This document presents how the collaboration among NIMBLE and the TeBES test bed platform has been implemented for supporting firms.

The following subjects are analysed in detail:

- the software architecture and logic of TeBES;
- how TeBES is embedded in the NIMBLE platform;
- the ontology-based tools to automatically generate test plans and rules tailored for the customised processes adopted by the firms in NIMBLE. This is very important for the sustainability of the services because it offers a good trade-off between the cost of an effective and accurate testing strategy and the need for customisation of the collaboration processes.

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, Industry 4.0, Internet-of-Things-enabled B2B platform on which European manufacturing firms can register, publish machine-readable catalogues 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 42 months.

Document History

Version	Date	Comments
V0.1	19/04/2018	Initial structure
V0.5	20/06/2018	Draft release
V0.6	21/06/2018	Updated draft release
V0.7	22/06/2018	Update and modifications
V0.8	26/06/2018	Revisions and updates
V0.9	27/06/2018	Revisions and abstract preparation
V1.0	28/06/2018	Final revision and release
V1.1	10/12/2018	Revision after reviewers' comments
V1.2	20/12/2018	Revision after second peer review
V2.0	31/12/2018	Final edits and Submission

Table of Contents

List of Figures

List of Tables

Acronyms

Table 1: Acronyms table

Acronym	**Meaning**
ABIE	Aggregate business information entity
API	Application Programming Interface
B2B	Business to business
BBIE	Basic business information entity
CRUD	Create – update – delete
CSV	Comma separated value
ebXML	Electronic Business using eXtensible Markup Language
EDI	Electronic data interchange
GUI	Graphical User Interface
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
JSON	Java Script Object Notation
LDPath	Linked data path
NIMBLE	Collaboration Network for Industry, Manufacturing, Business and Logistics in Europe
PEPPOL	Pan-European Public Procurement Online
RDF	Resource description framework
REST	Representational State Transfer
SDK	Software Development Kit
SPARQL	SPARQL Protocol And RDF Query Language
UBL	Universal Business Language
XML	Extensible Markup Language
TeBES	Test Bed Platform

1 Introduction

The ultimate aim of NIMBLE is to optimize companies' B2B operations throughout the supply chain by orchestrating and automating data exchange among the participants of the supply chain in different phases, such as sub-contracted manufacturing, transportation, etc. The orchestration of almost automatic data exchanges is realized through business processes involving at least two parties with specific roles feeding the process with relevant data.

The implementation of these business processes using NIMBLE implies the adoption of shared and agreed elements that can hold the structure of the data exchanged among the participants or actors. In many cases, B2B operations are controlled by software applications that manage the collaborations, extract data from databases or generate documents starting from the internal representation of the information. NIMBLE allows the application managers to express, describe and implement the business processes agreed with other companies speeding up the collaboration setting-up process, removing some obstacles to the electronic B2B and "greasing the wheels" that move the collaborations.

Several markets are requiring the companies to increase their dynamicity and to create new short-term collaborations with other industries: NIMBLE wants to ease this process by providing all the tools necessary for such a type of collaboration.

Among these tools, the availability of a set of templates for business documents can contribute in reducing the effort necessary for the creation of the information link among actors[1]: the more these documents are well-known, shared and accepted, the faster can be their adoption and less are the errors due to misunderstandings of the semantics of the data.

For this reason, NIMBLE has chosen to use the Universal Business Language and eBIZ/Moda-ML as "suggested" sets of business documents for the collaborations, leaving in any case the company the possibility of using specific, non-standard or personalized documents.

The adoption of a standard requires, also, that NIMBLE checks the format of the received and produced documents in order to avoid mistakes (that arise very often).

UBL is a very general and universal language, taking into consideration almost all the possible business cases and processes involving two or more actors exchanging business documents. For this reason, it may be a bit cumbersome and thus, in some cases, a document instance cannot be correctly interpreted by automatic systems even if they respect the standard.

Let us look at a simple example where an English company orders a set of shoes from an Italian company selecting different sizes: UBL template for Orders contains information related to size and, normally, it is a number with a single decimal. But in the UK the size of shoes for men has a value, normally, between 7 to 14, while in Italy it starts from 34. Machines can misinterpret this information when processing the order and they could start the production of shoes for children or block the order as wrong.

[1] E.g. In 2004, during the Leapfrog IP project (FP6 - NMP), the business document exchange between a Textile Controller, a Textile Producer and a Garnment producer lasted for several weeks due to the difficulties in understanding the information contained in different documents by the partner involved.

"Interoperability is the ability of two or more systems or components to exchange information and to use the information that has been exchanged".[2] Proving the interoperability among NIMBLE and user's system is the objective of the tools developed in this NIMBLE task.

For this reason, two different types of testing have been implemented:

- **the syntactic test** (or conformance test), that proves the conformance of the document against a precise structure of a template;
- **the semantic test**, that analyses whether the document can be correctly interpreted by a machine.

The combination of these two different types of testing can guarantee a high degree of interoperability among the systems that pass these tests.

These tests can be run using a **test bed** platform: a software tool that simplifies the execution of the tests by supporting the user with interfaces.

This document presents the software solutions implemented into the NIMBLE platform to allow the automatic creation of syntactic and semantic tests thanks to the introduction of "*rules*" that documents have to respect.

Two types of tools have been developed to support the NIMBLE dynamic scenario for B2B implementation:

- an advanced version of the TeBES test bed, enabling automatic testing without human intervention;
- a test generator that extracts information from NIMBLE ontologies and creates the necessary inputs for the test bed platform.

This document presents, after an introduction of the problems (chapters 2 and 3), the analysis of the NIMBLE requirements that can be satisfied by the adoption of a test bed platform (chapters 4 and 5), the description of the TeBES platform (chapter 6) and the activities done to implement a testing system in NIMBLE (chapter 7 and 8) that can reduce the risk of introducing errors originated by misinterpretations of business documents. A short conclusion section (chapter 9) summarizes the results of the completed activities.

[2] *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries (New York, NY: 1990).*

2 Standard adoption in business networks

Real time information exchange among heterogeneous and geographically distributed systems is required to support the execution of complex e-business scenarios as those of NIMBLE. Achieving seamless interoperability among heterogeneous communication systems and technologies remains a great challenge in particular for the industrial world.

XML emerged as a foundation for performing e-business with an increasing adoption in the new market economies. Based on XML, several interoperability standards emerged, which provide specifications on performing business-to-business e-business, what information to share, when and how.

Moving from the previous **EDI** (**Electronic data interchange**) general approach (like EANCOM or UN/EDIFACT or ANSI X12 specification) in the direction of the new XML based standards, different actions have been originated by different organizations and companies[3]. Even if the approach of EDI is correct and widely used, its use is cumbersome, extremely rigid, and the documents produced are complicated and not human-readable.

The **ebXML initiative** led by OASIS and UN/CEFACT provides an open, XML-based infrastructure that enables the global use of electronic business information in an interoperable, secure, and consistent manner by all trading partners. ebXML approaches all the collaborative layers, from the data channel definition to business processes description and contractual agreement on these transactions. It does not define the models for business documents.

In parallel with the development of the e-business framework, OASIS launched the **UBL** (**Universal Business Language**) initiative: an open library of standard electronic XML business documents for procurement and transportation such as purchase orders, and invoices of logistic documents. UBL defines two main sets of elements required for e-business implementation:

- the list of described and supported business processes;
- the list of standard document types.

The latest version of the UBL standard, 2.1, contains more than 65 different templates of documents and describes more than 15 different processes. Templates are written in XSD Schema format and can be used to check the conformance of a document to UBL specifications.

Unfortunately, the generalising approach of UBL, stemming from the complexity of the e-business description, has a negative impact on the efforts required to adopt this standard and to use the information contained in a business document. Actually, documents have to cover almost all the types of data structures available in business, leaving to the user the possibility of choosing which one fits better his requirements but, on the other side, requiring a detailed agreement on the shared data structure. A typical and simple example of this problem is related to the creation of the data structure that implements an international address: the variety of available formats ranges from a simple string containing the whole address to a set including a dozen of different elements, mainly strings.

[3] A partial list of business-related initiatives based on XML can be found at: https://schemas.liquid-technologies.com/Category/Business

Looking at a typical UBL template two main aspects arise that may create confusion and misunderstandings:

- high level of **redundancy**: elements are replicated in different positions in the document in order to include all possible cases into a template;
- elements are **recursive**: an element may contain a replica of its own structure or one of its ancestors.

These aspects reduce the readability of an UBL template both for humans and for applications that do not know where to find the correct version of data.

For this reason, different organizations, sustained also by OASIS, have created a lot of **UBL subsets or customizations,** reducing the number of elements and solving possible ambiguities on the meaning of the elements. A single subset is identified as "**Use Profile of UBL**" defined for a specific business context.

These customizations introduce a further level of templates that can be applied in specific contexts.

Among these, two customizations, adopted in real business scenarios, are more interesting for the NIMBLE project:

- **PEPPOL**[4] the Pan-European Public Procurement Online project includes UBL customizations and syntax serializations of different document types for European business processes. The main goal of this project is to support the Public Procurement across borders in EU;
- **eBIZ-TCF**[5] a UBL customization available for the Textile Clothing and Footwear industry. It includes the Moda-ML standard and uses a customised version of UBL for covering the last part of the business chain. It has been adopted by partners involved in the Textile Pilot.

Experience made in UBL suggested that a document can hardly be adopted AS-IS but needs some adaptations and customisations by the users: these adaptations are formalised into a template that, while conformant to the UBL rules, makes the document more readable by software tools. The set of resulting templates, covering the different processes peculiar for a supply chain, might be defined as a **USE PROFILE of UBL** for the specific case and has to be shared by the actors involved in the business.

This subsetting approach has also been adopted in the NIMBLE Project. After specifying the business processes in the NIMBLE use cases, relevant UBL documents have been identified and they are customized according to the requirements of these use cases. As of writing of this deliverable the following business processes and documents are potentially relevant for NIMBLE (it should be noted that the number of business processes and business documents may increase upon request from end user companies of NIMBLE):

a) Business processes potentially relevant (those in bold are actually under implementation in NIMBLE):

[4] https://www.peppolbasics.info/peppolineurope/

[5] http://ebiz-tcf.eu/success-cases/

1) Billing
2) **CPFR (Collaborative planning, forecasting, and replenishment)**
3) Fulfilment
4) **Ordering**
5) Payment
6) **Production**
7) Sourcing
8) **Transportation**
9) Information request
10) Negotiation

b) Documents identified as potentially relevant:

1. **CatalogueDocument**
2. CatalogueRequestDocument
3. **CertificateOfOriginDocument**
4. **DespatchAdviceDocument**
5. ForecastDocument
6. InformationRequestDocument
7. InventoryReportDocument
8. InvoiceDocument
9. **OrderChangeDocument**
10. **OrderDocument**
11. **OrderResponseDocument**
12. **OrderStatusDocument**
13. ProductionMonitoringDocument
14. ProductModelDocument
15. QualityReportDocument
16. **QuotationChangeDocument**
17. **QuotationDocument**
18. **QuotationResponseDocument**
19. QuotationRequestDocument
20. QuotationStatusDocument
21. ReceiptAdviceDocument
22. StockAvailabilityReportDocument

23. **TransportationStatusDocument**
24. **TransportationStatusRequestDocument**

During the experimentation phase, the Ordering process and the OrderDocument have been used.

3 NIMBLE requirements for test bed application

NIMBLE has defined several consolidated non-functional requirements (D4.5 – NIMBLE Platform Evolvement – Recommendations, Requirements and Roadmap) that have to be satisfied by the NIMBLE software platform and its components, to support all the possible uses of the platform itself. The list of the consolidated requirements regards many different aspects and part of these requires the adoption in NIMBLE of a test platform, such as TeBES, or can strongly benefit from this adoption to fully implement them.

Table 2 contains the elements excerpted from the document D4.5 and contains those requirements that imply the adoption of a test bed platform to be fulfilled.

Table 2: List of relevant consolidated requirements

Platform reqs. (DoA)	**DESCRIPTION**
DoA-PL-02	*The regional or sectoral platform instance is capable of interoperating with other platforms in the **federation, via semantic interoperability services**.*
DoA-PL-03	*Specialisations would be necessary to account for **sector specific practices and standards**…*
DoA-PL-08	*NIMBLE objective: To master the usage of the platform step-by-step to **evolve business cooperation***
DoA-PL-19	*Services for **matchmaking** between producers and consumers are available to establish business collaboration in a faster way*
DoA-PL-26	*Improve business integration between different sectors*
DoA-PL-32	***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*

The following section presents the links among the requirements in the above table and the services offered by TeBES.

As shown below, the interaction among NIMBLE and TeBES requires the definition of a testplan necessary to perform the check.

DoA-PL-02 - *The regional or sectorial platform instance is capable of interoperating with other platforms in the* ***federation, via semantic interoperability services***

The implementation of a new instance of the NIMBLE platform must respect several rules with the objective of not to compromise the functionality of the other instances in the federation by the introduction of incorrect information. This risk can be mitigated by the adoption of a set of testing tools that check the conformance of the new instance to the strict requirements of the federation. TeBES, as a test bed platform, can implement these rules into test plans and perform detailed analysis on the way the new platform instance collaborates with the others.

DoA-PL-03 *- Specialisations would be necessary to account for* ***sector specific practices and standards.***

The implementation of a new NIMBLE instance for a new sector has to take into consideration the specificities of the business context in order to keep as low as possible the initial cost that the use of the platform requires. Several industrial contexts, such as Textile, have adopted practices and standards to ease the collaboration among different actors in the supply chain. In this scenario, the TeBES platform can be used both to refine and better configure the NIMBLE instance for the correct adoption of those standards and to support new customers in the process of adopting the standard for their business.

DoA-PL-08 *- NIMBLE objective: To master the usage of the platform step-by-step to* ***evolve business cooperation***

Evolution of the business collaboration implies also the possibility of adopting shared document formats for business data exchange. Although the adoption of standards in a collaboration requires considerable efforts by the participants, the TeBES platform can be used to support these companies to gradually improve their familiarity with the characteristics of the standard data format, to learn the correct usage of it in a protected environment, strongly reducing the cost and the risk of this process.

DoA-PL-19 *- Services for* ***matchmaking*** *between producers and consumers are available to establish business collaboration in a faster way*

The matchmaking process defines also the mutual agreement on how the collaboration can behave, also taking into consideration the fact that the business partners can be conformant to a sectorial standard. The TeBES platform can support the matchmaking process and verify/guarantee to all the parties about this conformance.

DoA-PL-26 *- Improve business integration between different sectors*

The adoption of public or common format for business documents (as UBL, for example) eases the process of creating a strong collaboration on business, by allowing all the participants on sharing information in a secure way reducing the misunderstandings (semantic interoperability). A UBL document, for example an order or a catalogue, has a structure with a well-defined semantics of the elements in the template: every company can read it and can correctly interpret data contained in a real business instance of this template. TeBES can both reduce the presence of mistakes in document instances and support companies in the process of adopting the standard, enforcing the integration between different sectors or firms that implement a common format.

DoA-PL-32 *-* ***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*

This requirement implies that new versions of the NIMBLE platform, and consequently the instances that use them, are strictly interoperable with previous version and do not require the development of new tools from the companies or, even worse, break the active business of a company. TeBES can test the interoperability of different versions of the NIMBLE platform reducing this risk as much as possible.

3.1 Flow of actions between NIMBLE and TeBES

The use of TeBES, as detailed below, requires the identification of a testplan that acts as a configuration document for it. The testplan can be selected from a list of already available or uploaded through the web interface by a human user. For this reason, TeBES provides the human user with a list of available testplan: the human user can define a subset of the relevant testplan(s) and use this information to configure the NIMBLE platform.

After this configuration process is complete, the interaction between NIMBLE and TeBES does not require direct action of human users anymore unless some modification in the configuration is required.

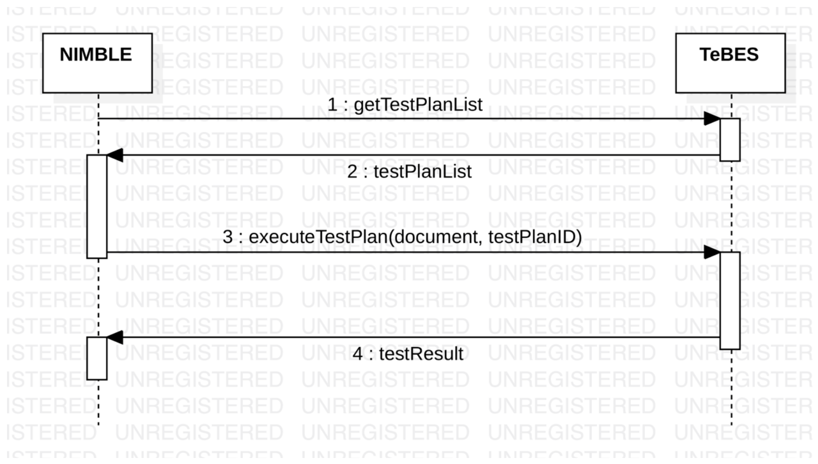


Figure **1**: NIMBLE - TeBES Sequence Diagram

Figure **1** presents the flow of actions among NIMBLE and TeBES: the interaction is quite simple because after the initial phase of configuration (steps 1 and 2) the successive collaboration can be reduced to a single REST call to a remote service that returns the result of the validation. The information on what to test and how to perform the test is all inserted into the testplans and, thus, completely transparent for the NIMBLE platform.

Looking at NIMBLE document "D2.1 – Platform architecture specifications and component design", it is possible to identify which components can interact directly with the TeBES platform to have information about the result of a specific test on business documents. In general, all those components that can use data from external sources or that can provide information to external users can have the necessity of testing the quality of the information received or produced. In particular, the "Data Sharing Service", the "Business Process Service" and the "Catalogue Service" are those which this collaboration is more relevant.

Below, **Figure** 2 shows the case in which NIMBLE receives a document or a set of data from an external module or source. Before using this data, NIMBLE can require a quality check to avoid issues on the successive interpretation and use of these data inside the platform (e.g. in case of incomplete or inconsistent data). Because the interaction with the external module has been modelled and configurated before a real collaboration happens, in the NIMBLE configuration is already present the information of which testplan TeBES has to use to test these data: by this way the test can be completed without the human intervention.

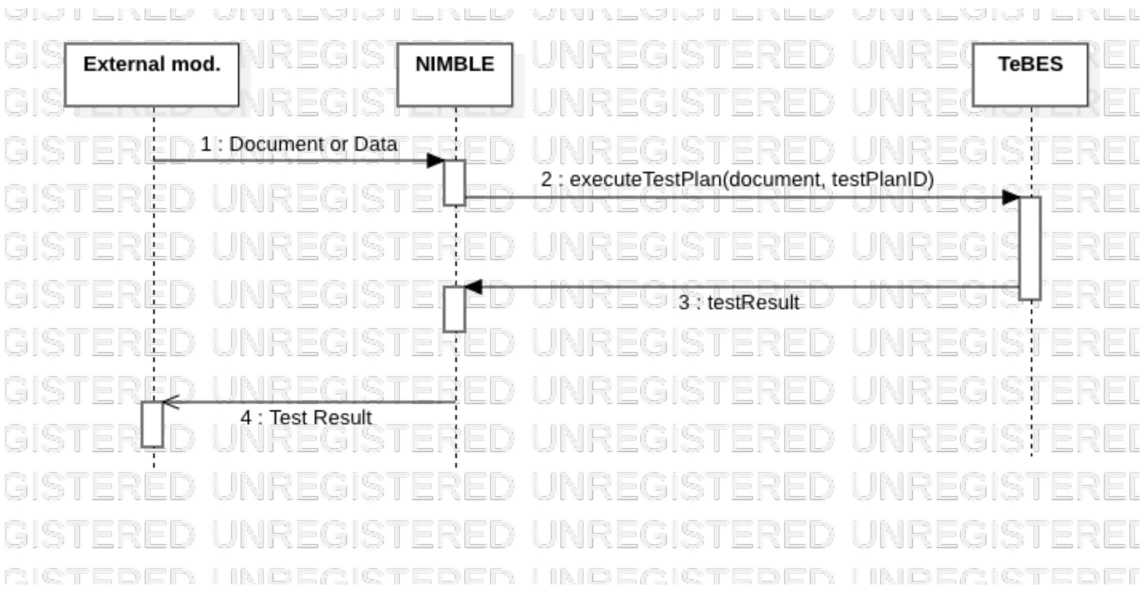


Figure **2**: NIMBLE receives a document from an external source and requires tests

For example, in the case of the Catalogue Ingestion Process, several check can be performed:

- Existence of catalogue ids
- Existence of catalogue line ids
- Existence of manufacturer id in items
- Equality of line id and manufacturer item id
- Existence of product names
- Existence of at least one commodity classification inside the items
- Equality of catalogue provider party and item manufacturer ids of items included in the catalogue lines
- Equality of catalogue document references ids with catalogue id

The symmetric situation is depicted in Figure **3**: NIMBLE has to produce a document for an external module (e.g. an XML version of an order to be imported into company's systems) and, to guarantee the quality of the data perform a test on the document before it is provided to the company. Only if the quality check is positive, the document is automatically provided to the external module, otherwise the NIMBLE manager is informed about the situation. As for the case before, the collaboration with such an external module has been modelled before the interaction happens, to the required testplan already identified in NIMBLE.

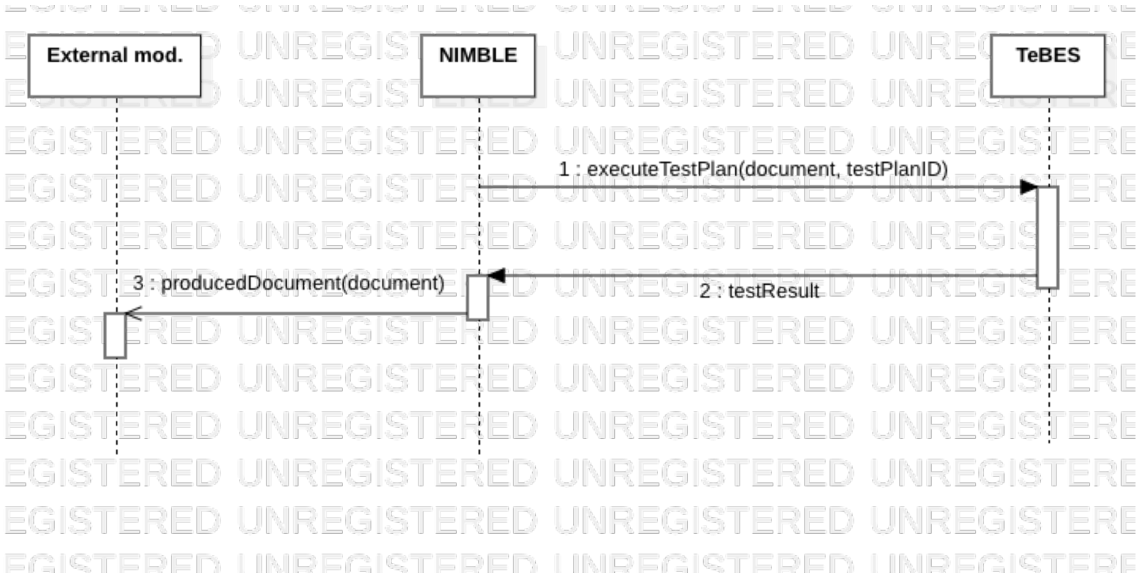

External mod.
NIMBLE
TeBES
1 : executeTestPlan(document, testPlanID)
2 : testResult
3 : producedDocument(document)

Figure **3**: NIMBLE produces a document used by external module

4 Test relevance for NIMBLE

NIMBLE as platform aims at reducing the “friction” for its clients when accessing B2B features: entering a new network is, for almost all the companies, risky and costly. The reduction of this friction eases the on-boarding process of new companies onto the platform.

Contributing factors for friction can be:

- substantial efforts required from personnel and resources that thus cannot be spent for the *core business* of the company;
- companies tend to buy services from external providers who want to sell their specific software solutions in order to strengthen the link with clients: this can create a lock-in situation that the clients can break by adopting standards.

Several actions can be taken to reduce this friction by simplifying the whole process of entering new business networks.

Among others, the adoption of a common standard for business documents, together with the clear definition of business process supported by the platform are two of the most relevant aspects from the point of view of those companies that already have an internal management system for production and administration.

Figure 4 below represents a common use case for NIMBLE: after registration, two companies interact with the platform to publish their products and to look for potential providers of products/services. After a positive match and the negotiation phase, the two companies can agree on the *use profile* for the common standard they want to adopt for this collaboration. By this step, they can receive data and documents from the other party, through NIMBLE platform, directly into their management systems, without other human activities and involvement that may create mistakes and introduce errors, as in the past.

The yellow circle identifies this action in the use case: lessons learned on connecting different companies suggest testing the different applications separately towards an external platform (as TeBES) before starting the interaction.

In fact, the largest part of the issues related to the process of translating data from the internal representation to an external format arises from the misunderstanding of the meaning of several descriptive elements of the new format: testing against a test bed platform highlights these errors and provides elements to solve them, suggesting the correct interpretation of the concepts expressed. In fact, a test bed platform may allow, when possible the **verification of the semantics** expressed in a document, and not only the correctness of the syntactic structure of it.

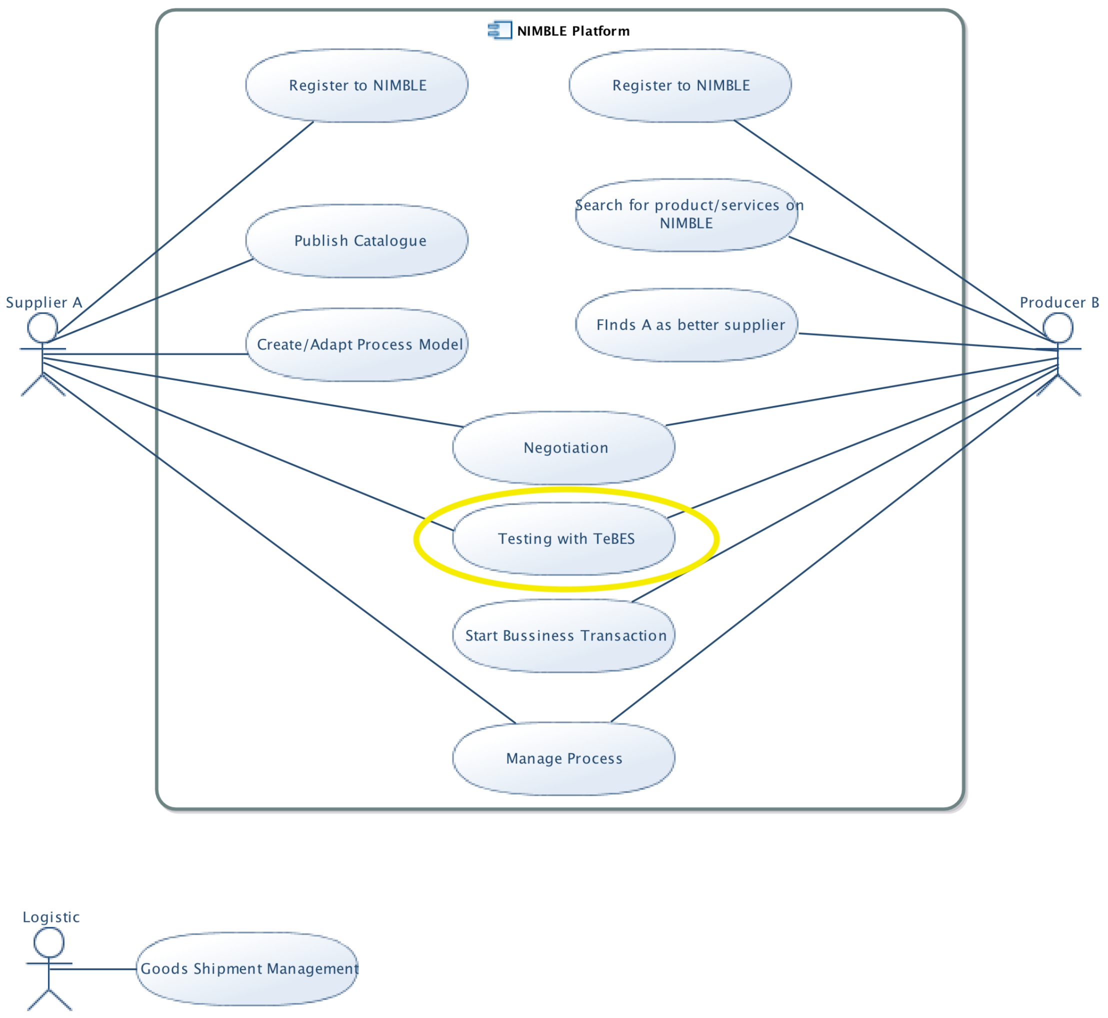

NIMBLE Platform
Register to NIMBLE
Register to NIMBLE
Publish Catalogue
Search for product/services on NIMBLE
Supplier A
Producer B
Create/Adapt Process Model
FInds A as better supplier
Negotiation
Testing with TeBES
Start Bussiness Transaction
Manage Process
Logistic
Goods Shipment Management

Figure 4: Simple NIMBLE use case for a business collaboration

NIMBLE uses the business process representation as a key element for enabling the business collaboration among companies and to reach a common objective that is the production of a good or provision of a service. For this reason, the test bed platform has to start from this item to build up the test process correctly: in any case, the objective of the test is to support the companies in correctly implementing the selected business process.

Figure 5 represents a general business process between two companies: the process is launched by the production of an order on NIMBLE and the consecutive share of this information with the receiver. In the most common case, the receiver wants to get this information inside its management system to plan the collection of necessary materials and the consecutive production of goods. The simplest way (and the least error-prone) is that the receiver gets a business document from the platform or directly from the other company.

The business process can continue with the following exchanges of data about the production and shipment supported, if requested, by the correspondent business documents.

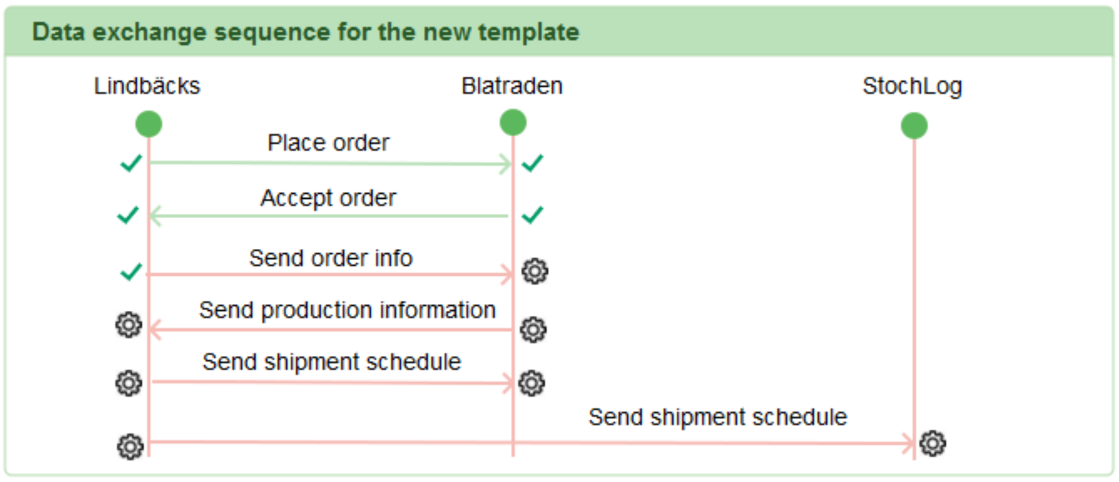


Figure 5: NIMBLE Process description on business collaboration among different companies

In **Figure 5**, a pre-fabricated house producer (Lindbacks) order some modular bathrooms to the manufacturer (Blatraden) and, after the production, a logistic operator (StochLog) is involved for shipment.

The corresponding test of this specific collaboration requires a **sequence** that implements each activity of this business process exactly as the companies do.

An example of this sequence (in XML format specific for testing) can be:

```
<TestActionList>
        <TestAction id="TA-1_TP-NIMBLE" number="1">
                <ActionName>Check Order</ActionName>
        </TestAction>
        <TestAction id="TA-2_TP-NIMBLE " number="2">
                <ActionName>Check Order-Accept</ActionName>
        </TestAction>
        < TestAction id="TA-3_TP-NIMBLE " number="3”>
                 <ActionName>Check Order-Info</ActionName>
        </TestAction>
        <TestAction id="TA-4_TP-NIMBLE " number="4">
                <ActionName>Check Product Info</ActionName>
        </ TestAction>
</TestActionList>
```

The text before, is a fragment of a more complex XML document that formalizes the testplan for the above use case. It can be easily recognised a list of actions, some of them linked with the steps in the business process in **Figure 5**.

Despite the fact that the previous code is less immediate for the reader than **Figure 5** a second reading can help in recognising that it represents in another language the same elements of the business process. It can be easily recognised that this plan covers both the aspects of the collaboration, enabling the actors in testing their systems before the real business can be started.

Even though the initial testing phase is completed correctly, errors in exchanged business data may arise during the collaborations caused by external factors (for example modifications in software tools by one of the companies, introduction of wrong data in internal databases etc…): to avoid the risk of this situation the test of the exchanged documents, when requested by the parties, can be done "*on-the-fly"* on the real data. In real business, anyway, documents have to respect the syntax of the document and the semantic of the data: a test of these elements can reduce the risk of exchanging wrong information thus creating misunderstanding and costs. In this case, each step of the previous plan can be executed separately (e.g. when an order is exchanged through NIMBLE) and automatically by a test service of the platform.

This NIMBLE feature for supporting tests, not already available on the other B2B platforms on the market, reduces the risk related to the use of it and, thus, the *frictions* associated to this process.

Implementing such type of tests in a dynamic business network, as the one originated by the NIMBLE platform, requires the development of several software solutions:

- the realization of easy to use, M2M interfaces allowing NIMBLE to execute tests and collect results in a machine treatable way;
- the creation of plans of test directly from the knowledge stored in NIMBLE (e.g. Products data and business processes) without human intervention that may become unsustainable for the platform.

Following sections of this document, in particular chapters 6 and 7, present the solutions adopted to face these requirements starting from a general-purpose test bed platform.

5 Test Bed Platform (TeBES)

5.1 Supporting adoption of standards by companies

UBL represents the more advanced and complete open standard, based on XML, for supporting business processes in almost all the product markets, covering almost all aspects of the business chain, from raw material collection to the consumer. However, as indicated before, the adoption of this type of standards requires some efforts by companies that can be reduced by the adoption of a specific subset or customisation of the document templates.

Several other actions can be planned and put on the table for supporting companies in adopting the standard as UBL:

- a communicative approach with **examples of successful cases** and a clear definition of the risk (mainly economic) related to this process;
- the support in identifying the company's needs, **requirements** and expectations;
- the provision of **tools** that make less complex the changes in legacy software tools, as ERPs, to accept new types of documents.

The first action is performed, normally, by the standard bodies (as ISO or OASIS) or by institutions that have interest in standard adoption (as national government, in the case of invoices).

The second action is performed by software providers that use the standard in their solutions as a key competitive factor and offer services to goods producers.

Finally, the last action is demanded to organizations involved in standard development or adoption with skills on software design, in particular to research entities or universities. For example, SRDC is supporting the adoption of the HL7 standard for health management with different actions and software provision.

A **test bed platform** for interoperability testing is a tool that can be used to actively support companies, and their software providers, during the process of adoption of a communication standard based on ebXML and XML, such as UBL.

In most cases, the use of an XML **Schema** with a related validation tool, that allows the definition of a document template and checks the document instances against its structure, is enough for checking the elements that characterise the standards. However, it is not powerful enough to support also the features of a use profile in such a way that most errors can be avoided. Returning to the example of shoes in chapter 1, the XML Schema can state that a specific element in the document has to be a number and the value must to be between a maximum and a minimum. But it cannot check the value of an element upon the value of another. For example, the statement "if the destination country is USA, then the size must be comprised among 7 and 14" cannot be expressed by a XML Schema. To solve this problem using only XML Schema, it should be necessary to define a document specific for a single state or region. So far, to support all the countries in the world and possible cross-boarding business collaborations the number of necessary templates may become very large for managing only one element.

XML Schema allows the "syntactic validation" of the document: it checks if the document is correctly formatted, its structure and data delimiters are in the correct position.

For this reason, what is advisable id the adoption of a validation tool in order to create conditional rules between elements in one document or more. In this scenario, Schematron is a rule-based validation language that allows to verify the presence and absence of patterns in XML documents.

For example the simple rule “the order identification number must be the same in all the documents originated from that order”, quite common in business processes, can be easily expressed using Schematron without the necessity of creating an application doing that.

Different from XML Schema, Schematron does not define a single document template but a set of rules that a document must respect to become a valid document in a specific profile.

Furthermore, it allows the creation of rules that involve a whole set of documents, as the previous one.

Schematron allows the “semantic validation” of the content of a business document, by checking the meaning of elements containing data that are present in the document itself.

While in some cases the syntactic validation is enough for establishing business collaboration, more often the semantic validation is required to avoid misunderstandings on data meaning and misinterpretation of the business documents.

5.2 Why a test bed platform?

The instantiation of business collaborations among different actors who are active in a supply chain is simplified by the adoption of specific use profile of the UBL standard in NIMBLE project. The use profile allows the definition of a set of rules that each document must respect to be considered a valid business document during the collaboration and, on the other side, can provide semantic information on the data contained.

The operation of checking all of these rules in one or more documents can be relatively long and costly. For this reason, the CEN organization launched the **GITB** (Global eBusiness Interoperability TestBed) initiative that “*focuses on methodologies and architectures that support e-business standards assessment and testing activities from early stages of eBusiness standards implementation, to proof-of-concept demonstrations, to conformance and interoperability testing*”[6].

The GITB initiative, in the normative documentation released in 2012, has defined several aspects, such as methodologies and architectures, for the document testing process. The GITB initiative moves from the assumption that the testing activity is a **key prerequisite** for e-business interoperability due to the fact that “*it is still cumbersome for software vendors and end-users to demonstrate full compliance with the specified standards and to achieve interoperability of the implementations”*[7]. This sentence has two faces:

1) there is a lack of instruments that can demonstrate the full compliance of a software solution to a specific standard and/or to a specific use profile;

[6] https://www.cen.eu/work/areas/ict/ebusiness/pages/ws-gitb.aspx

[7] CEN – CWA 16408:2012 ftp://ftp.cencenelec.eu/CEN/WhatWeDo/Fields/ICT/eBusiness/WS/GITB/CWA_16408.pdf

2) the effective interoperability among two or more implementations is not guaranteed by the simple fact that all actors use the same standard or even the same profile.

For these reasons CEN suggested the implementation of well-defined testbed solutions that can "certify" that a software solution is fully compliant with a standard.
In 2012 ENEA has launched an internal project, called TeBES, for implementing a GITB compliant solution for supporting interoperability testing and standard fostering in productive sectors.

The two main key elements in the GITB architecture are:

- modularity of the software components that are expected to be interchangeable;
- separation between the software modules of the platform and the instruction for planning and executing tests (descriptors of the planning, descriptors of the test); this allows to reuse the same platform for different types of tests on different domains.

The main benefit of this approach is that the same software platform might be used for many different kinds of test and test strategies with different specifications without changing the software.

In the next sections a short overview of the platform is presented with a focus on the instructions for the platform, organised in test plan, test actions and test suites.

5.3 Describing the test: Testplan

The main objective of a testbed platform is to support software vendors to demonstrate the compliance of their software solutions with a specific standard or to a part of it. The **Testplan** is a sort of guideline, human readable, which on one side allows the testbed platform to execute the test according to a strategy and, on the other, inform the user on how the test will run.

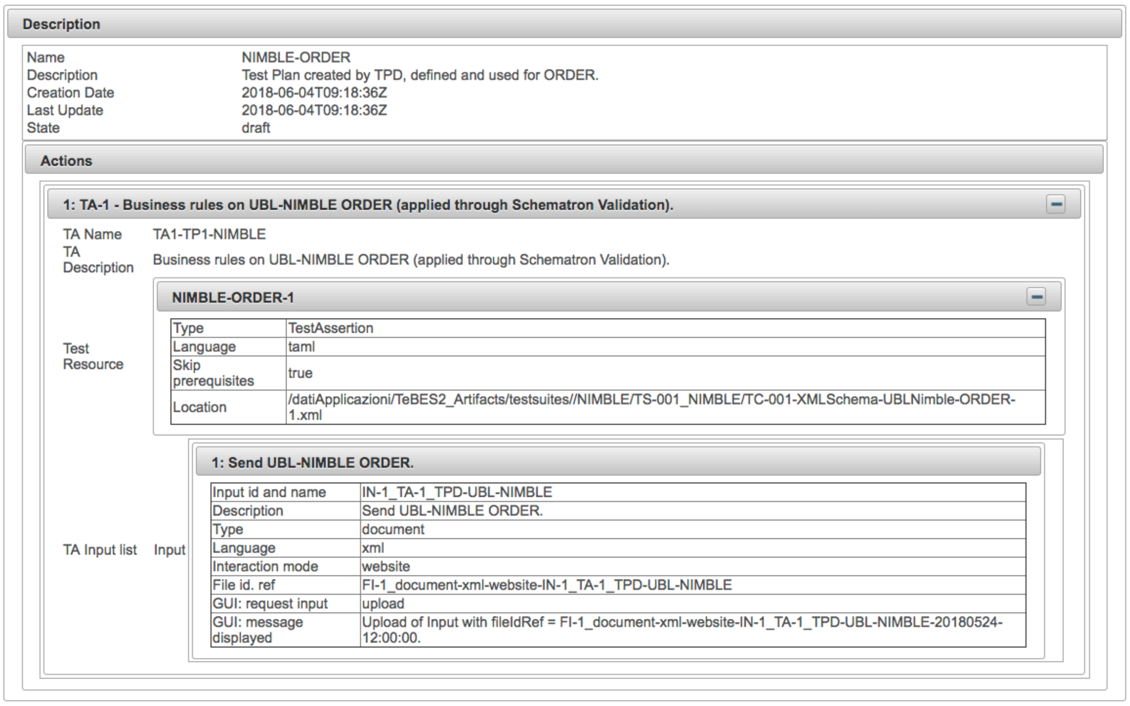
Examples of different testing strategies: "one-shot conformance test" (all the rules and constraints are applied in a single step of validation); "learn step by step by example" (different sets of rules are applied in a path that, through a number of steps, leads to the complete conformance assessment of a type of templates).

The Testplan is an XML document composed by different elements, from general information for the end user to more specific instruction to execute the test.

Figure 6 gives an idea of a very simple Testplan that aims at syntactically validating a NIMBLE order using a specific XML schema. In this case, the Testplan is composed by a single action.

The header is a simple textual description of the Testplan identified by a clear name, a short description and several auxiliary information on the creation date and successive updates. The "state" descriptor indicates if it can be improved in the future or not.

The "*actions"* section contains more operative data about the test, and in particular the list of actions that constitute the test plan. Each action is composed by a "test resource" that indicates what to test and by one or more inputs that are the objects to be tested. A detailed description of the concept of Testaction is presented later on this document.



Description

Name	NIMBLE-ORDER
Description	Test Plan created by TPD, defined and used for ORDER.
Creation Date	2018-06-04T09:18:36Z
Last Update	2018-06-04T09:18:36Z
State	draft

Actions

1: TA-1 - Business rules on UBL-NIMBLE ORDER (applied through Schematron Validation).

TA Name	TA1-TP1-NIMBLE
TA Description	Business rules on UBL-NIMBLE ORDER (applied through Schematron Validation).

Test Resource

NIMBLE-ORDER-1

Type	TestAssertion
Language	taml
Skip prerequisites	true
Location	/datiApplicazioni/TeBES2_Artifacts/testsuites//NIMBLE/TS-001_NIMBLE/TC-001-XMLSchema-UBLNimble-ORDER-1.xml

TA Input list Input

1: Send UBL-NIMBLE ORDER.

Input id and name	IN-1_TA-1_TPD-UBL-NIMBLE
Description	Send UBL-NIMBLE ORDER.
Type	document
Language	xml
Interaction mode	website
File id. ref	FI-1_document-xml-website-IN-1_TA-1_TPD-UBL-NIMBLE
GUI: request input	upload
GUI: message displayed	Upload of Input with fileIdRef = FI-1_document-xml-website-IN-1_TA-1_TPD-UBL-NIMBLE-20180524-12:00:00.

Figure 6: Test plan visual representation

5.4 Implementing the test: TestAction, TestAssertion and input elements

Looking at **Figure 6** it is worth to note that the test actions are composed by three main parts:

- a simple description of the action;
- the definition of the test resource, which identifies the type of action and the remote resource that contains operative data about it;
- the definition of a list of inputs to feed the test engine.

One of the objectives of TeBES is reusing as much as possible all resources related to a single test by decomposing it into simplest elements that can be recombined to create new test plans. The reason for this is the high effort necessary to think and formally describe the rules to be used in an interoperability testing strategy: they usually embed a high-level knowledge on interoperability and on specific domains.

While the test action is composed mainly by two parts, such that the first part identifies what and how to test and the other that indicates the input elements that must be tested, it is possible to reuse the same resource data to build up a new test action that may have different input.

5.4.1 Test Assertion and Test Suites

Few elements characterise the test items:

- the type of the resource as in the CEN CWA indicated before;
- the XML based language that describes the resource (in this case TAML);
- the parameter that allows skipping of all prerequisites (elements that otherwise have to be tested before the current one);
- the link to the implementation of the resource.

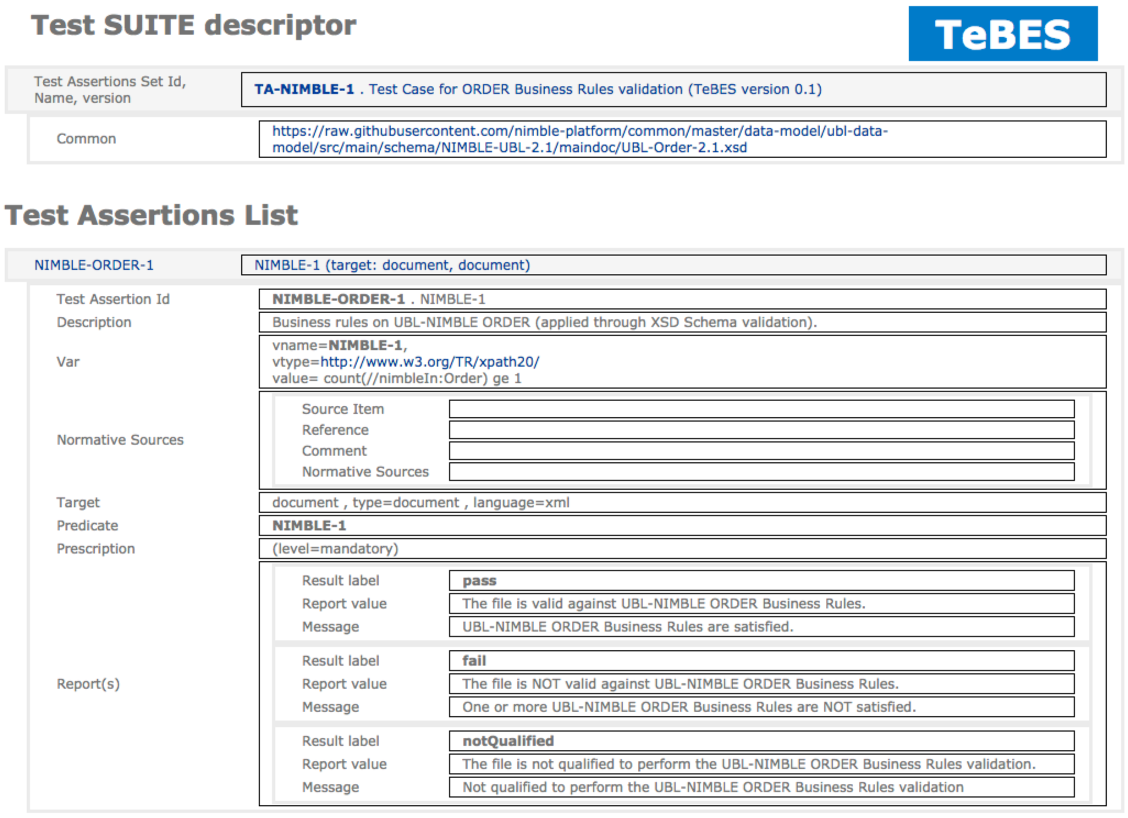
In order to simplify the reuse of knowledge already available, the different resources are put together into a **Test Suite** that contains different resource semantically or logically linked to each other. For example, the Test Suite for the NIMBLE platform will collect all the test resources connected with this platform.

Looking at the test suite XML file, it is possible to notice several elements:

- a short description containing also the identification code of the test suite itself;
- a set of common resources (for example the XSD schemas);
- a list of **Test Assertions** that are the core of the test.

The test assertion defines the “rules” that a document must respect, the prescription level for this rule and some response for reporting results. It also defines what type of input is expected by the Assertion (in this case an XML document) and what output is produced (the part of the final report on the test).

Figure 7 gives an example of a TestAssertion created for the NIMBLE TestSuite. It is worth to note that some elements are empty, in particular the part of the **Normative Sources** for the test, which do not apply for NIMBLE at the moment because a specific standard for business documents has not been selected, yet.



Test SUITE descriptor

TeBES

Test Assertions Set Id, Name, version	**TA-NIMBLE-1** . Test Case for ORDER Business Rules validation (TeBES version 0.1)
Common	https://raw.githubusercontent.com/nimble-platform/common/master/data-model/ubl-data-model/src/main/schema/NIMBLE-UBL-2.1/maindoc/UBL-Order-2.1.xsd

Test Assertions List

NIMBLE-ORDER-1	NIMBLE-1 (target: document, document)	
Test Assertion Id	**NIMBLE-ORDER-1** . NIMBLE-1	
Description	Business rules on UBL-NIMBLE ORDER (applied through XSD Schema validation).	
Var	vname=**NIMBLE-1**, vtype=http://www.w3.org/TR/xpath20/ value= count(//nimbleIn:Order) ge 1	
Normative Sources	Source Item	
	Reference	
	Comment	
	Normative Sources	
Target	document , type=document , language=xml	
Predicate	**NIMBLE-1**	
Prescription	(level=mandatory)	
Report(s)	Result label	**pass**
	Report value	The file is valid against UBL-NIMBLE ORDER Business Rules.
	Message	UBL-NIMBLE ORDER Business Rules are satisfied.
	Result label	**fail**
	Report value	The file is NOT valid against UBL-NIMBLE ORDER Business Rules.
	Message	One or more UBL-NIMBLE ORDER Business Rules are NOT satisfied.
	Result label	**notQualified**
	Report value	The file is not qualified to perform the UBL-NIMBLE ORDER Business Rules validation.
	Message	Not qualified to perform the UBL-NIMBLE ORDER Business Rules validation

Figure 7: An example of TestAssertion for NIMBLE

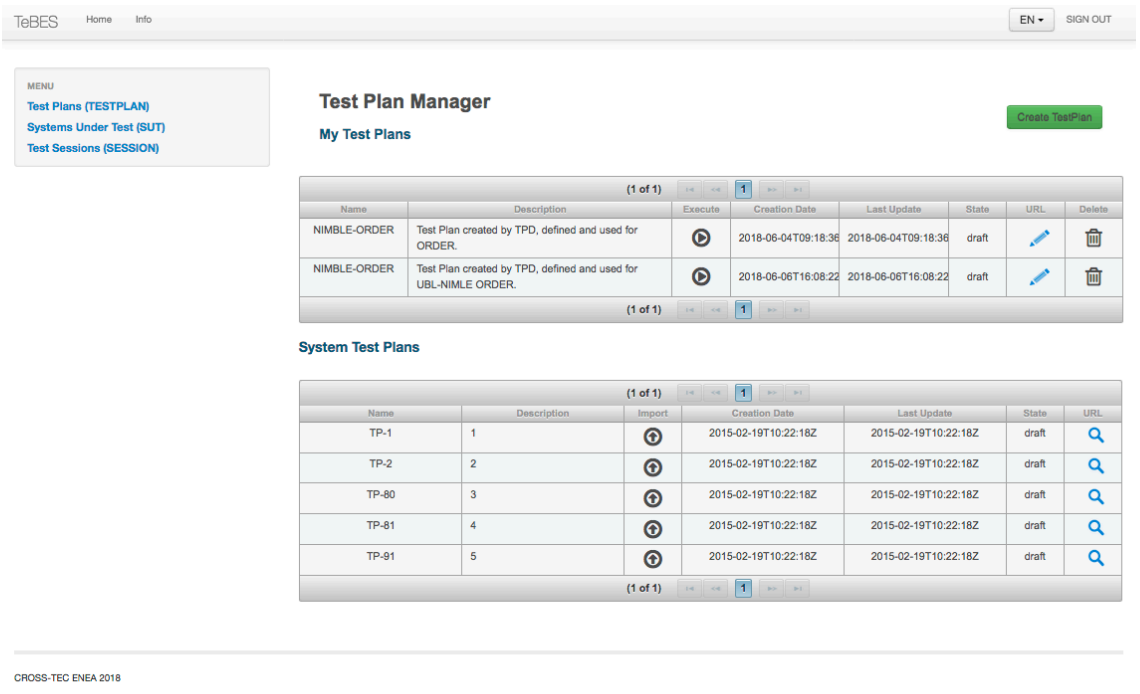
5.5 Executing the test: TeBES platform

The GITB infrastructure for testing is quite simple: few elements can be used, using powerful tools such as XML Schema and Schematron, to describe almost all possible scenarios for testing the interoperability among different software systems.

The TeBES platform is a web-based[8] application that allows the implementation of the Testplans as a set of actions that requires the interaction with the user (human or not-human) through a precise path described in the Testplan itself. This path is determined by both the actions contained in the Testplan and the order of these actions. Actions can be simple (for example requesting the upload of a single document) or complex, requiring multiple inputs in the correct order.

[8] A beta version of the platform is at link: http://winter.bologna.enea.it/tebes2. Dummy user for testing the platform is: username="dybuwy@yahoo.com", password="password" without quotes.

After a simple login page (or registration for new user), the user can view/run the Testplans, choose or modify the "System Under Test" options or receive information about past and active testing sessions.



Test Plan Manager

My Test Plans

Name	Description	Execute	Creation Date	Last Update	State	URL	Delete
NIMBLE-ORDER	Test Plan created by TPD, defined and used for ORDER.	▶	2018-06-04T09:18:36	2018-06-04T09:18:36	draft	✎	🗑
NIMBLE-ORDER	Test Plan created by TPD, defined and used for UBL-NIMLE ORDER.	▶	2018-06-06T16:08:22	2018-06-06T16:08:22	draft	✎	🗑

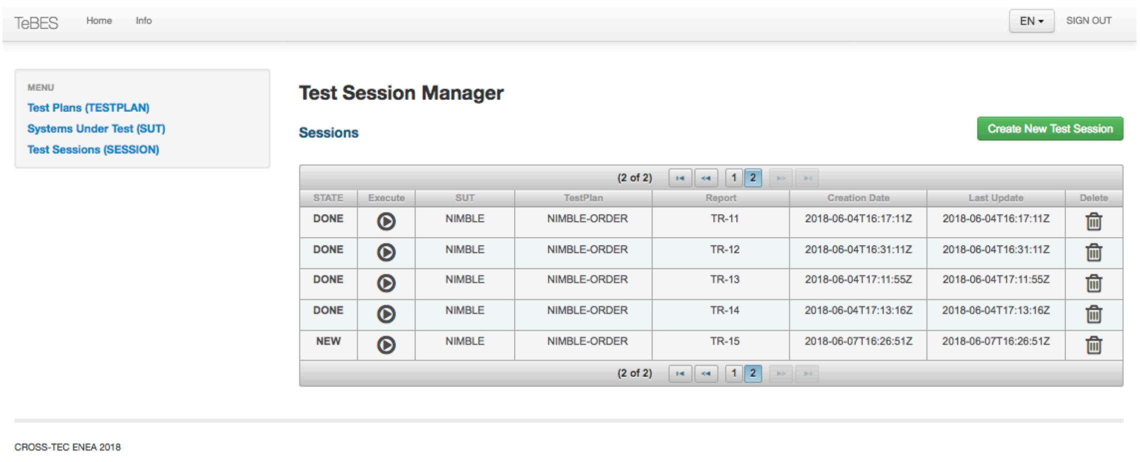
System Test Plans

Name	Description	Import	Creation Date	Last Update	State	URL
TP-1	1	⬆	2015-02-19T10:22:18Z	2015-02-19T10:22:18Z	draft	🔍
TP-2	2	⬆	2015-02-19T10:22:18Z	2015-02-19T10:22:18Z	draft	🔍
TP-80	3	⬆	2015-02-19T10:22:18Z	2015-02-19T10:22:18Z	draft	🔍
TP-81	4	⬆	2015-02-19T10:22:18Z	2015-02-19T10:22:18Z	draft	🔍
TP-91	5	⬆	2015-02-19T10:22:18Z	2015-02-19T10:22:18Z	draft	🔍

Figure 8: Test plan manager page of TeBES

A set of example test plan is already available on the platform: users can use them or create a new one by uploading the necessary XML files. After this process is complete, the Testplans are available in this page and can be run with a new test session.

At this point it is necessary to briefly introduce the concept of SUT (**System Under Test**) that represents the object that the user wants to test against a Testplan: few data are necessary to identify the SUT. In the case of NIMBLE, the SUT is the legacy system of the user and is always characterised as a system capable of uploading documents (even without the supervision of a human user). A single user can define one or more SUTs.



Test Session Manager

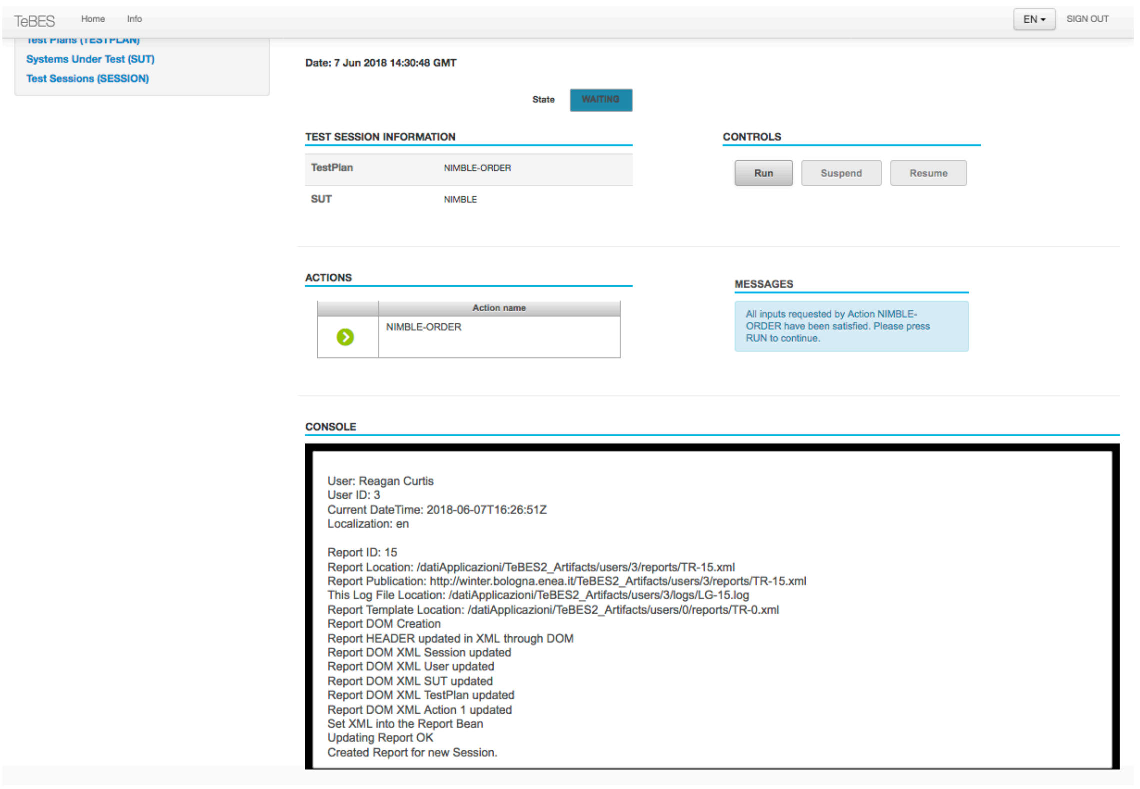
Sessions

STATE	Execute	SUT	TestPlan	Report	Creation Date	Last Update	Delete
DONE	▶	NIMBLE	NIMBLE-ORDER	TR-11	2018-06-04T16:17:11Z	2018-06-04T16:17:11Z	🗑
DONE	▶	NIMBLE	NIMBLE-ORDER	TR-12	2018-06-04T16:31:11Z	2018-06-04T16:31:11Z	🗑
DONE	▶	NIMBLE	NIMBLE-ORDER	TR-13	2018-06-04T17:11:55Z	2018-06-04T17:11:55Z	🗑
DONE	▶	NIMBLE	NIMBLE-ORDER	TR-14	2018-06-04T17:13:16Z	2018-06-04T17:13:16Z	🗑
NEW	▶	NIMBLE	NIMBLE-ORDER	TR-15	2018-06-07T16:26:51Z	2018-06-07T16:26:51Z	🗑

Figure 9: List of available user's test session on TeBES

After the identification of the Testplans and SUT, the user can access a list of test sessions already completed or new ones.

The execution of a test session is shown in Figure 10: this page allows the user to insert/upload the necessary inputs produced by the SUT that can be tested using the rules specified in each test action. In this case, the test plan is testing the validity of a NIMBLE order and requires as input an XML instance of this order.



Date: 7 Jun 2018 14:30:48 GMT

TEST SESSION INFORMATION

TestPlan	NIMBLE-ORDER
SUT	NIMBLE

ACTIONS

	Action name
●	NIMBLE-ORDER

MESSAGES

All inputs requested by Action NIMBLE-ORDER have been satisfied. Please press RUN to continue.

CONSOLE

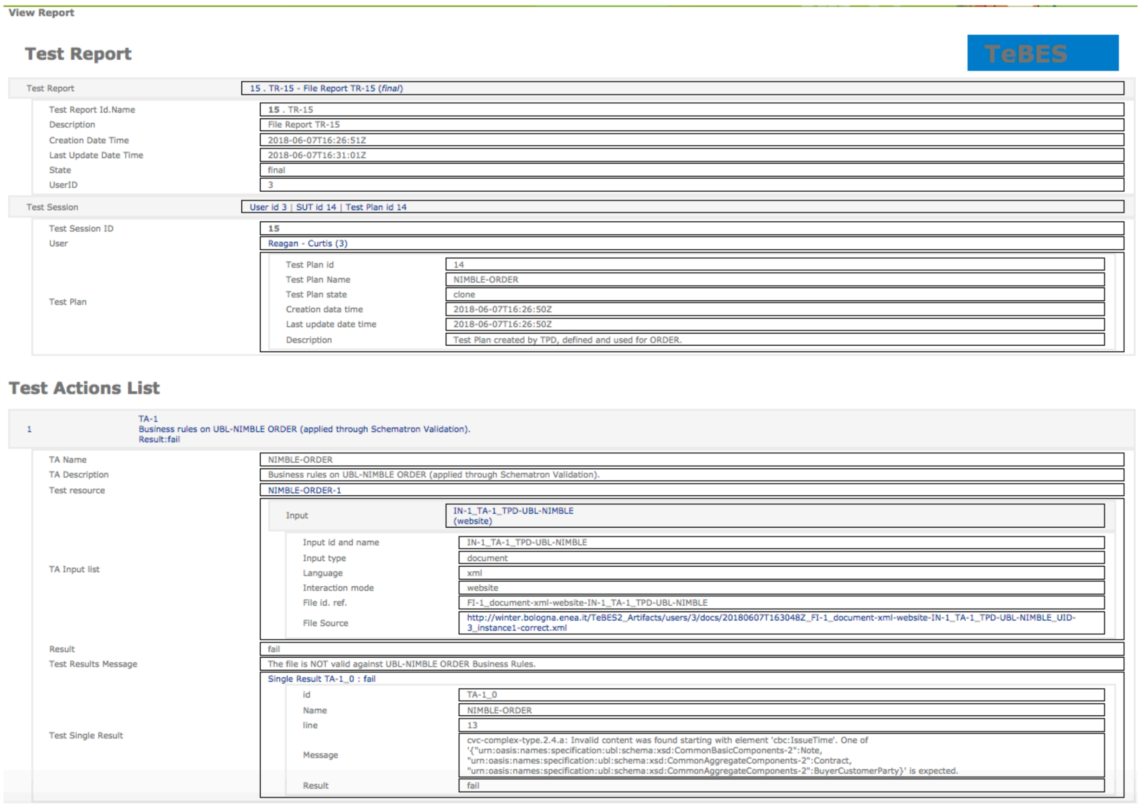
```
User: Reagan Curtis
User ID: 3
Current DateTime: 2018-06-07T16:26:51Z
Localization: en

Report ID: 15
Report Location: /datiApplicazioni/TeBES2_Artifacts/users/3/reports/TR-15.xml
Report Publication: http://winter.bologna.enea.it/TeBES2_Artifacts/users/3/reports/TR-15.xml
This Log File Location: /datiApplicazioni/TeBES2_Artifacts/users/3/logs/LG-15.log
Report Template Location: /datiApplicazioni/TeBES2_Artifacts/users/0/reports/TR-0.xml
Report DOM Creation
Report HEADER updated in XML through DOM
Report DOM XML Session updated
Report DOM XML User updated
Report DOM XML SUT updated
Report DOM XML TestPlan updated
Report DOM XML Action 1 updated
Set XML into the Report Bean
Updating Report OK
Created Report for new Session.
```

Figure 10: Test Session execution page

After the completion of all the actions in the Testplan, the complete report is available on the Test Sessions page.

A report showing that the document is not a valid NIMBLE-ORDER is shown in **Figure 11**



View Report

Test Report

TeBES

Test Report	15 . TR-15 - File Report TR-15 (*final*)
Test Report Id.Name	15 . TR-15
Description	File Report TR-15
Creation Date Time	2018-06-07T16:26:51Z
Last Update Date Time	2018-06-07T16:31:01Z
State	final
UserID	3
Test Session	User id 3 | SUT id 14 | Test Plan id 14
Test Session ID	15
User	Reagan - Curtis (3)
Test Plan	Test Plan id: 14
	Test Plan Name: NIMBLE-ORDER
	Test Plan state: clone
	Creation data time: 2018-06-07T16:26:50Z
	Last update date time: 2018-06-07T16:26:50Z
	Description: Test Plan created by TPD, defined and used for ORDER.

Test Actions List

1 TA-1
Business rules on UBL-NIMBLE ORDER (applied through Schematron Validation).
Result:fail

TA Name	NIMBLE-ORDER
TA Description	Business rules on UBL-NIMBLE ORDER (applied through Schematron Validation).
Test resource	NIMBLE-ORDER-1
TA Input list	Input: IN-1_TA-1_TPD-UBL-NIMBLE (website)
	Input id and name: IN-1_TA-1_TPD-UBL-NIMBLE
	Input type: document
	Language: xml
	Interaction mode: website
	File id. ref.: FI-1_document-xml-website-IN-1_TA-1_TPD-UBL-NIMBLE
	File Source: http://winter.bologna.enea.it/TeBES2_Artifacts/users/3/docs/20180607T163048Z_FI-1_document-xml-website-IN-1_TA-1_TPD-UBL-NIMBLE_UID-3_instance1-correct.xml
Result	fail
Test Results Message	The file is NOT valid against UBL-NIMBLE ORDER Business Rules.
Test Single Result	Single Result TA-1_0 : fail
	id: TA-1_0
	Name: NIMBLE-ORDER
	line: 13
	Message: cvc-complex-type.2.4.a: Invalid content was found starting with element 'cbc:IssueTime'. One of '{"urn:oasis:names:specification:ubl:schema:xsd:CommonBasicComponents-2":Note, "urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2":Contract, "urn:oasis:names:specification:ubl:schema:xsd:CommonAggregateComponents-2":BuyerCustomerParty}' is expected.
	Result: fail

Figure 11: Report for a test session

In this case, the message in the last part of Figure **11** indicates that the test has failed and shows the error message originated by the system, allowing the user to understand what errors have been found.

Using more complex rule sets or Schematron tools it is possible to increase the quality of the test thanks to tests on the semantic validity of the document by the same way: when an error is detected, the causes are reported in this part of the report to support the improvement of the SUT.

Figure **12** depicts a simple UML diagram of the use of TeBES platform, as a web-based service, to test the semantic and syntax of a document. The user generates a document and selects which testing activity to run. After the upload of the document is completed, TeBES runs the specific Testplan and produces the report, sending it to the user. The evaluation of the testing report by the user can allow the improvement of the document (e.g. some errors are corrected) and the loop of this process until the test is passed.

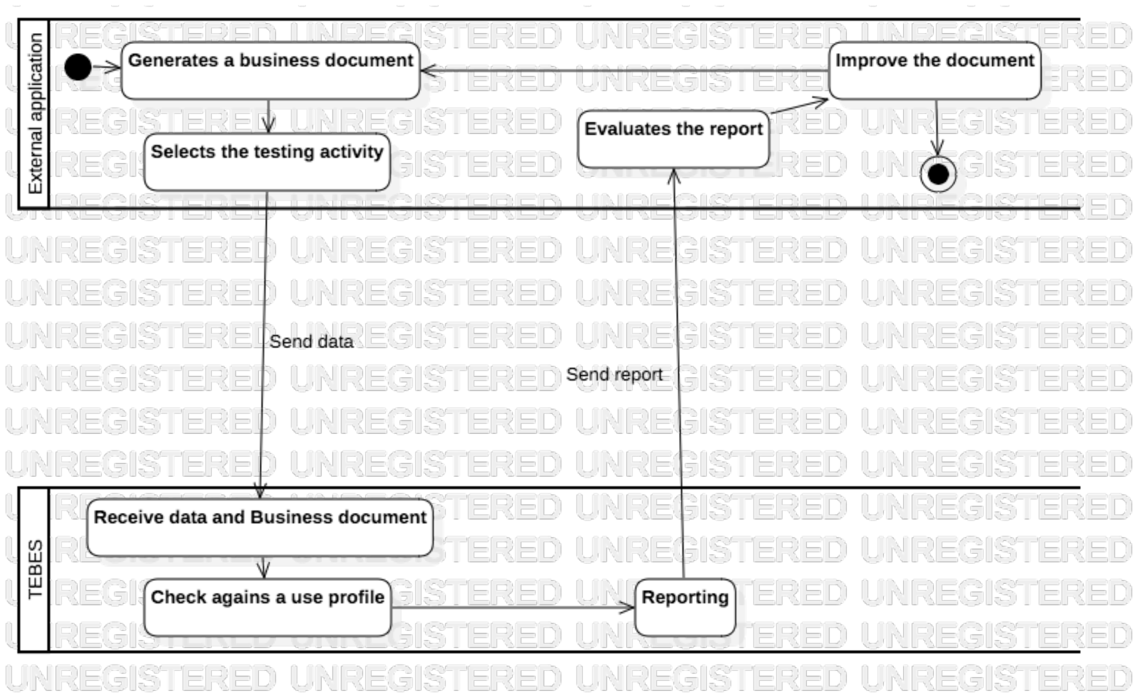

External application
Generates a business document
Improve the document
Selects the testing activity
Evaluates the report
Send data
Send report
TEBES
Receive data and Business document
Check agains a use profile
Reporting
UNREGISTERED

Figure 12: Simple UML activity diagram on the interaction among the user and TeBES

6 Testing in NIMBLE: Integrating TeBES and NIMBLE platforms

In a B2B scenario in which the data transactions are mediated by a platform such as NIMBLE, the testing activity requires different improvements to allow the NIMBLE platform to:

- support the users with the process of adopting a standard or starting a new collaboration (for example with a partner that requires modifications on exchanged documents);
- check the documents correctness before importing and, successively, forwarding it to recipients;
- identify possible errors in documents originated by software that has been modified or newly developed (for example during the conversions from one format to another).

Human interfaces are necessary only for the first bullet point, while for the others direct and faster computer-to-computer interfaces are required so that the provisioning of services substitutes human intervention.

NIMBLE is a platform using largely the REST technologies for data exchange and interaction between the different software components: for this reason TeBES has been provided with a simple REST interface for interacting directly with the other software components in NIMBLE.

For this reason, a set of simple REST calls have been set up:

Table 3: RunTestPlan REST calls

Service Name	Service Path	Service Description
listTestPlans (GET)	Lists Header: Token	Returns the list of available test plans, in JSON format, for the specific user identified by a token string obtained by the identity manager implemented in NIMBLE
executeTestPlan (POST)	executeTestPlan Header: token Param: XMLInput Param: TesplanId	Returns the JSON string with the result of testing activities using the TestplanId and the content of the XMLInput

Together with REST calls, a set of JAVA APIs is available for accessing the same services.

Table 4: RunTestplan JAVA methods

Method Name	Method Parameters	Method Description
listTestPlans	*String Token*	Returns the list of available test plans for the specific user identified by a token string obtained by the identity manager implemented in NIMBLE
executeTestPlan	String Token String XMLInput String TestplanId	Returns the result of the testing using the TestplanId and the content of the XMLInput
listTestSessions	String Token	Returns the list of user's test sessions that contain the data about previous testing activities

Using a RESTFul client as Postman[9], it is possible to test the calls and send to TeBES the data necessary to simulate the collaboration with the NIMBLE platform.

On the other side, it is necessary to develop a RESTful client in NIMBLE that allows the "on-the-fly" test of the documents.

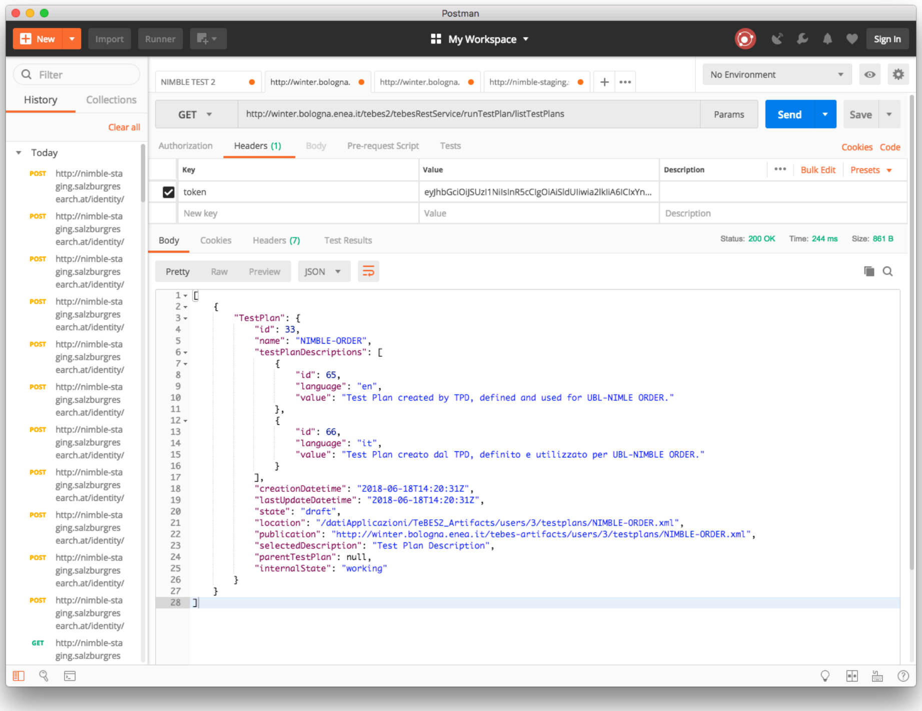
The following images are screenshots taken from Postman showing the results of the different calls:

Figure **13**: The call *listTestPlans* returns the user's available test plan, in this case the one related to the *NIMBLE-Order*. This information allows the user to select which Testplan to run. It requires as input the *identity token* managed by NIMBLE platform.

Figure **14**: the POST call executeTestPlan get as input the XML file (as a String uploaded through Postman), the number of the Testplan and the user's token to run the test. It returns a general status of the test (failure or success) and a link to the full report.

In the real collaboration among NIMBLE and TeBES, the REST calls are automatically made by NIMBLE. Successively, it processes the results of the test and decides the successive actions after the testing phase ends.

[9] Available at: https://www.getpostman.com/



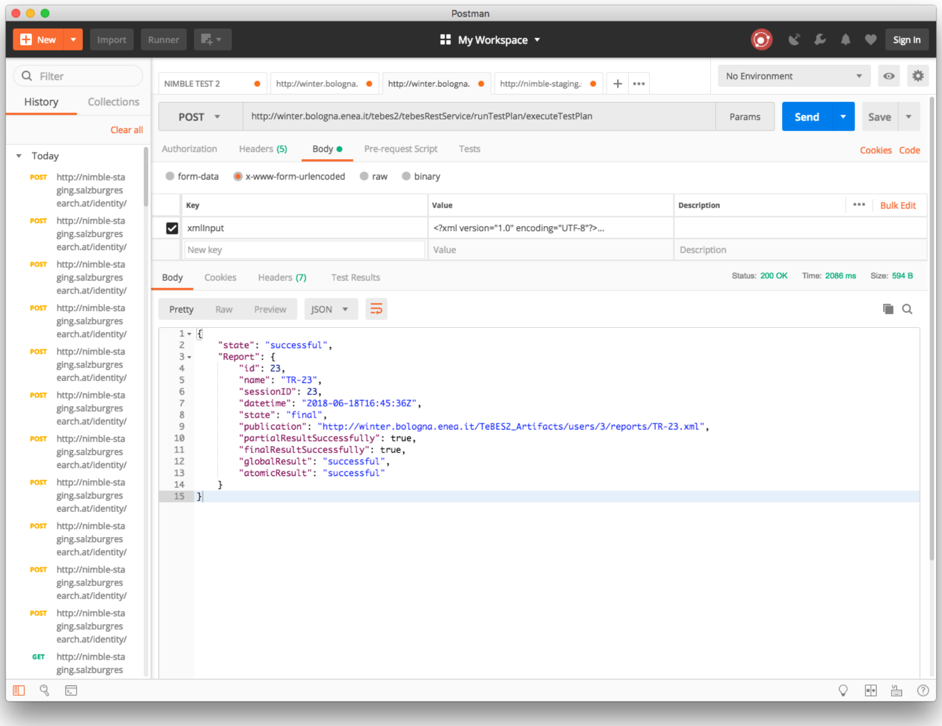
GET http://winter.bologna.enea.it/tebes2/tebesRestService/runTestPlan/listTestPlans

Key	Value	Description
token	eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICIxYn...	

Status: 200 OK Time: 244 ms Size: 861 B

```
[
    {
        "TestPlan": {
            "id": 33,
            "name": "NIMBLE-ORDER",
            "testPlanDescriptions": [
                {
                    "id": 65,
                    "language": "en",
                    "value": "Test Plan created by TPD, defined and used for UBL-NIMLE ORDER."
                },
                {
                    "id": 66,
                    "language": "it",
                    "value": "Test Plan creato dal TPD, definito e utilizzato per UBL-NIMBLE ORDER."
                }
            ],
            "creationDatetime": "2018-06-18T14:20:31Z",
            "lastUpdateDatetime": "2018-06-18T14:20:31Z",
            "state": "draft",
            "location": "/datiApplicazioni/TeBES2_Artifacts/users/3/testplans/NIMBLE-ORDER.xml",
            "publication": "http://winter.bologna.enea.it/tebes-artifacts/users/3/testplans/NIMBLE-ORDER.xml",
            "selectedDescription": "Test Plan Description",
            "parentTestPlan": null,
            "internalState": "working"
        }
    }
]
```

Figure 13: *listTestPlans* GET call



POST http://winter.bologna.enea.it/tebes2/tebesRestService/runTestPlan/executeTestPlan

Body: x-www-form-urlencoded

Key	Value	Description
xmlInput	<?xml version="1.0" encoding="UTF-8"?>...	

Status: 200 OK Time: 2086 ms Size: 594 B

```
{
    "state": "successful",
    "Report": {
        "id": 23,
        "name": "TR-23",
        "sessionID": 23,
        "datetime": "2018-06-18T16:45:36Z",
        "state": "final",
        "publication": "http://winter.bologna.enea.it/TeBES2_Artifacts/users/3/reports/TR-23.xml",
        "partialResultSuccessfully": true,
        "finalResultSuccessfully": true,
        "globalResult": "successful",
        "atomicResult": "successful"
    }
}
```

Figure 14: executeTestPlan POST call

Figure **15** provides a simple UML activity diagram that indicates how the collaboration with TeBES and NIMBLE can reduce the risk of input errors into the platform due to business document containing wrong data.
The user's legacy system (External application) creates a business document (e.g. an order for another user) and sends it to NIMBLE using a communication channel. The NIMBLE platform receives the document and asks TeBES for syntactic and/or semantic tests. TeBES collects information about the context and creates the set of rules to be verified on this specific document. After the test is complete, TeBES shares the result with NIMBLE that, based on this result, can accept or refuse the document. In the last case, it can inform the user about the situation (e.g. sharing the result of the test) asking for document improvements.

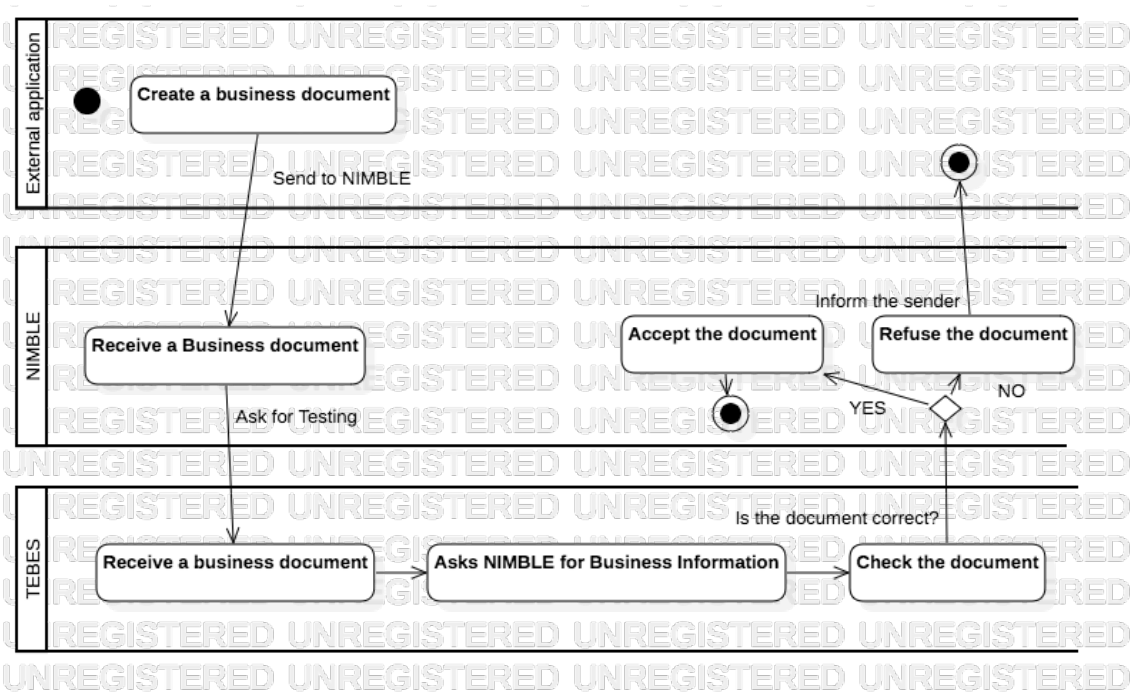


Figure 15: UML activity diagram of the collaboration among TeBES and NIMBLE

In this scenario, the test phase may be facultative for the user involved in the business process: only if the receiver asks NIMBLE for document verification, the testing process on the incoming documents can be activated.

7 Testing in NIMBLE: generate Testplan from business knowledge

NIMBLE is facing a dynamic productive world where the separations among different supply chains are not so clear and defined and the same business process changes rapidly. For this reason it is not reasonable to create the necessary Testplans manually starting from the actual business processes present in NIMBLE but it is necessary to implement a solution that is able to extract information from the internal knowledge bases and use it to create Testplans.

In NIMBLE there are already ontologies containing knowledge about the business plans implemented for different sectors (partially extracted from UBL and similar initiatives) and they are a very good source of information for building up a Testplan in a dynamic way and only when requested.

In the following sections the technical solution for this issue adopted by the project is reported.

7.1 Ontologies & rules (in NIMBLE): the Test Plan Designer (TPD)

The general idea is to get knowledge about the sectorial processes already described by the NIMBLE ontologies, describing the general knowledge about the most frequent supply processes in general and the requirements (rules) for testing related to specific domains and goods.

The aim is to be able to automatically generate Testplans and test resources tailored for the specific needs thanks to:

1) mechanisms of inheritance of requirements thanks to the hierarchical classification of the processes and transactions and the goods they are dealing with and the domains,

2) specialization mechanisms that transform abstract rules into syntax specific rules that are executable for a specific language or specification (like UBL) through a mapping.

The **Test Plan Designer (TPD)** is a software module designed and developed to automatically generate test plans that feed the Test Bed Platform (TeBES) using knowledge stored in different ontologies.

The TPD receives as input a description of the business scenario for which the test plan has to be created, and through a series of elaborations and subsequent steps, generates a set of linked files that constitute the test plan.

The ontologies in NIMBLE contain data about the specific business plan and, therefore, can define completely the relevant business scenario in terms of involved actors, necessary activities, data transactions and items exchanged. The group of these ontologies is defined as the "*Context Ontology*". However, the knowledge already presents in NIMBLE is not enough for building the Testplans because the *rules* are not defined, yet. A *rule* is a statement that a business document must observe to be compliant with a standard profile. The abstraction of these *rules* into a set of abstract rules that can be proved on almost all types of business document constitutes the "*Rules Ontology*" that is used by the software for creating Testplans.

The definition of the abstract rule set will be done together with NIMBLE use cases leaders if necessary.

The two ontologies are strongly different: while the Rules Ontology is simple but rigidly specified, and must be respected for the functioning of the TPD, the Context Ontology is not strictly fixed, but may vary depending on the use and of the rules that must be created.

7.1.1 Rules Ontology

The "Rules Ontology" contains the definition of the concept of **Rule**, and of all its properties. These properties allow defining the context of application of a rule, and to fix all the information associated with it. Rules can be organized in taxonomy, created freely.

To describe the concept of Rule, some existing properties are used (with namespaces rdf, rdfs and owl, for example) and, moreover, some new properties have been defined. The complete list of the properties used to manage the Rules Ontology is as follows:

Property	**Abstract Name**	**Description**
rdf:about	Instance/rule identifier	Identifier of the instance of the rule in the ontology
rdfs:label	First name	Rule name
rdfs:comment	Description	Text description of the rule
tr:isAppliedTo	is applied to	Indicates the extent of the ontology to which a rule is to be applied
tr:hasRule	has rule	Indicates a rule that is defined on an entity
tr:hasScope	has scope	Indicates a business scope in which a rule is valid and applicable, multiple scopes can be specified (united with a logical AND)
tr:Hasid	Identifier rules	Further identifier of the rule in the ontology (not necessary, the instance ID can also be used)
tr:hasPrescriptionLevel	Prescription level (importance) of the rule	Indicates the type of importance of the error in the rule (for example with indications like warning, error)

tr:hasAbstractRule	Abstract rule	Provides the abstract formalization of the rule
tr:hasAbstractContext	Abstract backdrop	Provides the validity context of the rule for its Schematron implementation
tr:insertedBy	Posted by	Specify who has inserted into the ontology the rule
rdfs:isDefinedBy statement	Defined by	Indicates who has defined and formalized the rule with its properties
tr:hasInsertDate	Date of input	Date when the rule was entered into the ontology
owl:versionInfo	Version	Version of the rule
owl:deprecated	It is deprecated	Indicates if the rule is deprecated.
tr:useTechnology	Technology adopted	Indicate how and with what technology the rule is implemented
tr:hasPriority	Priority	Indicates the priority of the rule
tr:hasParameter	Parameters	Provides the parameters of the rule
tr:hasImplementation	Implementation file	Indicates the files in which the verification/execution of the rule is implemented

Not all of these properties are mandatory in the description of a rule, only those highlighted in green. These are the rules that define the context of application of the rule and its abstract formalization. Some properties were included thinking about a support for future developments.

In particular, the properties tr:isAppliedTo and tr:hasScope can refer to an external ontology (which can change, and be chosen according to the needs) that defines the concepts of the business.

For example the definition of a simple abstract rule (where a specific context is not defined) could be:

-	IDRule
rdfs:label	ID Rule-01
rdfs:comment	The document must have an identifier in any case.
tr:isAppliedTo	Document
tr:hasPrescriptionLevel	fatal
tr:hasAbstractRule	**$id**

tr:hasAbstractContext	/*
tr:insertedBy	John Smith
rdfs:isDefinedBy statement	AENEAS
tr:hasInsertDate	01/10/2017
owl:versionInfo	1.0
owl:deprecated	no

This rule is very generic and states that every business document must be identifiable through a unique ID number, helping all the parties in avoiding confusion about the single document and its content. It can be assumed as valid for every specification on business documents.

But, written has "$id", this rule cannot be easily verified in a document because the system is not able to recognize in the document which element contains the data related to the id number. For this reason, in addition to the definition of the generic rule, it is clearly necessary to define a **mapping table** (see the **Translation configuration** section) that translates the abstract version of the rule into an executable rule.

7.1.2 Ontology of contexts[10]

The "Context Ontology" provides the basis for specifying the context of application and validity of the rules defined in the rules ontology. This ontology is not necessarily a static one, but it can be changed according to the rules that must be defined and precisely to their context of application. The ontology of contexts can also be the sum of various ontologies, each of them specialized in a particular area, which are included and integrated with each other.

Within the ontology of contexts, it is possible to define up to a maximum of 5 independent main concepts: Activity, Actor, Commercial Good (Product?), Document and Process.

These concepts define the application context of the rules (see section "Specifying the scenario for which to create Test Plans").

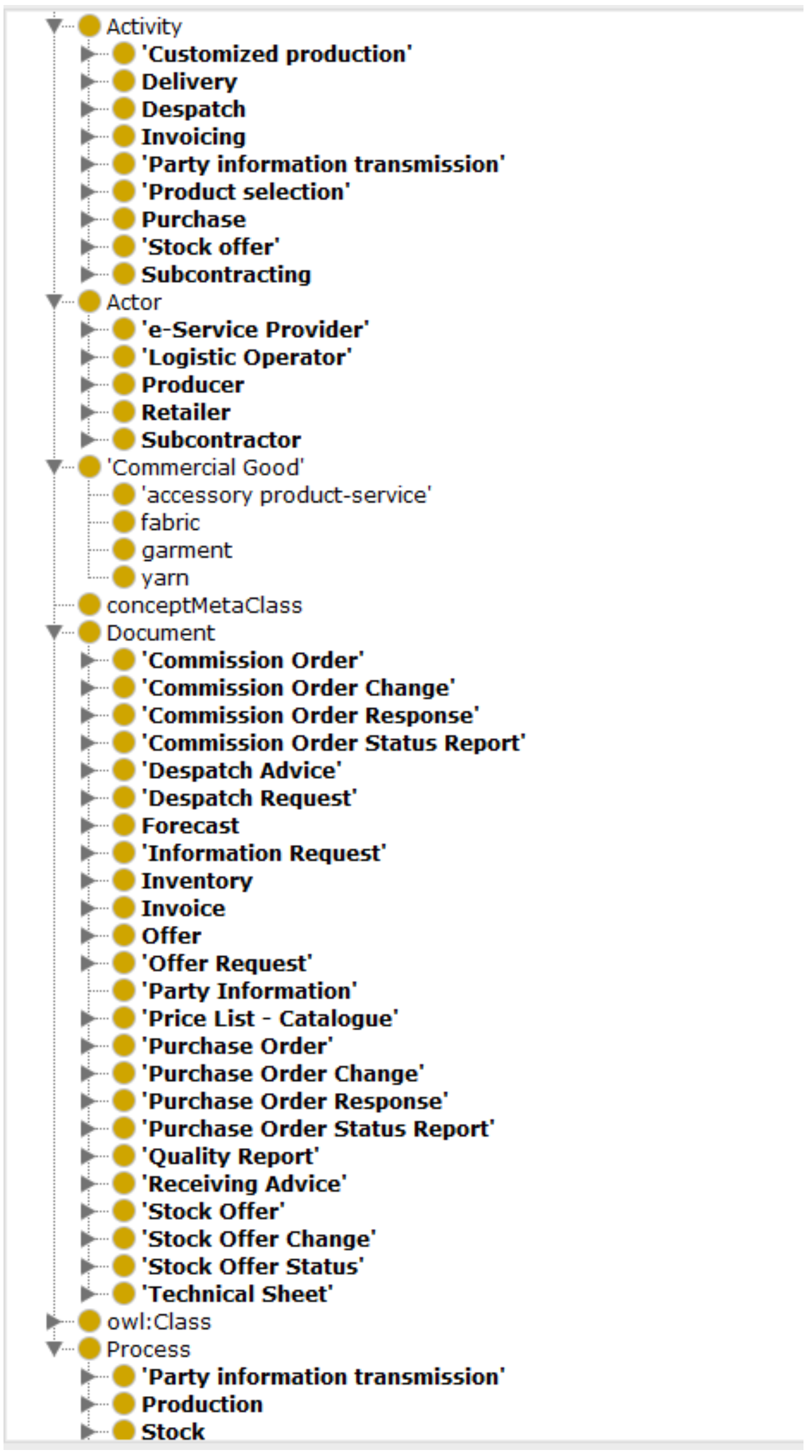
Concepts inside this ontology are inter-related to each other in a specific business context: for example the Process "Stock" is linked to the Activity "Stock Offer" that implies the exchange of the Documents "Stock Offer", "Stock Offer Change" and "Stock Offer Status", furthermore it involves the Actors "Yarn Producer" and "Textile Producer" in the category "Producer" and allows the stoking of Goods "Yarns".

As the example suggests, the combination of all the concepts in the ontology provides a description of the whole Textile/Clothing sector in terms of possible business processes, involved actors, activities to be performed, goods to be produced and documents exchanged. However, some of the possible combinations are simply not allowed by the reality of the

[10] Structured representations of the business processes that are currently supported by NIMBLE can be fetched from: http://nimble-staging.salzburgresearch.at/business-process/content

business: the Actor “Logistic Operator” is not involved in the “Production” process at all. This knowledge is stored inside the Context Ontology and used by the TPD to create correct rules for specific business process, avoiding un-real business scenarios.

In the case of the ontology set up for NIMBLE in the Textile/Clothing sector, for example, the taxonomy (represented only partially for reasons of space) is the following:



- Activity
  - 'Customized production'
  - Delivery
  - Despatch
  - Invoicing
  - 'Party information transmission'
  - 'Product selection'
  - Purchase
  - 'Stock offer'
  - Subcontracting
- Actor
  - 'e-Service Provider'
  - 'Logistic Operator'
  - Producer
  - Retailer
  - Subcontractor
- 'Commercial Good'
  - 'accessory product-service'
  - fabric
  - garment
  - yarn
- conceptMetaClass
- Document
  - 'Commission Order'
  - 'Commission Order Change'
  - 'Commission Order Response'
  - 'Commission Order Status Report'
  - 'Despatch Advice'
  - 'Despatch Request'
  - Forecast
  - 'Information Request'
  - Inventory
  - Invoice
  - Offer
  - 'Offer Request'
  - 'Party Information'
  - 'Price List - Catalogue'
  - 'Purchase Order'
  - 'Purchase Order Change'
  - 'Purchase Order Response'
  - 'Purchase Order Status Report'
  - 'Quality Report'
  - 'Receiving Advice'
  - 'Stock Offer'
  - 'Stock Offer Change'
  - 'Stock Offer Status'
  - 'Technical Sheet'
- owl:Class
- Process
  - 'Party information transmission'
  - Production
  - Stock

Figure 16: Taxonomy of the NIMBLE ontology for Textile/Clothing sector

7.2 Architecture of the TPD

The general architecture of the TPD is shown in the following figure:

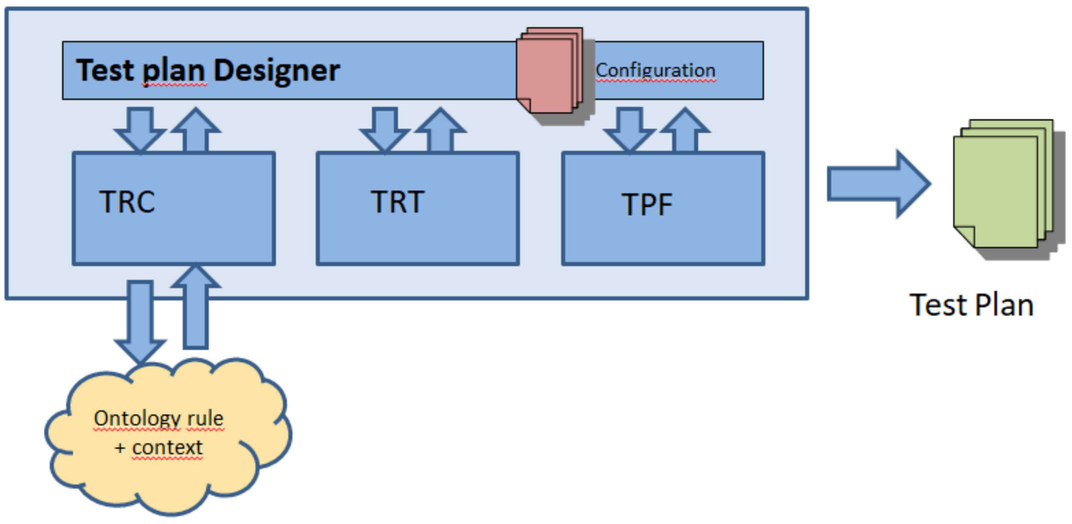


Figure 17: The general TPD architecture

As depicted in **Figure 17**, the main components are:

1. **Test Plan Designer (TPD)**: main component that acts as the manager for the creation of test plans, controlling the operation of the other components.

2. **Ontology of the rules + Ontology of the context**. Ontologies that contain the abstract business rules defined and allow specifying the business context for which the rules are valid.

3. **Test Rule Collector (TRC)**: component that collects from the rule ontology, according to the settings received as input from the TPD and the user configuration, the abstract rules to be used for test plans.

4. **Test Rule Translator (TRT)**: component that takes the (abstract) rules previously collected and transforms them into concrete and executable rules to be included in the test plan.

5. **Test Plan Formatter (TPF)**: using the concrete rules, the TPF sets and configures the files for the management of the test plan. These are the files that are finally used by TeBES to set up, configure and run the tests.

All these components rely on configuration files that allow specifying some parameters.

7.2.1 Configurations

The functioning of the TPD and its components is regulated by a set of configuration files that determine the running parameters.

There are in particular two types of configuration files:

- one that adjusts the operation of the application;
- one that sets the mapping tables necessary to manage the translation of the rules from abstract into concrete (see later).

Application configuration

The tdp.properties file (which is unique) concerns the general operation of the application and sets:

- the ontology to be loaded, all the relative files to be imported into the ontology and the prefixes to be used to collect the rules;
- the indication of the various areas that form the possible context in which a rule can be valid;
- the directories where to save the output files;
- the empty templates to use and fill in to create files for the test plan. These templates are used to define and set the basic structure (still empty and to be compiled) of the output files. They are loaded and completed with the information collected by the ontology and then transformed;
- the "objects" (with the relative context) that will be tested by TeBES (see next section on Input) through the application of the test plan generated by the Test Plan Designer.

Translation configuration

In the abstract rules, some parameters appear that must be transformed into values in order to have the concrete rules to be applied.

For example, an abstract rule for testing if the version of the template document is as expected could be in the form:

not ($ version) or ($ version = $ lastVersion)

Where the '$' symbol identifies a parameter whose value depends on the standard or format that is used and therefore to be tested.

To be able to translate abstract rules into machine executable ones, it is therefore necessary to set, for each standard, a mapping table. This table allows the translation of abstract rules into executable rules specific for the adopted standard by mapping the generic parameter "$version" with an item defined in the standard.

These parameter mappings are contained in a file that is named " *NameStandard.properties* ", for example for the UBL standard there will be the *UBL.properties* file or for the ModaML the *ModaML.properties* file will exist.

The content of one of these files is in the form:

```
# translation of the rules in ModaML #
#############################################

#STANDARD
name = Moda-ML

# NAMESPACE
# are the prefixes that must appear in the schematron files
# to be able to use correctly the rules XPath
Prefix1=  mml
NS1 = urn:moda-ml:repository:schema:TEXOrder

# STANDARD GENERAL VARIABLES
$ id                    = msgID
$ version               = @version
$ lastVersion           = '2013-1'
$ price                 = price
$ useProfile            = @ useProfile
$ requDocType           = child::*/child::refDoc[@docType = 'ORD']
$ docRefType            = refDoc / @ docType
$ ordType              = 'ORD'

...
....
```

Taking information from the mapping file specific for MODA-ML, the generic rules that test if the version adopted is the correct one become:

not (@version) or (@version = “2013-1”)

This rule, in the XML Schema Language, tests if the document attribute “@version” does not exist or its value has to be equal to “2013-1”.

Depending on the “tr:hasPrescriptionLevel” item in the generic rule, the test bed returns a fatal error or a simple warning if this rule is not satisfied.

7.2.2 Specifying the scenario for which to create Test Plans

One of the main inputs of the Test Plan Designer specifies which are the "objects" or business items that must be tested by TeBES, and for which business context this test must be prepared.

We define "Entity" as the object we want to test, and “Context” as the description of the elements within which the Entity is to be tested. The Context effectively represents the context of use of the entity.

For example, we may want to test an “Order” business document thus we assume it is the Entity. This document can clearly be adopted in different situations (Textile, Appliances, Furniture for example), thus requiring different checks and controls based on its use. The context in turn is composed of different scopes (Scope) that represent its various aspects.

We define then:

Entity = object to be tested

Context = context in which the Entity is tested

Test scenario = Entity + Context

In Nimble, a “test scenario” can comprise a maximum of 5 main “scopes”, which are what we have identified as the main elements in a business document exchange scenario. These areas are: Commercial Good (Product), Process, Activity, Document, Actor.

Commercial Good: it is the object of the business transaction (for example a *fabric* or a *yarn*), of the exchange of information, or anyway involved in the test scenario.

Process: it is the business process in progress (for example the *Textile Supply* process), which is in turn made up of various activities.

Activity: it is a sub-component of the process and specifies an action that must be carried out (for example, sending a catalogue or another document).

Document: the document that is exchanged in the activities (for example Order document or Catalogue)

Actor(s): the subjects that act within the processes and activities, exchanging documents or performing other actions. For example: Supplier, Subcontractor, Logistics manager.

A complete test scenario therefore may consist of:

Entity=Order Document

Context=Product, Process, Activity and Actor.

Specifying to the TPD which test plan to create means mainly to specify, within the tdp.properties file or through the methods provided (with direct function call):

- one and only one entity to be tested;
- a single context formed by the remaining areas.

The ontology is therefore the means by which the Entity and Contexts, which are to be considered in creating the test plans, are identified.

The terms to be used must be taken from the context ontology that has been specified in the configuration file.

The test scenario in the configuration file is indicated by specifying 2 variables, so it has

- a key-value pair that specifies the Entity
- a key-value pair that specifies the Context

These two pairs are connected to each other by an index: then the key pair is always of type *EntityN, ContextN*. By this way, in the configuration file, it is possible to indicate a long list of test scenarios at once, for each of which the corresponding test plan will be created.

For example, if you use the *tpd.properties* file as input, to specify the tests to be created, it is possible to write the file as following:

```
Entity1  = pad: ModaMLTextilePurchaseOrder
Context1  = pad: ModaMLFabricSupply; pad: ProducerActorType ;

Entity 2 = ......
Context 2 = ......
```

As you can see, you must always use the namespace prefixes, as also indicated in the configuration file. In the case of the value of the context, its scopes, if there are more than one, must be separated by ";".

The above example requests basically to create firstly a test plan to test the Entity `ModaMLTextilePurchaseOrder` (which represents the document of "Purchase order of fabrics" for the Moda-ML standard), in a context formed by:

`ModaMLFabricSupply`: it indicates that we are in a process of supplying fabrics.

`ModaMLRetailOrganisation`: it indicates that the reference actor is a retail company.

This is followed by the request to create a subsequent test plan. In fact, in the configuration file you can specify, one after the other, several *EntityN*, *ContextN* pairs for which to create the test plans. The TPD will consider them one after the other, creating for each the necessary files, differentiating them and naming them according to the entities indicated.

7.2.3 Create a test plan

The creation of test plans consists of the production of a set of files that allow the verification of the abstract business rules instantiated in a specific context.

Having defined all elements that are involved in the scenario of creating a test plan for a specific context, we can look at how the system works.

When launched, the TPD looks at the configuration file (or at the incoming arguments) to:

- instantiate information about the specific context and item to be checked;

- prepare the common environment with all the relevant data available for the other modules.

When this process is complete, TPD calls the TRC asking for a collection of generic rules that are valid for the entity and context previously described. TRC interrogates different ontologies to get all the necessary information together and provides these data to the TPD.

After this action is completed, TPD asks the TRT to translate the abstract rules into specific (and executable) rules. TRT loads the mapping file for a specific standard and performs the translation, providing the TPD with a set of executable rules.

In the end, the TPF puts all the data together and prepares the necessary files, which are:

- a schematron file (.sch), which includes the business rules translated into schematron executable code;
- an XML file that contains the Test Assertions, which collect and encapsulate the rules implemented in the schematron files and some meta-data for their use;
- an XML file that describes the Test Plan, which organizes and sorts all the Test Assertions.

The file that describes the Test Plan is the main one: it is used by TeBES for orchestrating the execution of the tests, and in cascade recalls the other files.

The TPD, in addition to creating the files for the execution of the tests, produces a text file (eg. Document_ModaML_1_ListOfUntranslatedRules.txt) in which it indicates the abstract business rules that were collected but not translated and therefore not included in the Test Plan, indicating the motivations and the occurring problems (e.g. because the map file prepared for the standards are not complete and lacks the definition of some parameters), allowing the further improvement in the mapping file.

7.2.4 Development

The TDP application was developed in the Java language. For each of the components of the application a specific *.java* file has been created, so the files are:

- TPDesigner.java
- TPRuleCollector.java
- TPRuleTranslator.java
- TPFormatter.java

In addition to these modules, one has been developed for the management of the set up and configuration of the application, and for the setting of all the parameters that regulate the operation.

Specific data structures (beans) were created to represent two fundamental elements for the creation of test plans:

- Entity: is a data structure that specifies both the entity for which the test plan is being created and the context for which the test plan is defined.
- Rule: is a data structure that describes a rule to be included in the test plan to be verified. It also contains its abstract formulation.

The structure "Rule" substantially reflects the information and the model defined in the rule ontologies. When the collector has to collect the rules (defined in an abstract way) to create a specific Test Plan, it executes SPARQL queries on the Rules ontology, and it organizes the extracted information using the Rule structure. This structure is then passed to the TPRuleTranslator, which translates and produces the schematron code.

For example, in order to select rules related to the test scenario

```
Entity1 = pad: ModaMLTextilePurchaseOrder
Context1 = pad: ModaMLFabricSupply; pad: ProducerActorType; goods: Fabric
```

The following SPARQL query is executed:

```
PREFIX  pad:  <http://www.moda-ml.org/moda-ml/Ontologies/Moda-
ML/ModaML-ProcActDoc.owl#>
PREFIX  rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  owl:  <http://www.w3.org/2002/07/owl#>
PREFIX  xsd:  <http://www.w3.org/2001/XMLSchema#>
PREFIX  goods: <http://www.moda-ml.org/moda-ml/Ontologies/Moda-
ML/CommercialGood.owl#>
PREFIX  rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX  tr:   <http://www.moda-ml.org/moda-ml/Ontologies/Moda-
ML/TestRules.owl#>
PREFIX  gpadm: <http://www.moda-ml.org/moda-ml/Ontologies/Moda-
ML/GPADModel.owl#>

SELECT DISTINCT  ?x ?a ?b ?c ?d ?e ?f ?g ?h ?i ?l ?m ?n ?o
WHERE
  { pad:ModaMLTextilePurchaseOrder (rdf:type)*/(rdfs:subClassOf)* ?y2
.
    ?x  tr:isAppliedTo  ?y2 ;
        tr:hasScope     ?w0 .
    pad:ModaMLFabricSupply (rdf:type)*/(rdfs:subClassOf)* ?w0 .
    ?x  tr:hasScope  ?w1 .
    pad:ProducerActorType (rdf:type)*/(rdfs:subClassOf)* ?w1 .
    ?x  tr:hasScope  ?w2 .
    goods:Fabric (rdf:type)*/(rdfs:subClassOf)* ?w2
    FILTER NOT EXISTS { ?x  tr:hasScope  ?z0 .
                        ?z0 (rdf:type)*/(rdfs:subClassOf)*
gpadm:Activity
                      }
    FILTER NOT EXISTS { ?x  tr:hasScope  ?z1 .
                        ?z1 (rdf:type)*/(rdfs:subClassOf)*
gpadm:Document
                      }
    OPTIONAL
```

```
      { ?x  rdfs:label  ?a }
    OPTIONAL
      { ?x  rdfs:comment  ?b }
    OPTIONAL
      { ?x  tr:insertedBy  ?c }
    OPTIONAL
      { ?x  rdfs:isDefinedBy  ?d }
    OPTIONAL
      { ?x  tr:hasInsertDate  ?e }
    OPTIONAL
      { ?x  owl:versionInfo  ?f }
    OPTIONAL
      { ?x  owl:deprecated  ?g }
    OPTIONAL
      { ?x  tr:hasPrescriptionLevel  ?h }
    OPTIONAL
      { ?x  tr:hasPriority  ?i }
    OPTIONAL
      { ?x  tr:useTechnology  ?l }
    OPTIONAL
      { ?x  tr:hasAbstractRule  ?m }
    OPTIONAL
      { ?x  tr:hasImplementation  ?n }
    OPTIONAL
      { ?x  tr:hasAbstractContext  ?o }
  }
```

8 Conclusions

Interoperability, quick response to orders and customisation of products are the key factors for companies that want to keep their market. This is particularly true for competitive markets as high quality and luxury markets.

Interoperability among different systems is a crucial element for supporting smart B2B collaborations in a dynamic network of companies that cooperates within the same supply chain.

The need for dynamic interactions is forcing companies to abandon the old model of relations, made by direct relations among the entrepreneurs, and to adopt the collaborative model as a paradigm for the business.

NIMBLE moves strongly in this direction by supporting companies in implementing B2B collaborations with few “clicks” and easing the setting up of the interaction among productive systems. However, the complexity of the companies requires also the possibility to exchange data and documents in an automatic way, thanks to the adoption of shared and free standards for document templates and on agreeing on the description of the business processes to be supported. The experience made by several organizations, ENEA among them, in supporting companies during the process of implementing B2B collaborations by adopting standards proves the claim that a test bed system reduces the efforts required by companies and lowers the risks, and the costs, related to this process.

Implementing a test bed system such as TeBES in NIMBLE can ease the process of productively using the platform, reducing the initial “friction” that structured companies (with a cumbersome collection of management software solution) can suffer at start.

TeBES may in fact play different roles in the NIMBLE ecosystem: by supporting the developers of software improvements (willing to check the conformance of the new modules with already agreed specifications) as well as NIMBLE’s customers (aiming at checking the conformance of the external in/outcoming messages from commercial partners before launching a serious and massive B2B collaboration).

Appendix A: The NIMBLE-Order Test Plan in XML

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="http://winter.bologna.enea.it/TeBES2_Artifacts/xsl/TeBES_TestPlan.xsl" ?>
<!—
   Test Plan by TPDesigner for the TeBES Platform
   For ORDER in NIMBLE
   20180524-12:00:00
   Version=1.0
-->
<tebes:TestPlan lg="en" xmlns:tebes=http://www.ubl-italia.org/TeBES
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance>
            <tebes:TestPlanHeader>
                        <!—
                        The "datetime" elements are related to the creation and last modify datetime of file.
                        The state element can take 2 values: "draft" or "final"
            -->
                        <tebes:Id>1</tebes:Id>
                        <tebes:Name>NIMBLE-ORDER</tebes:Name>
                        <tebes:State>draft</tebes:State>
                        <tebes:Description lg="en">Test Plan created by TPD, defined and used for UBL-NIMBLE
ORDER.</tebes:Description>
                        <tebes:Description lg="it">Test Plan creato dal TPD, definito e utilizzato per UBL-NIMBLE
ORDER.</tebes:Description>
                        <tebes:UserId>0</tebes:UserId>
                        <tebes:CreationDatetime>2018-05-24-T12:00:00Z</tebes:CreationDatetime>
                        <tebes:LastUpdateDatetime>2018-05-24-T12:00:00Z</tebes:LastUpdateDatetime>
            </tebes:TestPlanHeader>

            <tebes:TestActionList>
                                    <tebes:TestAction id="TA-1" number="1">
                                    <tebes:ActionName>TA1-TP1-NIMBLE</tebes:ActionName>
                                                            <tebes:ActionDescription lg="en">Business rules on UBL-NIMBLE
            ORDER (applied through XSD Schema Validation).</tebes:ActionDescription>
                                                            <tebes:ActionDescription lg="it">Business rule per UBL-NIMBLE
            ORDER (utilizzando validazione XML Schema).</tebes:ActionDescription>
                                                            <tebes:Test lg="taml" type="TestAssertion"
            skipPrerequisites="true"
            location="http://winter.bologna.enea.it/TeBES2_Artifacts/testsuites//NIMBLE/TS-001_NIMBLE/TC-001-
            XMLSchema-UBLNimble-1.xml">NIMBLE-ORDER-1</tebes:Test>
                                    <tebes:Inputs>
                                                <tebes:Input>
                                                            <tebes:Name>IN-1_TA-1_TPD-UBL-NIMBLE</tebes:Name
                                                            <tebes:InputDescription lg="en">Send UBL-NIMBLE
            ORDER.</tebes:InputDescription>
                                                            <tebes:InputDescription lg="it">Invia UBL-NIMBLE
            ORDER.</tebes:InputDescription>
                                                            <tebes:SUT interaction="website" type="document" lg="xml"
                                                                                    fileIdRef="FI-1_document-xml-website-IN-
            1_TA-1_TPD-UBL-NIMBLE"> </tebes:SUT>
                                                            <tebes:GUI reaction="upload">
                                                                        <tebes:GUIDescription lg="en">Upload of Input with
            fileIdRef = FI-1_document-xml-website-IN-1_TA-1_TPD-UBL-NIMBLE-20180524-
            12:00:00.</tebes:GUIDescription>
                                                                        <tebes:GUIDescription lg="it">Upload dell'Input con
            fileIdRef = FI-1_document-xml-website-IN-1_TA-1_TPD-UBL-NIMBLE-20180524-
            12:00:00.</tebes:GUIDescription>
                                                            </tebes:GUI>
                                                </tebes:Input>
```

```
                </tebes:Inputs>
        </tebes:TestAction>
                <!--
                        Each tebes:TestAction MUST have a @number attribute used to enumerate
every action.
                        The N actions will be execute from 1 to N, sequentially,
                        excepted different specification expressed through the
tebes:TestPlan/tebes:Choreography element

                        Each tebes:Test element MUST have a @type attribute used with one and
only one of these values:
                                - "TestAssertion"
                                - "TestCase"
                                - "TestSuite"
                -->

</tebes:TestActionList>
        <tebes:Choreography/>

</tebes:TestPlan>
```

Appendix B: The NIMBLE Test case with references to NIMBLE-Order

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="http://winter.bologna.enea.it/TeBES2_Artifacts/xsl/TeBES_TestSuite.xsl" ?>
<!--
  Created by TPDesigner
  For ORDER in NIMBLE
  29-05-2018
  Version=1.0
-->
<taml:testAssertionSet
  setname="Test Case for ORDER Business Rules validation" tebes:version="0.1"
  xsi:schemaLocation="http://docs.oasis-open.org/ns/tag/taml-201002/ http://docs.oasis-
open.org/tag/taml/v1.0/cs02/xsd/testAssertionMarkupLanguage.xsd"
  xmlns:taml="http://docs.oasis-open.org/ns/tag/taml-201002/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:tebes="http://www.ubl-italia.org/TeBES" setid="TA-NIMBLE-1">

  <taml:common>
    <taml:namespaces
      xmlns:nimbleIn="https://raw.githubusercontent.com/nimble-platform/common/master/data-model/ubl-
data-model/src/main/resources/NIMBLE-UBL-2.1/maindoc/UBL-Order-2.1.xsd"/>
    <taml:namespaces
      xmlns:nimbleParty="http://winter.bologna.enea.it/TeBES2_Artifacts/xmlschemas/NIMBLE/nimple-party-
schema.xsd"/>
  </taml:common>
  <taml:testAssertion id="NIMBLE-ORDER-1" name="NIMBLE-1" tebes:isNotePresent="false">

    <taml:description>Business rules on UBL-NIMBLE ORDER (applied through XSD Schema
validation).</taml:description>

    <taml:normativeSource>
      <taml:comment/>
      <taml:refSourceItem/>
      <taml:textSourceItem/>
    </taml:normativeSource>


    <!-- <taml:var vname="NIMBLE-1"
vtype="http://purl.oclc.org/dsdl/schematron">http://winter.bologna.enea.it/tebes-
artifacts/schematrons/NIMBLE/$FILENAME</taml:var>
    -->
    <taml:var vname="NIMBLE-1" vtype="http://www.w3.org/TR/xpath20/">count(//nimbleIn:Order) ge
1</taml:var>
    <!-- XML document, target of Test -->
    <taml:target type="document" lg="xml">document</taml:target>


    <taml:predicate>NIMBLE-1</taml:predicate>

    <taml:prescription level="mandatory"/>

    <taml:report label="pass" message="UBL-NIMBLE ORDER Business Rules are satisfied.">The file is valid against
UBL-NIMBLE ORDER Business Rules.</taml:report>
    <taml:report label="fail"
      message="One or more UBL-NIMBLE ORDER Business Rules are NOT satisfied.">The file is NOT valid against
UBL-NIMBLE ORDER Business Rules.</taml:report>
    <taml:report label="notQualified"
      message="Not qualified to perform the UBL-NIMBLE ORDER Business Rules validation">The file is not
```

```
qualified to perform the UBL-NIMBLE ORDER Business Rules validation.</taml:report>
  </taml:testAssertion>

  <taml:testAssertion id="NIMBLE-PARTY-1" name="NIMBLE-PARTY-1" tebes:isNotePresent="false">

    <taml:description>XSD Validation of the NIMBLE Party profile (applied through XSD Schema
validation).</taml:description>

    <taml:normativeSource>
      <taml:comment/>
      <taml:refSourceItem/>
      <taml:textSourceItem/>
    </taml:normativeSource>

    <taml:var vname="NIMBLE-PARTY-1" vtype="http://www.w3.org/TR/xpath20/">count(//nimbleParty:Party) ge
1</taml:var>
    <!-- XML document, target of Test -->
    <taml:target type="document" lg="xml">document</taml:target>
<taml:predicate>NIMBLE-PARTY-1</taml:predicate>
    <taml:prescription level="mandatory"/>
    <taml:report label="pass" message="Valid NIMBLE Party document.">The file is valid against NIMBLE Party
document.</taml:report>
    <taml:report label="fail"
      message="The gile is not a valid NIMBLE Party document.">The file is NOT valid against NIMBLE Party
template.</taml:report>
    <taml:report label="notQualified"
      message="Not qualified to perform the validation">The file is not qualified to perform the NIMBLE Party
validation.</taml:report>
  </taml:testAssertion>

</taml:testAssertionSet>
```