Addressing Testing Objectives for e-Learning

Antonia Bertolino, Eda Marchetti, Andrea Polini
ISTI-CNR, Pisa Italy

Hans-Christian Herrmann
Institut für Wissensmedien, Universität Koblenz-Landau, Koblenz, Germany

Abstract

This paper outlines different objectives for testing, and surveys some relevant techniques and tools used to address some of those objectives. The overview is conducted in the perspective of transferring this body of knowledge to the emerging domain of technology enhanced learning. Some specific techniques under investigation within the frame of the European FP6 project TELCERT are summarized.

1 Introduction

Recent years have seen a growing interest into the e-Learning domain, thanks also to the rapid advances of the Information and Communication Technology (ICT), which have made available new powerful and low-cost means to provide on-line and interactive training services. The spreading out of the Internet, of wireless and mobile devices, as well as the widespread adoption of open systems specifications and standards have set up a wealth of new capabilities and application opportunities for the future of Technology-Enhanced Learning (TEL).

As better discussed in Section 2, a learning system consists of the e-Learning platform (the LMS) and the "courses" themselves, that form the learning content. From both the producer's and the user's sides a urging demand comes for high quality, flexibility and interoperability of platforms and contents, and all these qualities are best built on top of open system specifications. Indeed, both the medium and the content are the result of high investments from the respective producers, who clearly want to invest in a future safe product. Considering the content, due to the ubiquity and heterogeneity of consumers, this requires independency from the current platform and prevention of lock-in effects by proprietary formats: ideally a product (i.e., the content) should be usable on any platform, and the vice versa. There is also a vivid interest of the users themselves (teaching institutions) in interoperability. Users want to move their content between systems. They want to aggregate content from different sources and also probably in the future combine system components from different vendors. Hence they want to be assured of the interoperability before buying the product.

Speaking in general to achieve such characteristics rigorous and controlled development processes have to be put in place. But of course, it is not thinkable to impose a strictly regulated development process. Development of learning systems actually is a complex and articulated process carried on by different institutions/stakeholders, in which standardization certainly plays a key role. However, although it may be practicable to make some general recommendations for content development or give some sort of best practices, the actual development and control is always in the hands of the developers (of content or medium) with their own interests, standards and practices.

What could be done, and is in fact one of the main goals of the recently started TELCERT project [13], is to pursue the routine adoption of conformance testing as a means to facilitate/validate interoperability. Not only a standardized conformance test process for e-Learning can directly check whether a specific content instance is compliant with the agreed terminology and syntactical rules, and/or whether the LMS behavior conforms to the specified sequencing of events. Also, the introduction of a conformance testing program brings in as a side effect to raise the producers' concern for standard specifications (as customers will naturally tend to prefer conformant products).

But exactly because of the large economical and political implications behind conformance validation (which subsequently may also lead to certification), the adopted test suites should be the product of neutral, accredited and authoritative body, institution or service.
The above mentioned TELCERT project is a research and development project under the European Union's 6th Framework Programme. Led by a consortium of e-Learning providers, research and industry organizations, TELCERT aims at developing innovative software testing and conformance systems to assure interoperability in e-Learning content and technology.
The TELCERT work programme includes the following tasks:

- Identifying the 'state of the art' in testing and conformance
- Test suite research and development
- Creating application profiling and content re-engineering tools
- Developing the TELCERT Test Suite and conformance programme

In line with the above tasks, this working paper will first discuss the need for interoperability in the e-Learning domain. Then we overview the field of software testing. Testing techniques can be applied at different stages and with different objectives: we overview these different test objectives and outline existing techniques and tools. This overview is done with a perspective on the usage of such techniques and tools in the e-Learning domain, in which testing and specifically conformance testing is still preliminary.

2 Development processes for e-Learning systems

The practical adoption of eLearning involves different types of resources: human resources, knowledge resources and technical resources. The human resources, typically lecturers, students and technical administrators are not subject to the technology based testing and will therefore not be discussed here further.

The basic technical resource that the testing will concentrate on is the *learning management system* (LMS). It is the central mediator between all other resources and typically not exchangeable like hardware since most suppliers of LMS provide some added value to their product despite the basic features found in all LMS. Also LMS are not as technologically mature and standardized like e.g. networking technology. The LMS is one critical eLearning component the testing in the TELCERT project is focusing on.

The knowledge resources involved in the learning process are the courses themselves and all the content integrated in the courses. This content can be of many digital formats like texts, diagram, collections of ideas, discussions, simulation programs, spreadsheets, etc. Some of the content is prepared in advance by the lecturer, but other content might be worked out by the course participants while a course is taking place. All the content is organized in e-Learning courses which make up a complete learning unit. Therefore the courses are the second important eLearning component the TELCERT testing is concentrating on.

2.1 Structure of E-Learning Systems

An eLearning system is typically based on internet technology and most information is delivered in form of web pages or other digitalized materials from the LMS to the persons involved in the learning process. The learners and teachers work on personal computers at home, at the teaching institution or some other learning venue with internet access. Recently also mobile devices may be used to support learning in any place and at any time. The LMS consist of a minimum of three basic components from the functionality point of view: the web server front-end, a user management component and a course management component.
The *front-end component* receives requests from the users just like any web server and processes it. Processing includes checking and updating the user authentication and settings, requesting and

updating content from the course management component, assembling the response into a web page and delivering it back to the user.
The *user management system* that must authenticate the user and provides personal and system wide information about the registered users. Most of this information will be based on predefined roles of the user in the system (e.g. administrator) or in a course (e.g. designer, lecturer, student). But further information can be considered like personal settings set by the user or learner profiles maintained at the same or a remote location. The information known about the user is used to provide personalized content, but also to restrict the user to content that is considered to suit his personal knowledge and information needs.
Several standards and specifications exist to handle the information about the users. The user information can even be distributed on more than one system, e.g.

- a local relational database storing LMS specific information, authenticating a user with an
- institution wide authentication server accessed via LDAP protocol and
- getting information on a learners current knowledge from some remote server maintaining a learner profile or portfolio.

The *course management component* maintains the courses registered on the system. A course is made up of several parts and relations between those parts. It basically consists of a course structure defined by the lecturer and content provided by the lecturer and the learners before and during the course realization.
But in real life a course is much more complex and many other information can be needful and necessary to describe what makes up a course; this includes alternative course structures, dependencies between elements, relations between learners, tools that are used for activities, a workflow adopted by the learners, predefined learning objectives, etc. Normally not all this can be reflected in a technical system. Limitations are given by the capabilities of the LMS and by the power of the specification used to express the course. Courses may also exhibit a dynamic behavior, providing or denying access to learning objects depending on the learner's achievements.

2.2 Structure of e-Learning Courses

A course is generally not held at only one time, but distributed over several lessons and other *units of learning*. The course designer chooses an adequate course structure of smaller units of learning that can be arranged in a reasonable ordering for the purpose of imparting knowledge. The ordering can be linear; but often an ordering made up of forks, parallel sessions or other alternative orderings can be more adequate. It depends on the LMS and the specifications applied what course structures can be expressed in an eLearning course.

A unit of learning is defined by the intended learning objective persuit and a set of activities that are planned to take place in order to achieve those objectives. The activities make use of learning objects that are also part of the unit of learning and were supplied by the teacher or possibly the students.

The learning object can be in any type of digitalized form when included in an eLearning course. A distinction between static and dynamic objects must be made regarding the handling of object in the LMS. Static objects are not modified during the realization of the course and only read or played by the course participants, i.e. text documents, images, diagrams, film sequences and even programs. Since they are not modified, they can easily be reused in other courses or used at different places of one course. Therefore it can be reasonable to technically place those objects outside a course and just make a reference to them in a course. The object is then loaded from the original storage place

(e.g. media library or external server) at the moment a user requests the object, but invisible to the user.

Dynamic learning objects are modified during a course by the students or the teacher. They are initially started with only some basic data and worked on during the course. This can be a discussion forum where student and teachers communicate, spreadsheets where students can collect and summarize their results or quizzes where students must demonstrate their current knowledge and understanding of the subject. Dynamic objects can also be reused if a mechanism exists to initialize them before starting a new course. So for a dynamic learning object there is a basic instance and some course or even user specific instance with the information provided during the realization of the course. Static and dynamic objects can be closely intertwined in a course. For example learners may dynamically create annotations for static learning objects.

The users involved in an eLearning course perform different tasks within the course. The restrictions and permissions given to each user are assigned through predefined roles, typically the teacher and students. The lecturer normally creates the course and has full control over it, like having access to all information the current work and data contributed during the course and granting students access to resources. Students can enroll into the course and work on the course according to the restriction or recommendations given by the teacher in form of the course structure. Other roles are possible, e.g. tutors have more access to resources than student but without full control over the course.

3 Standardization issues and profiling

e-Learning has experienced several attempts to establish standards in the last years. But standardization can be considered a longsome process. The IETF RFC 2026 outlines the process of standardization to be based on a specification that is developed in several iterations of review and revision; then it must be approved by an appropriate body before finally being published[1]. The goals of the standardization process according to the RFC 2026 are declared to be "technical excellence; prior implementation and testing; clear, concise, and easily understood documentation; openness and fairness; and timeliness." This complicated standardization process is intended to produce a specification of high technical quality, taking into account the interests of all affected parties, and still establish a "widespread community consensus".

Standardization bodies are usually established on national level (DIN[2], ANSI[3], IEEE[4]) or international level (ISO[5], CEN[6]). This ensures them a wide acceptance, which is essential for the adoption of the standards. But especially in the area of internet technology other organizations like IETF[7] and W3C[8] have also established. They are not officially accredited as standardization bodies and publish technical *specifications* instead of standards. But their technical specifications are highly accepted in the respective communities and can be considered *de facto standards* that are not published by official standardization bodies but play almost the same role as the official standards. It is not unusual to incorporate de facto standards in later developed official standards.

---

[1] S. Brader: "RFC 2026 - Internet Standards Process – Revision 3". Harvard University, October 1996, http://www.ietf.org/rfc/rfc2026.txt.
[2] DIN Deutsches Institut für Normung e.V. (http://www.din.de/).
[3] ANSI - American National Standards Institute (http://www.ansi.org/).
[4] IEEE - Institute of Electrical and Electronics Engineers (http://www.ieee.org/).
[5] ISO - International Organization for Standardization (http://www.iso.ch/).
[6] CEN - European Committee for Standardization (http://www.cenorm.be/).
[7] IETF - Internet Engineering Task Force (http://www.ietf.org/).
[8] W3C - World Wide Web Consortium (http://www.w3.org/).

If the field of e-Learning quite some technical specifications have been published, but only few standards. Although most standardization bodies are currently working on e-Learning standards, the lengthy standardization process has not yet reached the publication phase in most organizations. Some of the standards relevant for eLearning are IEEE LOM[9] (metadata) and IEEE CMI[10] (course description). In the mean time specifications have taken the place of standards and are currently used by the e-Learning practitioners and often established as de facto standards. A collection of complementary e-Learning specifications are published and maintained by the IMS Global Learning Consortium[11], including specifications like LIP (Learner information Package), LD (Learning Design), QTI (Questions & Test Interoperability) and CP (Content Packaging). IMS specifications are also highly accepted, by both suppliers and users. Other important specifications for eLearning are OKI OSID[12], AICC Guidelines[13], but also non-eLearning specific specification like XML[14] or LDAP[15] affect eLearning.

Standards and specifications claim to cover the necessities of a large number of related communities. This means that with respect to the needs of one specific community they most probably include elements that are not needed by all communities, or allow a larger range of values than needed, or don't allow values needed by one community, or they might even miss elements that are used by community. Which elements are included in a specification and the details on the data type and usage of these elements are given in the specification's information model. An XML binding is often used to map the abstract information elements to an XML representation. Finally a best practice guide is sometimes provided with recommendations on practical implementation issues.

The idea of *application profiling* is to provide a specialized implementation of one or more standards or specifications, tailored to the needs of specific community of practice. An application profile is an adapted implementation of a standard or specification, possibly completed with elements from other standards and specifications. It often comes together with guidelines specific for the targeted community. It can e.g. declare optional elements to be required, exclude elements of the base specification, set default values for elements, or declare terms to be included in a vocabulary.

Application profiles are commonly used and some popular examples are SCORM[16] (often wrongly declared to be a specification or de facto standard), e-GIF[17] and CanCore[18]. There are some advantages of developing an application profile on basis of an established standard rather than developing another new highly specific specification:

- Compliance: depending on whether the profile is restrictive or extensive with respect to the base specification, the resulting document instances remain read- or write-compliant[19] to the base specification.

---

[9] IEEE LOM - IEEE Learning Object Metadata (http://ltsc.ieee.org/wg12/).
[10] IEEE CMI – IEEE Computing Managed Instruction (http://ltsc.ieee.org/wg11/).
[11] IMS Global Learning Consortium (http://www.imsglobal.org/).
[12] OKI OSID – Open Knowledge Initiative's Open Service Interface Definitions (http://web.mit.edu/oki/specs/index.html).
[13] AICC – Aviation Industy CBT Committee (http://www.aicc.org/).
[14] XML - Extensible Markup Language (http://www.w3.org/XML/).
[15] LDAP – Lightweight Directory Access Protocol (http://www.ietf.org/rfc/rfc3377.txt).
[16] SCORM – Sharable Content Object Reference Model (http://www.adlnet.org/index.cfm?fuseaction=scormabt).
[17] E-GIF – UK e-Government Interoperability Framework (http://www.govtalk.gov.uk/egif/).
[18] CanCore - Canadian Core Learning Object Metadata Application Profile (http://www.cancore.ca/).
[19] An application profile can define restrictions or extension to the specification it is based on. If it is strictly restrictive the number of possible document instances is reduced and any document instance that complies with the application profile will also be compiant to the base specification. This is called read-compliance. If the application profile is strictly extensive then number of possible document instances is augmented and any document that complies with the base specification will be compliant to the application profile. This is called write-compliance. If an application is both restrictive and extensive, then no general conclusion on the compliance can be given.

- Maturity: standards and specifications run through a development process that ensures wide acceptance and technical maturity and practical relevance. Making use of an existing standard or specification reduces the work load and saves time.
- Interoperability: system and content can be used together with other systems and are not tied to only one proprietary system due to e.g. the encoding or protocols.
- Reusability: content can be reused in different system, settings and institutions.
- Durability: standards, once established, evolve; this can avoid obsolescence of systems and content.

But making use of specifications does not automatically guarantee that all these benefits are really accomplished. Adopting a specification must rather be considered as enabler and necessity to gain the stated benefits. Also specifications do not necessarily regulate all details and must leave some scope to the implementers for their specific needs. Several incompatible solutions are sometimes possible for the same task. For this reason the recommendations of the best practice guide should be followed for cases that are covered there.

4 Quality requirements for modern e-Learning

<<Koblenz + ISTI>> ~2 pages

The keyword today is **interoperability**, however this term has been overcharged [ID1]to mean related yet different things

Discussion of relevant needs and requirements

with regards to medium:

for instance:

Interoperability

Usability

…

With regards to contents:

Compatibility[ID2]

Conformance

….

5 Testing objectives

The same term *testing* can refer to many related yet different activities, all aimed at evaluating some property of a software system, and ultimately to increase our confidence in the "suitability" of the product to our objectives. Admittedly, the above sentence is rather vague, but it is so on purpose for trying to embrace into a statement the many widely different objectives that one may want to address by testing.

A coherent definition for testing comes from the IEEE Standard Glossary of Software Engineering Terminology [3], stating that testing is the "*process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component*". So, testing is carried on for evaluating some specified aspect or property, and the aspect being pursued establishes the objective for testing.

There exist many test techniques, and the one that is the most effective in each circumstance precisely depends on the aspect that has to be evaluated, as well as, of course, on the conditions under which the system is being evaluated. This is why distinguishing and explicitly establishing relevant test objectives is important to plan and manage the test process.

A useful comprehensive overview of test objectives can be found in ([9] Chapter 2). Test objectives are there called "test factors" and include, among others, Correctness, File integrity, Access control,

Compliance, Reliability, Ease of use, Maintainability, Portability, Performance. For example ([9] Fig. 2.7), if the test objective is "Maintainability" then the test strategy must ensure that the separate segments of the program will be associated to appropriate identifiers, and they point to the other segments that need to be changed concurrently with each; if the test objective is performance the test will have to check that the system achieves the performance acceptance criteria, and so on. No list of test objectives can ever be exhaustive, and from case to case different test objectives can be established. It is useful to reproduce here the *testing methodology cube* proposed by Perry ([9] Chapter 2): this cube is a three-dimensional work program (Figure 1). On a first dimension Perry puts the **test factors** (i.e., the test objectives), and this is considered the most important dimension. Along a second dimension he puts the **test strategy**, which is decomposed into the phases of the development process. For each phase, the relevant concerns have to be identified. Finally, the third dimension of the cube corresponds to the "**test tactics**", which embraces test criteria, test techniques and tools.

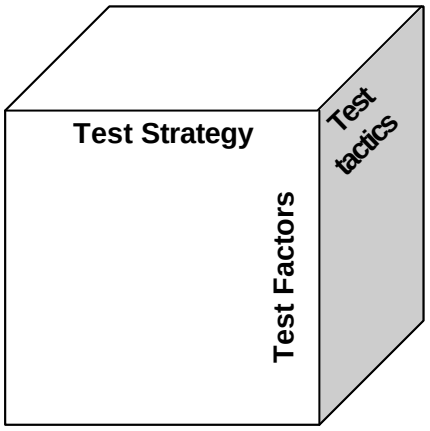

Figure 1 The test cube

Of course the decision of which aspects of the product should be evaluated depends strictly on the application domain. In safety critical applications, correctness becomes the predominant aspect, because no failures, even "small" ones, can be tolerated; instead, testing for reliability, i.e., prioritizing the detection of those failures which would be the most frequent in operation (see Section 5.5), has traditionally had a leading role in the telecommunications.

We will now consider in more detail some test objectives considered to be important for the e-Learning sector. For each test objective considered, we will also briefly address techniques and tools employed.

Another implicit assumption of the above IEEE definition is that testing presupposes the execution of the software, and in this sense testing is usually referred to as a "dynamic" evaluation activity. As such it is distinguished from other complementary checks, based on static analysis techniques, that can be used to evaluate a system or component. However, it is also not unusual to find the terminology "static testing" and "dynamic testing" to distinguish between the two different types of techniques, and this is in fact the attitude we have taken within TELCERT, where both kind of analyses are used (see Section 6).

5.1 Conformance testing

Description: One very important test objective is to validate that an implementation actually *does what it is expected to do*. Testing in this case is said *conformance testing*, and the objective is to assure that the product under test is built in accordance with established policies, procedures and standards. In particular, we focus on standard specifications covering either the software structure (e.g., the used identifiers, the multiplicity and types of arguments, etc..) or its behavior, i.e., the interactions with the user and between different components. The specification can be expressed

with different degrees of formalization, and under different levels of authority. While ensuring conformance is important for many reasons, it becomes mandatory where the specification is embedded into a normative (*de jure*) standard. The ultimate goal for establishing these standard specifications is always that of ensuring **interoperability** among products from different producers, and this is in fact the case for the e-Learning domain as we have discussed in the previous sections.
Notably, when considering in general software systems we can distinguish between different kinds of interaction that may be sought between the involved parties (be them humans or machines).
In the same way there also exist different types of conformance.

- **Data Conformance**: this is aimed to assure that "information" can be shared and exchanged, hence conformance testing consists here of checking that the syntax of data is in agreement with the established format in the standard. The testing techniques employed more properly belong to the techniques referred above to as "static testing". Data conformance testing from a syntactic point of view involves checking the format of data exchanged, the content required structure, and the interfaces. On a different layer would be checking the semantic conformance of data, i.e., that the syntactically coherent data actually are used by the different providers with the same meaning.
- **Behavior Conformance:** testing for behavior conformance implies to verify whether the observed behavior of the tested system conforms to the specification, and this is done by executing the system on a sample test suite and observing its I/O behavior (see Figure 2), i.e., by "dynamic testing" properly said. This test can be implemented and executed against different tests targets, including units, integrated units, and systems [9].

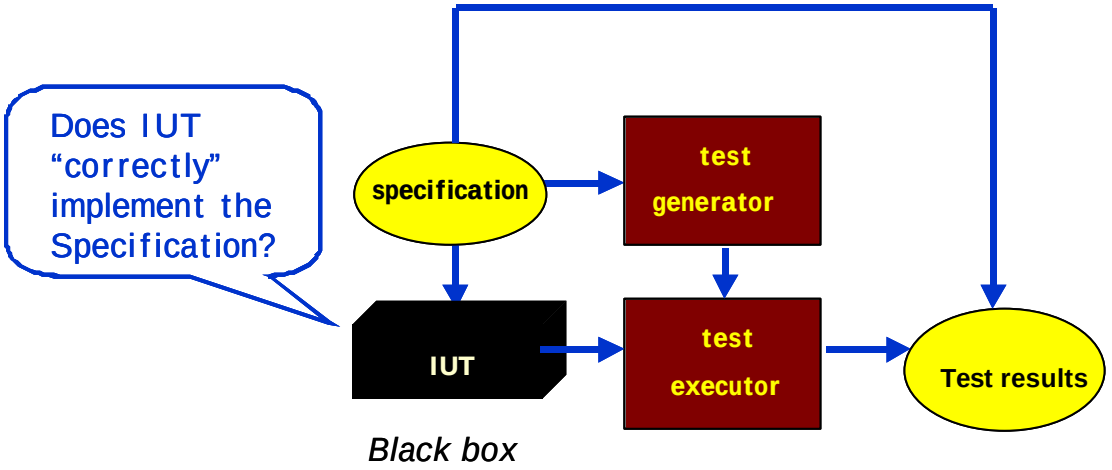

Figure 2 Conceptual schema for Conformance testing

Techniques and tools: The techniques and tools for conformance testing are quite different for the two contexts of data or behavior conformance. Data conformance can now rely on quite standard approaches which can directly provide exact response as to whether some instances of data are compliant with a syntax specification.
In a context in which the specification describes data that can be exchanged among software applications, the previous definition can be rephrased as the activity of assessing that the information enclosed in an "information instance" correctly contains the data that are described by the information model (that is specification of the information). The features of the eXtensible Markup Language (XML) has made possible the fast adopt affirmations of this language as the standard for data interchange. XML can be considered a meta-language that permits the definition of specific languages through the choice of specific tags.
For XML documents,, another important element that needs to be defined is the admissible structure of the document in terms of relations among the different tags defined by the language. These choices are the result of the agreement among different stakeholders that need to exchange

data or can be the result of the work of an authority that has the recognized "power" of defining the language elements.
Obviously the general agreement on the definition of a language does not guarantee in practice the interoperability of different software applications produced by different stakeholders, instead it is necessary to define instruments that make it possible to verify the conformity of a document instance written with reference to a specific language. Some specific techniques and tools based on the XML notation, currently under consideration within TELCERT for this purpose, are discussed in more detail in the next section.
Considering instead the medium, we are mostly interested in ascertaining the conformance of behavior. A quite mature theory for conformance testing has been developed in the last decade [14], [15], rooting on earlier results on equivalence testing of transition systems. This theory provides the foundations of test techniques developed in the domain of telecommunication protocols (expressed in SDL, LOTOS, or other specialized languages). Recently work has been carried on for extending such results to the testing of generic reactive systems (i.e., systems which behave by reacting to the environment stimuli). The main problem to overcome remains a problem of scalability.
Approaches for behavior conformance testing are based on a formalized notion for Conformance as follows [14], [15]. There is a formal specification, there is an implementation, and there are a set of properties (the Conformance Rules, or Conformance Clauses) that all conforming implementations should satisfy. The aim of a formal testing framework is to define a conformance relation between the implementation I and the (formal) specification S. Such a relation precisely establishes when I is a correct implementation of S. However, to do this, we need to reason on the naturally informal implementations as if they were formal objects. The prerequisite to this is the test hypothesis, allowing a tester to assume that the output observed for one test case can be taken as a representative for (infinite) many other possible executions. For testing purposes it is useful to allow the tester to distinguish between the *controllable* events and the *observable* events.
Having established formally a conformance relation, formal techniques can now be used to automatically derive from the specifications an ideal test suite T, i.e., a set of tests by which for any implementation its conformance can be established. This ideal test suite is called *complete*, and holds the properties of *soundness*, i.e. conformant implementations are never rejected, and *exhaustiveness*, i.e. all non conformant implementations are rejected. The latter property however would require infinite test suites in almost all practical cases, therefore a selection of a finite test suite is made, by which only soundness is preserved, while exhaustiveness is lost.
The selection can be made randomly, or alternatively, the tester can use his/her knowledge of the implementation under test and of the context to guide the selection; this second approach is implemented in the TGV tool and is formalized through the notion of a test purpose.
Applicability to e-Learning: In the e-Learning domain, the above dichotomy between data and behavior conformance is naturally reflected in the distinction between content and medium.
With regard to data conformance testing, the techniques discussed can be used directly for content validation. We describe such an application in Section 6.
Use of the techniques for conformance behavior testing clearly requires deep expertise in formal methods, which is not obviously yielded in standard test laboratories. However, in our opinion such results could be fruitfully transferred in the future to the conformance testing of LMS behaviour, provided that the standard required behaviour is specified in the form of state machines or equivalent formalism.

5.2 Development testing

Description: During software development, testing is carried on to **monitor product quality** and to **early detect problems and bugs**. Alongside the development process, the testing activities are themselves organized into a process involving several phases, addressing different portions of a system or the whole of it. Whichever the process adopted, we can at least distinguish in principle among the three stages of **unit, integration** and **system test**.

Unit test purpose is to ensure that the unit (the smallest separately testable piece of software, e.g., a class or a module) satisfies its functional specification and/or that its implemented structure matches the intended design structure. Unit tests can also be applied to check interfaces (parameters passed in correct order, number of parameters equal to number of arguments, parameter and argument matching), local data structure (improper typing, incorrect variable name, inconsistent data type) or boundary conditions. Integration testing instead is specifically aimed at exposing the problems that can arise while software pieces or components are aggregated to create a larger component. Even though the single units are individually acceptable when tested in isolation, in fact, they could still result in incorrect or inconsistent behavior when combined in order to build complex systems. For example, there could be an improper call or return sequence between two or more components. Integration testing thus is aimed at verifying that each component interacts according to its specifications as defined during preliminary design. In particular, it mainly focuses on the communication interfaces among integrated components. Finally, system test involves the whole system embedded in its actual hardware environment and is aimed at ensuring that each system function works as expected (or, according to the user requirements), any failures are exposed and analyzed, and additionally that interfaces for export and import routines behave as required. Such analyses increase the confidence that the developed product correctly implements the required capabilities; in addition system testing collects information useful for deciding the release of the product.

Techniques and tools: Many different techniques and tools exist for development testing of software applications. A first distinction can be made between **white-box** and **black-box** approaches, depending on whether the testing technique is based on accessibility of the source code or otherwise if the system internals are not visible (such as it is the case for COTS), and the system is only tested at its I/O interface. The two types of techniques are also distinguished as **structural** vs. **functional** testing.

In white-box, or structural testing, the program is modelled as a flowgraph, whose entry-exit paths represent the flow of control. Code-based test cases then aim at systematically covering as many paths as possible along the program flowgraph. Various code-based criteria have been proposed to realize cost/effective approximations to exhaustive path coverage (which is clearly infeasible), by identifying specific elements of the program control-flow or data-flow that are deemed to be relevant for revealing possible failures, and by requiring that enough test cases to cover all such elements be executed. For instance, the *branch coverage* criterion requires that each branch in a program be exercised (in other words, for every predicate its evaluation to true and false should both be tested at least once). Branch coverage is also said “decision coverage”, because it considers the outcome of a decision predicate. An example for data-flow based testing is instead the *all-uses coverage* criterion, which requires that for every variable, every possible use of a definition is covered by at least one test case. Note that if a variable V is assigned a value at node X of the flowgraph, a reference to the same variable at some other node Y is a proper “use” only if there exists at least a path from X to Y that contains no other definition of V.

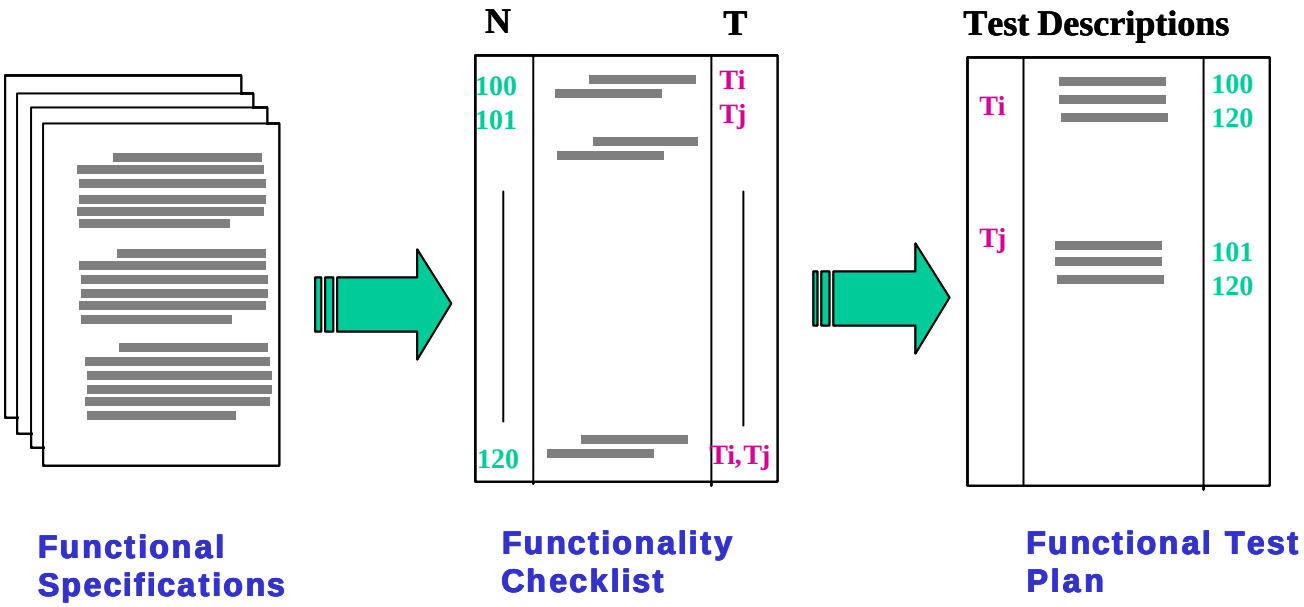

Figure 3 The functional test process.

In black-box, or functional testing, the high level specifications of a software application are analysed to identify all features and functional areas that have to be tested and to develop a comprehensive set of test cases that outline how each feature and functional area will be tested (see Figure 3). In this view, functional testing and conformance testing share a quite similar objective and present many common aspects. In fact, functional testing includes technologies for specification-based testing aimed at validating that the system as built is compliant with the developer's specifications and/or requirements, or otherwise to reveal potential inconsistencies. Therefore, the techniques and tools described above for the testing of behaviour conformance can also be applied in functional testing. Moreover, other kind of techniques can be applied to help the systematic derivation of test cases by exploring the input domain, e.g.:

- *boundary conditions*: i.e., those combinations of values that are "close" (actually on, above and beneath) the borders of the equivalence classes identified both in the input and the output domains. This test approach is based on the intuitive fact, also proved by experience, that faults are more likely to be found at the boundaries of the input and output subdomains.
- *equivalence classes*: by partitioning the input domain into subdomains of "equivalent" inputs, in the sense that any input within a subdomain can be taken as a representative for the whole subset. Hence, each input condition must be separately considered to first identify the equivalence classes. The second step consists of choosing the test inputs representative of each subdomain; it is good practice to take both valid and invalid equivalence classes for each conditions. One effective method that belongs to this approach is the well known Category-Partition (CP) that we describe below.

In the CP method, for each functional unit, the tester identifies the environment conditions (the required system properties for a certain functional unit) and the parameters (the explicit inputs for the unit) that are relevant for testing purposes: these are called the test *categories*. For each category the significant (from the tester's viewpoint) values that it can take are then selected, called the *choices*. A suite of test cases is obtained by taking *all the possible combinations of choices for all the categories*. To prevent the construction of redundant, not meaningful or even contradictory combinations of choices, in CP the choices can be annotated with *constraints*, which can be of two types: *i*) either properties or *ii*) special conditions. In the first case, some properties are set for certain choices, and *selector* expressions related with them (in the form of simple IF conditions) are

associated with other choices: a choice marked with an IF selector can then be combined only with those choices from other categories that fulfill the related property. The second type of constraints is useful to reduce the number of test cases: some markings, namely "error" and "single", are coupled to some choices. The choices marked with "error" and "single" refer to erroneous or special conditions, respectively, that we intend to test, but that need not to be combined with all possible choices. The list of all the choices identified for each category, with the possible addition of the constraints, forms a "Test Specification". It is not yet a list of test cases, but contains all the necessary information for instantiating them by unfolding the constraints.

Finally we also mention *Exploratory testing* [5] that is an alternative approach to system testing. It is defined as simultaneous learning, test design, and test execution; that is, the tests are not defined in advance in an established test plan, but are dynamically designed, executed, and modified. The effectiveness of exploratory testing relies on the tester's knowledge, which can be derived from various sources: observed product behavior during testing, familiarity with the application, the platform, the failure process, the type of possible bugs, the risk associated with a particular product, and so on.

As a consequence of the large spreading of Web applications a high demand is emerging for techniques and tools for testing Web based systems. Moreover, the new available technologies allow developers to insert sophisticated functions and dynamically linked services. Unfortunately, the responsibility for their organization and evolution is often left to the developers themselves. Another problem is posed by the rapid evolution of these systems. Some researchers [10] have proposed UML models for the high level representation of Web applications. Such models have then been exploited as the starting point for several analyses, which can help in the assessment of the static site structure, and to drive Web application testing. For instance web-based white box testing criteria for the semi-automatic generation of the test cases have been developed [10], [11].

Applicability to e-Learning: Considering the e-Learning domain, traditional techniques of development testing naturally apply, and should be considered, in the development process of the e-Learning medium. With regard to the content, they are not relevant, but it can be interesting to consider instead the application of approaches proposed for the functional testing of web systems to the analysis and testing of the content.

5.3 API Testing

Description: An API (Application Programming Interface) specifies a set of software functions that can be invoked by other software applications to perform some required service. By invoking the available APIs, an application can use the functions provided in seamless way, i.e., the end-user does not need to know, and is generally unaware, of being making an external service invocation.
For API testing [REF Jorg] then we need to simulate the potential invocation of the API by end-user applications. The test cases must exercise in systematic way the relevant features and functionalities of the provided services. Hence the test cases must on one side vary the parameters of the API test invocations, so to solicit interesting conditions. On the other side, the invocations must be run in varying sequence orders.
Techniques and Tools: Techniques for functional testing overviewed in the previous subsection are also useful for API testing. An hybrid approach combining the exploration of API parameters boundary conditions and state-based test generation techniques is proposed in [REF]. The authors propose to adopt together the Category-Partition method described above, with Markov models.
By means of the latter we derive the set of relevant states, and afterwards we consider the relevant variations of parameter values in each state.
Applicability to e-Learning: techniques for API testing are of high relevance in the e-Learning domain for what concerns the deployment of the content on the LMS.

5.4 Usability testing and GUI testing

Usability is a very important aspect to test against, especially for those systems, such as in fact e-Learning systems, whereby the product is going to be used by a large number of generic users, meaning that we cannot make any assumption or restriction about the expertise and capability of the perspective users.

Since the last twenty years methodologies for designing and implementing usable software have been defined. They are generally classified under the discipline of the Human-Computer Interaction (HCI) and include the definition and observation of the end users and their tasks, empirical measurements of system usage, and iterative development [6]. Thus usability testing can be rigorously structured or highly informal, quite expensive or virtually free, time-consuming or very quick.

In Usability testing, the aspect that is evaluated is the measure to which the system can be used, understood, learned, used and liked when the application is used under specified conditions. To evaluate the system overall usability, features that are tested can be the ease-of-navigation, layout and design, error feedback, performance, and consistency of the software application.

To this purpose one of the most referred work for usability testing is the Rubin's book [12] in which usability testing is defined as " a process that employs participants who are representative of the target population to evaluate the degree to which a products meets specific usability criteria". In particular he indentifies six main steps for performing usability testing:

- Define the problem statements or the test objectives
- Use a representative sample of end users (possibly randomly chosen)
- Represent the actual work environment
- Observe the end user while they are using the SUT, possibly using test monitor.
- Collect information representing the performance and the preferences observed.
- Individuate the possible improvements for the product.

Techniques and Tools:

As the above description suggests, usability testing rely heavily on human aspects, and hence it is hardly automatized and very expensive. Thus over the years a set of usability testing techniques have been developed and applied in different field. Among them it is important to mention [6]:

- **Card sorting:** It is generally applied for analyzing the organization and structure peculiarity aspects. It consist in giving a set of randomly ordered index cards, each one associated with a different task domain, to a group of end users and asking them to organizing the card in different small piles according certain similarity. A cluster analysis procedure is then applied for evaluating the result obtained.
- **Heuristic Evaluation:** It consists on the exploration of the system by a group of CI experts. Their purpose the identification of the usability problems (violation of one or more usability principles or heuristics) and their consequent classification. The documentation required is the project overview, describing the objectives, the target audience, expected usage etc. and a list of heuristics.
- **Scenario Based Testing:** it consists on the representation of the end-users scenarios, or specific tasks, each one associated with the major functionality of the system, and the consequent

simulation of the expected real-life usage patterns. Specific measures are then applied to the results obtained for verifying if the tasks have been correctly accomplished and evaluating for each task the time required for the completion and the number of pages accessed.

- **Questionnaire for user interaction satisfaction:** it is used mainly for evaluating the not quantifiable aspects of the interface design that contribute to the users subjective feeling of satisfaction or frustration.
- **Mining the Logs:** it uses the logs of the standard web server or httpd. The analysis of the logs provides a very useful source of information about usage patterns one a web site has gone live and prevent the use of usability experts or representative users.

In the case of e-Learning systems, usability testing of content may involve measuring the learning curve of students, and hence usability testing is based both on pedagogical aspects, as well as in well conducted statistical evaluation.

Related, yet different is GUI testing, which in modern live applications may take a large part of testing effort. It in fact requires a clear methodology that employs tools and techniques integrated to use a standardized GUI representation. Over the last years different GUI testing techniques have been developed but many times they result incomplete, ad hoc, and largely manual. The most common tools use record-playback techniques, which is developed by a test designer interacting with the GUI. The design generates mouse and keyboard events which are recorded as user events, and stored in a session—usually as a script. The tester later plays back the recorded sessions to re-create the events with different inputs [8].
Generally the process adopted for implementing GUI testing should include the following steps[8]:

- **Defining coverage criteria**. For instance requiring the execution of each user interface event to determine whether it behaves correctly.
- **Generate test case inputs:** it could consist of events such as mouse clicks, menu selections, and object manipulations.
- **Define the expected output:** It consists on the definition of the expected screen snapshots and window positions and titles.
- **Execute test cases and verify output:** Test cases execute on the software, and the tester compares the output with the expected output from, for example, an oracle.
- **Established stop criteria:** i.e. determine whether the GUI was adequately tested by analizing the checks events and resulting GUI states.

In the e-learning context another important aspect is the test for verifying the accessibility of software, which has the purpose of verifying how the learners can interact with the system based on their preferences and needs. To this purpose the Web Accessibility Initiative (WAI) outlines approaches for preliminary and conformance reviews of web sites [1] which recommend the use of "accessibility evaluation tools" to identify some of the issues that occur on a web site. A list of these kind of facilities can be find at [2]. This list includes tools such as the Bobby [16] and WAVE [17].

5.5 Reliability testing

Description: The term reliability is used in general to mean those various qualities or measures characterizing a product's successful operation over time. In today's industrial production, reliability is a strategic product variable, as important as cost and performance. For example, when we select a car, not only we compare price, comfortability, speed, but also repair records. In other words, wrt reliability we require products to prove that they can work without failing for "enough" long time. In reliability theory, the term reliability assumes a precise, mathematical meaning and is defined as a function over time R(t).

Indeed reliability is always measured against a time duration. In computer-based applications we have the choice between a continuous or a discrete time variable, depending on the failure process and on whether it is correlated with the duration of user's command execution or rather with the probability of failure of each single execution. Continuous time is best measured by CPU or execution time (more significative) than by calendar time (more easy to record): indeed, software does not fail while it is not executed! Discrete time is instead measured via the number of demands.
The reliability of a product depends on just how the customer will use it: the **operational profile** is the key notion in evaluating software reliability, it is what distinguishes operational testing from traditional development or conformance testing approaches previously discussed.
Thus the only way to validate quantitatively the reliability of a software product is via rigorous statistical testing in which the test case must be randomly generated according to the operational profile. Software reliability is a measure of the probability that a software will execute without failure for a specified time period within a specified environment. Software Reliability Engineering (SRE) encompasses state-of-practice techniques and tools for analysing, managing and improving the reliability of software products.
Techniques and tools: Software reliability measurement is a relatively new technology. The central focus is on the quantitative evaluation of the operational behaviour, as contrasted with qualitative evaluation of conventional (non reliability based) validation and verification activities.
Musa has introduced the SRET methodology, whereby "SRET is testing guided by reliability objectives and expected usage and criticality of different operations in the field" [7]. Reportedly ([7] Cap. 6), Musa's SRET approach has been successfully applied to many projects with documented strong benefit/cost ratio results. We report a very brief summary of SRET features.
The SRET approach consists of five steps:

1. Define the reliability objectives: this implies defining possible failures in the distinct modes of operation and identifying their potential impact on users (severity class); then the reliability level to be achieved for each operational mode and for each severity class must be stated quantitatively.
2. Develop the operational profile: Musa defines an operational profile as "the set of operations and their probability of occurrence". To obtain the list of operations, one can follow a five step procedure, starting from identifying a list of possibly different customer types, and progressively breaking them down into different user types, then into the different modes in which a user can invoke the system. In turn, for each system mode a functional profile is defined, i.e., a list of the functions needed by the user in each mode and their occurrence probability. Finally the functional profile, that has been defined from the user's point of view, has to be converted to the operational profile, that is system oriented: an operation is associated with running the system to accomplish a defined task.
3. Prepare the tests: this involves specifying the test cases to be executed. In this phase the scripts for automatically launching the test cases are prepared.
4. Execute the tests: the execution must be automated with the use of a test management system. In this phase we must carefully record failures and their time of occurrence.
5. Interpret test results: failure data collected during test execution are interpreted differently depending on whether we attempt to resolve detected failures or not. There are two distinct testing regimes:
   i) we modify the software in order to remove the fault: hence what we observe is actually a sequence of program versions, with plausibly growing reliability:
   ii) we leave them: in this case the test objective is to certify the reliability level of the current version of software, for example in acceptance testing. This is also called life testing, and inference techniques applied are as much the same as in testing of hardware apparatuses.
   In the first case, we are doing debug test, i.e., our goal is to increase the product reliability. We then have to select a suitable reliability growth model (RGM) and monitor the system failure intensity values as we find and remove faults. We are finished when we reach the originally set

reliability objectives. In the second case, we are doing acceptance test, i.e., our goal is to decide if a product is acceptable or must be rejected, and we decide based on the product reliability. We then simply evaluate the product failure behaviour against the required reliability levels.

Applicability to e-Learning: Of course the techniques for reliability testing can be translated directly to any domain, including e-Learning, with regard to both content and medium.

6 The case of TELCERT

In the last years different mechanisms have been proposed to make easier both the definition of an XML document model (language and structure) and the automatic check for conformance validation of a document instance against the model. Notable examples of XML document modeling techniques are Document Type Definition (DTD), XML Schema, Schematron, Relax NG. In this section we intend to give a short overview on some of these instruments with particular reference to those that could be usefully used within the Telcert project.

A first essential conformance test that must be assessed on an XML file instance is called *well-formedness*. This is a basic requirements for an XML file, if this property is not satisfied the tested file cannot even be classified as an XML file. Well-formedness can be easily verified, also simple browser generally provides conformance validation features for such kind of validation, however well-formedness cannot give guarantees on the quality of an XML file. To really enable the correct exchange of information among different applications it is necessary to establish more complex levels of conformance and to develop instruments that should permit to define document classes against which a document instance can be verified for compliance.

A lot of different document model languages, that permit to express the qualities that a document instance should have to be declared conform (hence a class), have been defined. Correspondingly, for each of them, validator parsers have been developed to automatically check the conformance of an instance to a model. We can classify two different kinds of languages:

- *Structure oriented languages* that define the structure of an XML instance providing a sort of template
- *Rule based languages* that specify a set of constraint that must hold to declare an instance conforming to the model.

Each modeling language has points of weaknesses and points of strength and in our opinion sometimes the use of more than one can rise the level of conformance obtained increasing the number of constraints that the instance must satisfy.

XML Schema

XML Schema is a powerful standard for XML document modeling, released in 2001 by the *World Wide Web Consortium*; it provides a set of mechanisms for establishing how a conformant document must be formatted. XML Schema Standard aims to replace the Document Type Definition (DTD) for modeling XML document and overcome some of its limitations. Its main feature are:

- it is functionally backward compatible with the DTD standard
- it is itself based on XML syntax
- it is namespace aware
- it permits the definition of data type and inheritance among them

The main limitation of this modeling technique is that do not permit to define relations among parts of the XML instance belonging to different subtrees.

Writing an XML Schema document is not an easy task and requires to the modeler the acquisition of complex skills in document modeling even though the XML syntax is used.

XML Schema its the most important example of structural oriented modeling language, and several validator and IDE, for making easier the instance creation phase, have been developed for XML Schema.

The IMS consortium, which specifications are generally used within TELCERT, recognized the advantages of having a description of the information model based on XML related technologies, and as consequence it has defined, for most of its specifications, an XML binding document that describes how an information model can be mapped into an XML Schema.

Schematron

Schematron [SCHE] is a simple and powerful Schema Language, which is currently undergoing ISO standardization to become ISO/IEC 19757 - DSDL Document Schema Definition Language - Part 3: Rule-based validation – Schematron.

Specifically Schematron is a rule oriented language system for specifying and declaring assertions about arbitrary patterns in XML documents (it is in fact an example of rule based language), based on the presence (or absence), of names and values of elements and attributes along paths. Schematron capabilities are strongly related to XPath capabilities being this language the main stone on which Schematron is funded. This means that is possible to define with Schematron all the relationship that it is possible to express using XPath (example of this relations are: child, descendant, parent, ancestor, following, preceding, attribute, self, etc...).

The design goals from which the Schematron design is started are:

1) Promote natural language descriptions of validation failures
2) Reject the binary valid/invalid distinction which is inherent on other schema languages permitting to express more explicative comments when an assertion/report is false/true,
3) Aim for a short learning curve by layering on existing tools (XPath and XSLT)
4) Trivial to implement on top of XSLT
5) Provide an architecture which lends itself to GUI development environments
6) Support workflow by providing a system which understands the phases through which a document passes in its lifecycle

The basic building blocks of the schematron language are **assert** and **report**. These define the constraints which collectively form the basis of a Schematron schema. It is possible to group constraints using the operator such **rule**, that permits to specify a context for the constraints, and **pattern** that permits to group rule to create a sort of template for the instance.

Schematron defines a language which, when transformed through a meta-stylesheet produce XSLT validators that can be applied to the instances producing a report file which contains clear indication of the constraints that do not hold for the instance under test as illustrated in Figure 4.

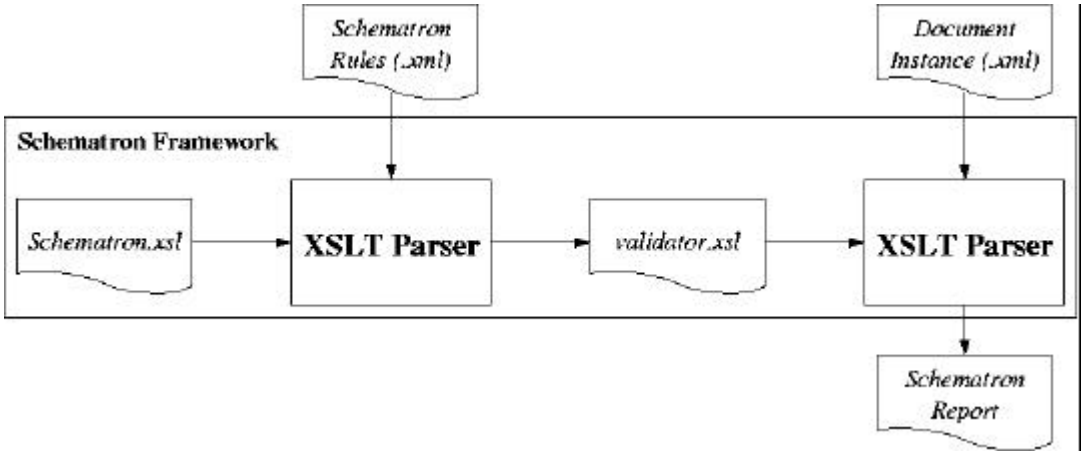

Figure 4 The Schematron Framework

A Combined Conformance Checking Methodology for Application Profiles

In this section we sketch a possible approach for the validation of an XML instance produced on the

basis of an Application Profile (AP).
From the study that we carried out within the Telcert project we noticed that it is convenient to derive an XML Schema from the Basic profile and the modifications defined in the AP. In such manner the Derived Schema (DS) can be used to validate instances, gaining the obvious advantages that the presence of an XML Schema can provide for the test a document instance. An XML Schema, in fact, embeds a lot of complex test cases that can be automatically checked reusing already available software component.
However it is not possible to express, using XML Schema, all check that could be used within an AP specification (as for instance relations among different sub-trees of an XML instance), for this reason we proposed a combined use of the XML Schema mechanism and of the Schematron framework. It is, in fact, possible to derive a set of schematron rules from the constraints embedded in the AP, as demonstrated by the successful implementation of a proof-of concept tool that we carried out within the Telcert project. In Figure 5 we give a simple outline of the proposed approach in the form of the dataflow used to implement the tool.

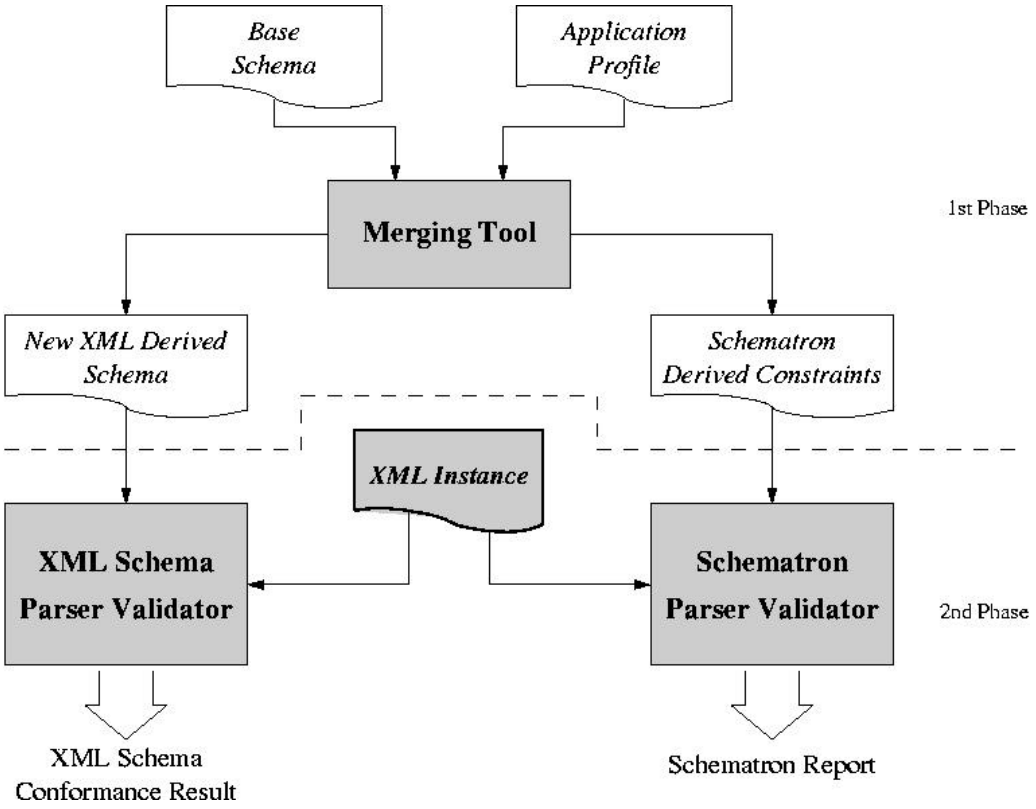

Figure 5 Proposal for a validation strategy

The advantage of the proposed solution is that it make easier the validation of an instance reducing the validation process in a two phase process, and for each phase a lot of development effort could be saved reusing already available free software component.

7 References

[1] J. Brewer, and C. Letourneau, "Evaluating Web Sites for Accessibilit"y, World Wide Web Consortium(Eds) 2002 http://www.w3.org/WAI/eval/

[2] W. Chisholm, and L. Kasday, "Evaluation, Repair, and Transformation Tools for Web Content Accessibility", World Wide Web Consortium (Eds) 2002, http://www.w3.org/WAI/ER/existingtools.html

[3] IEEE Standard Glossary of Software Engineering Terminology IEEE STD 610.12-1990

[4] A. Jorgensen and J. A. Whittaker, "An API Testing Method", on line at www.model-based-testing.org/api_testing_method.doc

[5] C. Kaner, J. Bach, and B. Pettichord, Lessons Learned in Software Testing, Wiley Computer Publishing 2001.

[6] M.D. Levi and F.G. Conrad "Usability Testing of World Wide Web Sites" BLS research paper, August 2002, on line at http://www.bls.gov/ore/htm_papers/st960150.htm

[7] M. R. Lyu (Ed.), "Handbook of Software Reliability Engineering", McGraw-Hill, 1996.

[8] A.M. Memon, "GUI Testing: Pitfalls and Process", IEEE Computer Society Press, Los Alamitos, CA, USA, v.35 n.8, p.87-88, August 2002

[9] W. Perry, "Effective Methods for Software Testing", Wiley 1995

[10] F. Ricca and P. Tonella, "Analysis and Testing of Web Applications", Proc. of *ICSE'2001, International Conference on Software Engineering*, pp. 25-34, Toronto, Canada, May 12-19, 2001.

[11] F. Ricca and P. Tonella, "Testing Processes of Web Applications", *Annals of Software Engineering*, vol. 14, pp. 93-114, 2002.

[12] J. Rubin "Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests" New York: John Wiley and sons, 1994.

[13] TELCERT: Technology Enhanced Learning Conformance - European Requirements and Testing, http://www.opengroup.org/telcert/.

[14] Tretmans, J.: Test Generation with Inputs, Outputs and Repetitive Quiescence. Software–Concepts and Tools, 17(3):103-120, 1996.

[15] Tretmans, J.: Testing Concurrent Systems: A Formal Approach. Proc. of CONCUR'99, LNCS 1664 (1999), pp. 46–65.

[16] Watchfire, 2003, Welcome to Bobby. http://bobby.watchfire.com/bobby/html/en/index.jsp

[17] WebAIM, undated, WAVE 3.0 Accessibility Tool http://www.wave.webaim.org:8081/wave/index.jsp