



Technology Enhanced Learning: Conformance - European Requirements & Testing

Title:	AD31 - TELCERT Glossary of Terms
Editor(s):	Colin Smythe & Kevin Riley (eLoki Ltd)
Version:	AD31-v1.1
Version date:	5 th August, 2004
Status:	Working document
Distribution:	Public
Summary:	This is the glossary of terms for usage within the TELCERT project. The document is intended to capture the common understanding of the consortium on key terms used. As such, its contents may be subject to change as the project advances and is thus released as a working document.

GLOSSARY OF TERMS

Activity Diagram	Within UML, an <i>Activity Diagram</i> describes the sequencing of activities and includes support for sequential and parallel behaviour. These diagrams are particularly useful for modelling workflow.
Application Profile	An <i>Application Profile</i> is a derivation of a base specification that addresses: localization - the specialization of one or more conceptual data schemas (source schemas) to the precise needs of a community, generating a derived schema; representation - mapping the localized conceptual schema(s) to a generic binding for interchange; and transaction – the definition of how the abstract interface and service model (i.e. the APIs, and implied/stated transactions) are to be realized utilizing a concrete platform technology.
Behaviour	The term behaviour is used to describe the ways in which a system, an application, an object, a component, etc. is permitted to respond to stimuli. The behaviour is normally represented by a set of stable states across which the system moves in response to stimuli that cause the transition. Behaviour is normally time dependent and so a pure data model representation is insufficient.
Black Box Test	Black-box testing, also called <i>Functional Testing</i> , relies on the input/output behaviour of the system. In particular the system is subjected to external inputs, so that the corresponding outputs are used to verify the conformance of the system to the specified behaviour, with no assumptions of what happens in between. Therefore in this process we assume knowledge of the (formal or informal) specification of the system under test, which can be used to define a behavioural model of the system, which can be represented by a transaction flowgraph.
Bound Data Profile	This consists of a profile of both an information model and a binding for a static structure. The binding may be to XML, Resource Description Format (RDF), or some other technology. It is also possible that a Bound Data Profile will contain a single information model profile but more than one binding profile
Bound Service Profile	This is the complete set of profiles for the abstract service definition, the service binding (for example, in <i>WSDL</i>), the information models of its static structures, and their bindings.
Boundary Value Analysis	This is a complimentary approach to <i>Equivalence Partitioning</i> , and concentrates on the errors occurring at boundaries of the input domain. The test cases are thus chosen near the extremes of the class.
Business Integration-Information Conformance Statements (BI-ICS)	This specification addresses the need for business partners to declare information conformance to well-known type systems and processes in order to facilitate different constraints that vary depending on how the information is used. In particular BI-ICS intrinsically supports the declaration of a sequential conformance model with specific W3C XML Schema instances, Multipurpose Internet Mail Extensions (MIME) types, and <i>XML Stylesheet Transformation (XSLT)</i> transforms based on Schematron assertions. BI-ICS is extensible for extended support of alternate type systems, process mechanisms, and conformance models;

Category Partitioning	The category-partition method provides the tester with a systematic method for decomposing a functional specification into test specifications for individual functions. A particular benefit is that the tester can control the size of a product's test suite, not by arbitrarily stopping when a given number of tests have been written, but by specifying the interaction of parameters and environments that determine the tests.
Certification	<i>Conformance testing</i> , in addition to aiding the development of specifications and product implementations, forms the basis of product certification. <i>Certification</i> increases the assurance of quality delivered by conformance testing and also adds significant market advantages to vendors of conforming products.
Class	The <i>Class</i> of an <i>Object</i> is the abstract description of a set of object instances. It describes the set of objects instances that share the same attributes, operations and relationship with other objects. The dependencies between classes are shown using a <i>Class Diagram</i> .
Class Diagram	A <i>Class Diagram</i> describes the objects in the system and the various kinds of static relationships that exist among them. The static relationships are Associations (a student may read a number of books) and Sub-types (a student is a type of person). Class diagrams also show the attributes and operations of a class and the constraints that apply to the way the objects are connected e.g. the multiplicity.
Collaboration Diagram	In <i>UML</i> , a <i>Collaboration Diagram</i> is one form of <i>Interaction Diagram</i> . The Collaboration Diagram shows the exchange of messages between the objects for a particular use-case. Unlike the <i>Sequence Diagram</i> the order of the messages is denoted by explicit numbering and not by their order in the collaboration diagram.
Communication Diagram	In <i>UML</i> , the <i>Communication Diagram</i> describes how objects interact and focuses on the collaboration in space, explicitly showing the relationships/links between the objects. As per the <i>Sequence Diagrams</i> the Communication Diagrams are important for testing purposes because they represent collaborations relationships and the collaboration integration is the primary pattern for developing application specific test. In fact it exercises the interfaces between the participants of collaboration and verifies integration according to collaborations.
Component	A <i>Component</i> is a physical and replaceable part of a system that conforms to and provides the realization of a set of interfaces. Classes and components are similar however components physically exist and they consist of operations i.e. they have attributes, that are reachable only through their interfaces. The dependencies between a set of components is shown using a <i>Component Diagram</i> .
Component Diagram	In <i>UML</i> , a <i>Component Diagram</i> shows the various <i>Components</i> in a system and their dependencies. It is used to represent the physical realisation of a system. A Component Diagram can be combined with a <i>Deployment Diagram</i> . This diagram should be used when the physical information is different from the associated logical information.
Conformance	This is the statement of the properties that a system must possess in order to be defined as providing the functionality defined within the requirement specification. The implementation may provide other functionality beyond the scope of the defined conformance.

Conformance Profile	This is the further profiling of a <i>Domain Profile</i> to create a definitive statement of how the specification(s) must be used to support conformance testing. Successful completion of the corresponding conformance testing is used to identify systems that are compliant with the conformance profile.
Conformance Test Suite	A <i>conformance test suite</i> is a test suite that can be set-up and configured to test a device against certain conformance requirements.
Conformance Testing	The goal of conformance testing is to test a product against published standards. This type of testing differs from the more familiar kind of ‘development’ testing in that the target of a conformance test is to verify that each testable requirement of the standard has been correctly implemented but does not ordinarily concern itself with the means of that implementation. Conformance testing treats the implementation as a “black box” rather than examining the internal workings of the implementation. Therefore, if bugs are present that have no impact on the functional requirements of the standard then they are not the concern of conformance testing.
Content Packaging Specification	An IMS information model describing data structures that are used to provide interoperability of Internet based content with content creation tools, learning management systems, and run time environments. http://www.imsglobal.org/content/packaging/index.cfm
Data Profile	This consists solely of a profile of an information model of a static structure. The original model may be expressed in formal notation, such as UML, or as a human-readable document. Sometimes such a model is called a “Conceptual Data Schema”;
Document Type Definition (DTD)	The <i>Document Type Definition</i> (DTD) was the original XML binding control document format but this has now been superseded by XSD. The DTD defines a set of elements, their content models and attributes, and their simple relationships.
Domain Profile	Customizing parts of one or more standards and/or specifications to meet the needs of a particular market or community i.e. a domain. A set of one or more base standards and/or specifications, and where applicable the identification of chosen classes, subsets, options, vocabularies and parameters of those standards/specifications necessary for accomplishing a particular function. In this context, the SCORM is an <i>Application Profile</i> . In general an <i>Application Profile</i> will not consist solely of IMS specifications.
Dynamic Binding	This means code that implements an operation that is unknown until run time. These features make testing more difficult because the exact data type and implementation cannot be determined statically, and the control flow of the object oriented program is less transparent.
Dynamic Testing	This is the observation of the execution of the program under test and analyzes its responses in order to determine its validity. Dynamic testing reveals <i>failures</i> during execution, and a consequent analysis stage is needed to identify the <i>faults</i> that caused them.
Encapsulation	Modelling and storing with an object the attributes and the operations that an object is capable of performing. This increases the difficulty in controlling the object interactions and consequently in preparing suitable test cases to verify such interactions.

- Enterprise Service Specification** The reworking of the IMS Enterprise Specification to support behavioral definitions. This is an aggregation of the *Party Management Application Service*, *Group Management Application Service* and *Membership Management Application Service*.
- <http://www.imsglobal.org/es/index.cfm>
- Enterprise Specification** An information model that supports the interoperability of information about people, groups, and group affiliations between two or more systems.
- <http://www.imsglobal.org/enterprise/index.cfm>
- Equivalence Partitioning** Functional tests are derived from the specifications written in structured, semiformal language. The input domain is partitioned into equivalence classes so that elements in the same class behave similarly. In this context the *Category Partition* is a well-known and quite intuitive method, which provides a systematic, formalized approach to partition.
- Error** The cause of a *failure* i.e. the missing or incorrect code is a *fault*. In particular, a *fault* may remain undetected until some stirring up event activates it. In this case it brings the program into an intermediate unstable state, called error, which if propagated to the output causes a *failure*. The process of failure manifestation, which can be iterated recursively: a *fault* can be caused by a *failure* in some other interacting system.
- Failure** A *failure* is the manifested inability of the program to perform the function required i.e. a system malfunction evidenced by incorrect output, abnormal termination or unmet time and space constraints. The cause of a *failure* i.e. the missing or incorrect code, is a *fault*. In particular, a *fault* may remain undetected until some stirring up event activates it. In this case it brings the program into an intermediate unstable state, called *error*, which if propagated to the output causes a *failure*. The process of failure manifestation, which can be iterated recursively: a *fault* can be caused by a *failure* in some other interacting system.
- Fault** A *failure* is the manifested inability of the program to perform the function required i.e. a system malfunction evidenced by incorrect output, abnormal termination or unmet time and space constraints. The cause of a *failure* i.e. the missing or incorrect code, is a *fault*. In particular, a *fault* may remain undetected until some stirring up event activates it. In this case it brings the program into an intermediate unstable state, called error, which if propagated to the output causes a *failure*.
- Functional Test** *Functional testing*, also called *Black-box* testing, relies on the input/output behaviour of the system. In particular the system is subjected to external inputs, so that the corresponding outputs are used to verify the conformance of the system to the specified behaviour, with no assumptions of what happens in between. Therefore in this process we assume knowledge of the (formal or informal) specification of the system under test, which can be used to define a behavioural model of the system, which can be represented by a transaction flowgraph.

General Web Services Specification	This is a generic IMS specification that has been created to enable the <i>service-oriented architecture</i> specifications developed by IMS to be supported using a <i>Web Services</i> infrastructure. This specification defines the IMS Web Services Basic Profile (based upon the WS-I Base Profile v1.0). It also explains how to take an IMS information model and provide a <i>WSDL</i> -based binding.
Impartiality	This means that the test suite does not favour one conformant product over another.
Implementation Conformance Statement (ICS)	This is the Supplier's documented set of claims describing precisely the way in which the implementation meets the Conformance Requirements, including which optional features are supported. An ICS contains one or more <i>Static Conformance Requirements</i> . The ICS also provides a precise identification of the Certified Product and the environment in which conformance is guaranteed.
Implementation Under Test (IUT)	This refers to the entity that is being tested. This can be anything that implements the specification to which it is claiming conformance. So this may be a hardware component, a software component, etc.
IMS Abstract Framework (IAF)	The abstract representation of the systems, applications, components and infrastructure used to provide e-learning. The abstract framework can be considered a meta-architecture and as such it is not possible to create an implementation of abstract learning framework. This is an IMS white paper.
Infrastructure	The mechanism through which the <i>Application Components</i> and <i>Common Components</i> interact i.e. communication, messaging, discovery, workflow, security, encryption, transaction. The infrastructure can be supplied through any appropriate combination of communications technology.
Inheritance	The properties defined for a class are inherited by its subclasses, unless it is otherwise stated. However, a method that is tested to be "correct" in the context of the base class does not guarantee that it will work "correctly" in the context of the derived class. The retesting of inherited methods in a different context is therefore a rule, which increases the number of tests to perform.
Integration Test	Integration is a process by which components are aggregated to create a larger component. Even though the single components are individually acceptable when tested in isolation, they could produce incorrect or inconsistent behaviour when combined in order to build complex systems. For example, there could be an improper call or return sequence between two or more components. Therefore integration testing is specifically aimed at exposing the problems that arise from the combination of components by the verification that each component behaves according to its specification defined during preliminary design. In particular, it is mainly focused on the communication interfaces among integrated components.
Interaction Diagram	<i>Interaction Diagrams</i> are models that describe how groups of objects collaborate in some behaviour. There are two kinds of Interaction Diagram: <i>Sequence Diagrams</i> and <i>Collaboration Diagrams</i> .

Learner Information Package Specification	<p>An IMS information model used to describe learners and their learning history. This specification is well suited to supporting transcripts, life-long learning logs and other similar profiles.</p> <p>http://www.imsglobal.org/profiles/index.cfm</p>
Learning Design Specification	<p>A description of a method enabling learners to attain certain learning objectives by performing certain learning activities in a certain order in the context of a certain environment. This is an IMS specification.</p> <p>http://www.imsglobal.org/learningdesign/index.cfm</p>
Object	<p>An <i>Object</i> is an abstraction of a physical entity or conceptual thing. It has states and an inherent identity. It attains certain behaviour through a set of predefined operations that may access or change its state. An object encapsulates its properties (called attributes) and the operations that access or change those properties. The state of the object is determined by the values of its attributes. The <i>Class</i> of an object is the abstract descriptor of a set of object instances.</p>
Object Constraint Language (OCL)	<p>This is the language that is used with <i>UML</i> to express constraints that are to be applied to the attributes defined within a class. OCL is a pure expression language. Therefore, an OCL expression is guaranteed to be without side effect; it cannot change anything in the model. This means that the state of the system will never change because of an OCL expression, even though an OCL expression can be used to specify a state change, e.g. in a post-condition. All values for all objects, including all links, will not change. Whenever an OCL expression is evaluated, it simply delivers a value.</p>
Open Knowledge Initiative (OKI)	<p>The <i>Open Knowledge Initiative</i> is defining a service-based architecture, consisting of service and <i>Open Service Interface Definition</i> specifications, designed to support educational software, e-learning applications, and learning management systems. OKI also provides support services to its developer and architectural specification communities, through on-line forums, documentation, training, and community events. OKI is led by the Massachusetts Institute of Technology and is a collaborative effort among a number of higher education institutions.</p> <p>http://web.mit.edu/oki/</p>
Open Service Interface Definition (OSID)	<p>The <i>OKI</i> has produced a series of <i>OSIDs</i> informed by a broad architectural view of the educational technology landscape. As LMSs have become a core component of the campus information technology infrastructure, OKI seeks to simplify and enhance the creation of educational applications. The <i>OSIDs</i> are an abstraction layer between the programmer and the enterprise infrastructure systems of his or her campus. Each <i>OSID</i> is characterized by a tightly defined set of methods and strict boundaries.</p>
Package Diagram	<p>A <i>Package Diagram</i> is a visualisation of the dependencies between two or more <i>Packages</i>.</p>

Partition Testing	A broad category of testing in which the program input domain is divided into sub-domains and it is tacitly assumed (this is the underlying test hypothesis) that for every point within a same sub-domain the program either succeeds or fails. Therefore, thanks to the test hypothesis only one or few points within each sub-domain need to be checked, and this is what allows for getting a finite set of tests out of the infinite domain.
Polymorphism	The ability to bind a reference to more than one object. This means that each possible binding of a polymorph component requires a separate test.
Question & Test Interoperability Specification	An information model that describes question (item) and test (assessment) data and their corresponding results reports. This is an IMS specification. http://www.imsglobal.org/question/index.cfm
Regression Testing	According to (IEEE610-90), regression testing is the “selective retesting of a system or component to verify that modifications have not caused unintended effects [...]”. In practice, the idea is to show that a system that previously passed tests still does. It defines repetition of tests intended to show that the software’s behaviour is unchanged, except insofar as required. A trade-off must be made between the assurance given by regression testing every time a change is made and the resources required to do that. Regression testing may apply to functional and non-functional testing.
Repeatable	This is defined to mean that if a test is repeated at a different location, on the same (or functionally equivalent) implementation, then the test will produce the same result.
Reproducible	This is defined to mean that if a test is repeated at a different location, on the same (or functionally equivalent) implementation, then the test will produce the same result.
Resource List Interoperability (RLI) Specification	The IMS RLI specification details how structured meta-data can be exchanged between systems that store and expose resources for the purpose of creating Resource Lists and those that gather and organize those Resource Lists for educational or training purposes. A typical example of such a Resource List is a reading list. http://www.imsglobal.org/rfi/index.cfm
Reusable Definition for Competency or Educational Objectives (RCDEOs)	A computer-accessible representation of skills, knowledge, tasks, and learning outcomes that can be used in any particular context.
Reusable Definition for Competency or Educational Objectives Specification	An information model for describing, referencing and exchanging definitions of competencies, primarily in the context of online and distributed learning. This is an IMS specification. http://www.imsglobal.org/rcd/index.cfm

Schematron	This is a simple and powerful Structural 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 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. Its target uses are for software engineering requiring document validation, for scholarly research over patterns in graph-structured data, for automatic creation of external mark-up, and to aid accessibility of documents for people with disabilities.
Sequence Diagram	A <i>Sequence Diagram</i> is one form of <i>Interaction Diagram</i> . The Sequence Diagram shows the exchange of messages between the objects for a particular use-case. The order of the messages on the diagram denotes the sequence.
Service & Bound Data Profile	This consists of both a profile of a service definition, and profiles of related static structures in both their information models and their bound forms;
Service & Data Profile	This consists of both a profile of a service definition, and profiles of related information models of static structures. For example, an abstract interface model and its parameter data types - both expressed using <i>UML</i> - could be profiled using a Service and Data Profile.
Service Profile	This consists of a profile of an abstract service definition. The service may be defined using UML models e.g. as per an <i>OKI Open Service Interface Definition</i> .
Service-oriented Architecture (SoA)	A Service-oriented Architecture defines how two or more entities interact in such a way as to enable one entity to perform a unit of work on behalf of another entity. The unit of work is referred to as a service, and the service interactions are defined using a well-defined description language. Each interaction is self-contained and loosely coupled, so that each interaction remains independent of any other interaction. While SOAP-enabled Web Services are the most common implementation of SoA, Web Services are not necessarily required to define a SoA. The protocol independence of SoA means that different consumers can use services by communicating with the service in different ways. Service Orientation is a method of architecting systems of autonomous services. Services are built to last, with good availability and reliability, and systems are built to change, with new service topologies evolving over time.
Sharable Content Object Reference Model (SCORM)	The Sharable Content Object Reference Model (SCORM) defines a Web-based learning ‘Content Aggregation Model’ and ‘Run-Time Environment’ for learning objects. At its simplest, it is a model that references a set of interrelated technical specifications and guidelines designed to meet the DoD’s high-level requirements for Web-based learning content. From an IMS perspective, SCORM is an Application Profile.

<http://www.adlnet.org/index.cfm>

Sharable State Persistence Specification

The Shareable State Persistence specification describes an extension to e-learning runtime systems e.g. SCORM that enables the storage of and shared access to state information between content objects. There is currently no prescribed method for a content object to store (arbitrarily complex) state information in the runtime system that can later be retrieved by itself or by another content object. This capability is crucial to the persistence of the sometimes complex state information that is generated by a variety of interactive content e.g. simulations and that is currently stored and retrieved in proprietary formats and through proprietary methods. This is an IMS specification.

<http://www.imsglobal.org/ssp/index.cfm>

Simple Object Access Protocol (SOAP)

SOAP provides the definition of an XML document which can be used for exchanging structured and typed information between peers in a decentralized, distributed environment. It is fundamentally a stateless, one-way message exchange paradigm, but applications can create more complex interaction patterns (e.g. request/response, request/multiple responses, etc.) by combining such one-way exchanges with features provided by an underlying transport protocol and/or application-specific information. *SOAP* is silent on the semantics of any application-specific data it conveys, as it is on issues such as the routing of *SOAP* messages, reliable data transfer, firewall traversal, etc. However, *SOAP* provides the framework by which application-specific information may be conveyed in an extensible manner. Also, *SOAP* provides a full description of the expected actions taken by a *SOAP* processor on receiving a *SOAP* message.

<http://www.w3.org/2000/xml/Group/>

Simple Sequencing Specification

A method for representing the intended behaviour of an authored learning experience. Therefore, any learning technology system can sequence the discrete learning activities in a consistent way. This is an IMS specification.

<http://www.imsglobal.org/simplesequencing/index.cfm>

SOAP with Attachments

SOAP with Attachments is the extension of *SOAP* to support the exchange of MIME files as a part of the XML message data body. This allows *SOAP* to be used to exchange non-XML information in the same overall structure as the XML message document.

<http://www.w3.org/2000/xml/Group/>

State Diagram

Within *UML* the *State Diagram* captures object life cycles and shows events (messages, time, errors, and state changes), which can affect object states over time. Some of the uses of the state diagrams are to capture all possible scenarios for a given object in a unique view and to form the basis for a comprehensive code generation, and to show the possible state transitions that can be used for white box testing.

Static Conformance Requirement (SCR)

An *Implementation Conformance Statement* is the Supplier's documented set of claims describing precisely the way in which the implementation meets the Conformance Requirements, including which optional features are supported. These claims are described as *Static Conformance Requirements* (SCRs): an SCR being a reference to a single conformance requirement. Therefore an ICS contains one or more SCRs.

Static Testing	This does not involve any program execution. To speak of “ <i>Static Testing</i> ” is a sort of contradiction: static techniques are not “testing” in the strict interpretation of this term i.e. according to the above definition. Static verification techniques are applicable during all the development process for different purposes such as for instance to check the adherence of the code to the specification or to detect defects in code by its inspection or review.
Structural Test	<i>Structural Testing</i> , also called <i>White-box testing</i> , requires complete access to the object’s structure and internal data, which means the visibility of the source code. The tests are derived from the program’s structure, which is also used to track which parts of the code have been executed during testing.
System Test	<i>System Test</i> involves the whole system embedded in its actual hardware environment and is mainly aimed to verify that the system behaves according to the requirements document. In particular it attempts to reveal bugs that cannot be attributed to components as such, to the inconsistencies between components, or to the planned interactions of components and other objects.
Unit Test	A unit is the smallest testable piece of software, consisting of hundreds or a few lines of source code, and generally representing the result of the work of one programmer. The Unit test’s purpose may be to ensure that the unit satisfies its functional specification and/or that its implemented structure matches the intended design structure. When the tests reveal an anomalous behaviour, it is said that there is a unit bug. Unit tests can also be applied for testing interfaces (parameter passed in correct order, number of parameters equal to number of arguments, parameter and argument match), local data structure (improper typing, incorrect variable name, inconsistent data type) or boundary condition.
Use-case	A <i>use-case</i> is a set of scenarios tied together by a common user goal; a scenario is a sequence of steps describing an interaction between a user and a system. A use-case can be explained through a <i>Use-case Diagram</i> .
Use-case Diagram	A Use-case Diagram is a combination of visual and textual information used to describe a use-case. Both tabular and graphical representations are used to realise use-case diagrams. A use-case diagram identifies the Actors undertaking the user actions.
Web Services	A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format - specifically <i>WSDL</i> . Other systems interact with the Web service in a manner prescribed by its description using <i>SOAP</i> messages, typically conveyed using HTTP with an <i>XML</i> serialization in conjunction with other Web-related standards.
Web Services Description Language (WSDL)	<i>Web Services Description Language (WSDL)</i> is an <i>XML</i> format for describing Web services. WSDL enables the separation of the description of the abstract functionality offered by a service from concrete details of a service description such as ‘how’ and ‘where’ that functionality is offered.
White Box Test	<i>White-box testing</i> , also called <i>Structural Testing</i> , requires complete access to the object’s structure and internal data, which means the visibility of the source code. The tests are derived from the program’s structure, which is also used to track which parts of the code have been executed during testing.

World Wide Web Consortium (W3C)	<p>A non-profit organization created to develop common protocols and promote the evolution and interoperability of the World Wide Web.</p> <p>http://www.w3c.org</p>
WS-I Basic Profile	<p>The Web Services Interoperability Consortium (WS-I) has released the final specification for the <i>WS-I Basic Profile</i> v1.0a. This profile provides interoperability guidance for core Web services specifications such as <i>SOAP</i>, <i>WSDL</i>, and <i>UDDI</i>. In addition, they have also released some testing tools that help developers determine whether their <i>Web Services</i> comply with the Basic Profile guidelines.</p>
XForms	<p>This is <i>W3C</i>'s name for a specification of Web forms that can be used with a wide variety of platforms of varying capabilities, for instance, desktop computers, television sets, personal digital assistants, cell phones, computer peripherals and even paper. An important concept in XForms is that forms collect data, which is expressed as <i>XML</i> instance data. Among other duties, the <i>XForms</i> Model describes the structure of the instance data. This is important, since like <i>XML</i>, forms represent a structured interchange of data. Workflow, auto-fill, and pre-fill form applications are supported through the use of instance data.</p>
XML	<p>Extensible Markup Language. A uniform method for describing and exchanging structured data that is independent of applications or vendors.</p> <p>http://www.w3.org/XML/</p>
XML Binding	<p>The permitted syntax and semantics of the <i>XML Binding</i> is defined using the appropriate <i>XSD(s)</i>.</p>
XML Instance	<p>An <i>XML Instance</i> is a data structure, typically a file, which contains a series of <i>XML</i> statements. <i>XML</i> control documents (<i>DTDs</i> and <i>XSDs</i>) are used to determine the validity of the structure and contents of the <i>XML</i> instance. An <i>XML Binding</i> specification is in effect a definition of the valid structure and contents of the corresponding <i>XML</i> instance.</p>
XML Schema Definition (XSD)	<p><i>XML Schema Definition (XSD)</i> is the primary <i>XML</i> binding control document format of <i>IMS</i> (at present these bindings are working to the May 2001 version of <i>XML Schema</i>). The <i>XSD</i> defines elements, their content models, and attributes. It also defines the standard <i>IMS</i> vocabularies. The <i>XSD</i> defines the element types and attribute groups separately from the elements.</p> <p>http://www.w3.org/XML/Schema</p>
XML Transformations (XSLT)	<p><i>XSLT</i> is designed for use as part of <i>XSL</i>, which is a stylesheet language for <i>XML</i>. In addition to <i>XSLT</i>, <i>XSL</i> includes an <i>XML</i> vocabulary for specifying formatting. <i>XSL</i> specifies the styling of an <i>XML</i> document by using <i>XSLT</i> to describe how the document is transformed into another <i>XML</i> document that uses the formatting vocabulary. <i>XSLT</i> is also designed to be used independently of <i>XSL</i>. However, <i>XSLT</i> is not intended as a completely general-purpose <i>XML</i> transformation language. Rather it is designed primarily for the kinds of transformations that are needed when <i>XSLT</i> is used as part of <i>XSL</i>.</p>

**XML Transition Network
Definition (XTND)**

This provides a generic *DTD* that uses XEXPR. In many systems, transition networks are used to describe a set of states and the transitions that are possible between them. Common examples are such things as automatic teller machine control flows, editorial review processes, and definitions of protocol states. Typically, each of these transition networks has its own specific data format, and its specific editing tool. Given the rapid transition to pervasive networking, and to application integration and interchange, a standard format for transition networks is desirable. This document defines such an interchange format, defined in *XML*.

XSL

The eXtensible Stylesheet Language is a family of recommendations for defining *XML* document transformation and presentation. It consists of three parts: *XSLT*, XPath and XSL Formatting Objects (XSL-FO).