DIFFICULTIES IMPLEMENTING STANDARDS

Tineke M. Egyedi[1] and **Ajantha Dahanayake**
DELFT UNIVERSITY OF TECHNOLOGY

Abstract
This paper explores why diversity among and incompatibility between standards implementations arises. An answer is sought top-down by means of institutional analysis, and bottom-up by analysing standards cases (SGML/ XML, OSI standards, and UML).
The analyses highlight very diverse causes (errors, ambiguities, inconsistencies, and parallel options in standards; functional deviations, etc.). To structure the findings, a taxonomy is drawn up. Its aim is to help identify and localise causes of standards implementation problems.
The authors conclude that, although further research is required, the aim of implementability should acquire higher priority in standards development.

IT standards policy usually focuses on standards development – which implicitly assumes that having a standard or implementing[2] it suffices to achieve interoperability (compatibility) between products. However, interoperability is only assured if standards' specifications are implemented consistently. This requires, for example, that standards are unambiguous, which is often not the case. If standards are interpreted differently, incompatibility and lack of exchangeability between different implementations is likely to occur.

Sometimes companies intentionally introduce deviant standards' implementations as aggressive market strategy. There are several such implementation-oriented strategies. The most well known one is the *embrace-and-extend* strategy. Thereby extra functionalities are built into a standards implementation in a manner that undermines its interoperability with other standard-compliant products. Egyedi & Hudson[3] (2001) specify two other strategies as well.

> "As much harm can be caused by not implementing part of the standard (*embrace-and-omit* strategy), and by introducing local adaptations to standards (*embrace-and-adapt* strategy)."

A classic example of the latter is Microsoft's use of Sun's *de facto* Java standard (Findings of Fact, 1999; Egyedi, 2001). Had it been successful, it would have further locked users into Microsoft's proprietary technology, and fragmented the Java market. As it was, it frustrated the development of a competitive platform technology - if not more. These aggressive embrace strategies require cumbersome, extra effort by third parties to repair.

Company strategies with 'bad' intentions are better documented than functional and unintentional deviant implementations. As noted before, unintentional deviance usually results from ambiguity in standards specifications. Functional reasons for introducing changes to a standard during implementation are, for example, that some of its features are superfluous, too complex, or too expensive to implement for

[1] The additional research on which this paper was based, was funded by a research grant from Sun Microsystems. We gratefully acknowledge Sun's contribution. We would also like to heartily thank the anonymous reviewers for their useful comments.
[2] The term *implementation* refers here in a narrow sense to the translation of standard specifications into hard- or software. This use of the term differs e.g. from Jakobs (2000), whose use of the term refers to a wider system only part of which is the standards-conform artefact.
[3] Egyedi & Hudson's (2001) article focuses on the question, which issues are at stake *when (de facto) standards are adapted, extended or selectively implemented.* They refer to these instances as problems of integrity - that is, as a specific subset of compatibility problems.

the intended context of use. To stay with the Java example, to make Java better applicable for small devices a group of companies (the Java Consortium) adapted the *de facto* Java standard. This was against Sun's wishes because adapting the standard would harm the scaleability of Java programs. Nonetheless, Sun's own efforts in this direction at a later stage suggest that tailoring Java to the requirements of small devices was a functional necessity. At stake is a difficult dilemma.

Summarising, the consequences of deviant standards implementations are widespread. Where compatibility standards are concerned, if a company's implementation is not standard-compliant, this diminishes its interoperability with implementations of other companies and fragments the market. Perhaps the most harmful effect is that the interoperability of standard-conforming products is not self-evident anymore. Lack of transparency on this issue can be a grave set back for market development.

In this paper, we more systematically explore the *causes for and types of problems with implementing standards*. Is the cause retraceable to features of the standard or the standards process, or are there other explanations? We restrict ourselves to deviant implementations that come about unintentionally or for functional reasons, and start our quest by focusing on interoperability standards.

The structure of the paper is as follows. We first look at the institutional setting of standards development - formal and otherwise - and whether this explains why standards give rise to problems of interpretation and implementation. That is, we examine whether the standards setting can be a source of deviance. Next, we examine three clusters of standardisation: Standard Generalized Markup Language (SGML)/ Extensible Markup Language (XML), Open Systems Interconnection (OSI) standards, and Unified Modeling Language (UML). We thus widen our scope - from interoperability standards - to include reference and modelling standards. The case findings are used to develop an initial taxonomy of standards implementation problems. In the conclusion we discuss its value and make some recommendations for further research.

Dilemmas in the Institutional Setting

There is little specific literature about problems of standards implementations (Söderström, 2002). However, a few studies exist that throw light on possible institutional causes of incompatible standards implementations. These causes are largely captured by two prominent ideals in formal standardisation, that is, the ideal of developing standards in a democratic, consensus-oriented manner and the ideal of developing implementation-independent standards. Both are highly relevant and directly related to problems of standards implementation.

Shift in emphasis *Aspects of formal standardis.*	*from ...*	*to ...*	*and to ...*
Priorities	process	outcome	use
Specified priorities	democratic process	timely delivery of standards	implementability of standards
Scope of activity	standardisation	standardisation, implementation, testing & marketing	

Table 1: *Expected shifts in emphasis in the focus of formal standardisation. (extracted from Egyedi, 1996)*

Consensus decisions

The formal standards bodies are sometimes criticised for issuing standards that are difficult to implement. The institutional set-up is blamed. The ideal of democratic, consensus-oriented decision making more or less solicits political compromises in committee standardisation. This can result in a standard which includes several options, or in intentional vagueness in the way a standard is formulated so that opposing parties can rally behind it). That is,

> " (...) [although] parsimony and consistency of standards and standards options are salient institutionalized goals of organized standardization (...), [o]ccasionally, if at the technical working level of the CCITT [Comité Consultatif International Télégraphique et Téléphonique] no consensual solution can be found, politics helps to achieve an agreement precisely because it does not consider technical details. (...) Terminating a

> conflict through the adoption of incompatible options (...) keeps the organization viable." (Schmidt & Werle, 1998, pp. 303, 271, 270)

Indeed, such outcomes are a natural consequence of the formal standards bodies' past emphasis on their guardianship of the quality of the standards process (*voluntary consensus* process), rather than on features of the standard or on standards implementability.

> "[T]he question of implementation of standards lies outside the framework of the [formal standards bodies]." (Schmidt & Werle, 1998, p. 304)

In the past, the formal standards bodies mostly treated the standards process, standards and standards implementations as successive occurrences. Consortia and other standards development fora, captured by the term 'grey standardisation' in Table 2, more often treated them as parallel occurrences. For example, with Internet standardisation the standards process includes demonstrated implementability; IETF/ RFC 2026). *De facto* standards reflect successful market implementation. Thereby standards inherently follow implementations.

Faced with new approaches and stiffer competition in the 1990s, Egyedi (1996) expected the formal standards bodies to widen their scope of activity and include provisions and procedures for implementation-oriented activities. See Table 1. Indeed, the International Organization for Standardization's (ISO) strategies for the years 2002-2004 suggest that standards conformance will be getting a very prominent place (ISO, 2001).

Implementation independence

A second main difference between formal, grey and *de facto* standardisation is their concern with *implementation-independent* standards (see Table 2). The formal standards bodies strive for standards that do not favour certain companies, technologies or markets. That is, they aim for solutions that are highly implementation-independent. In contrast, *de facto* standards are usually defined by a company (e.g. Acrobat's PDF) and often with a specific application environment in mind. Overall, they are thus implementation-dependent. The standpoint of grey standardisation groups (consortia and others) varies greatly on this issue. Standards consortia generally favour context-independent solutions that create equal market opportunities. However, sometimes a specific implementation environment is catered to (e.g. Internet).

Seeking implementation-independent solutions, as the formal standards bodies do, can pose problems with respect to implementability. For it sooner leads to the development of generic standards. Generic standards need to cater to divers application environments. The inclusion of multiple standards options is a much-used solution to address several, partly incompatible standards requirements. Such genericity usually conflicts with the specificity required for unambiguous, univocal and consistent implementations.

Style of stand. *Aspect in stand.*	*Formal standardisation*	*Grey (incl. consortium) standardisation*	*De facto standardisation*
Process, standards & implementations	successive occurrences	parallel occurrences	standards follow implementations
Implementation independence	high	medium	low

Table 2: *Characteristics of three styles of standardisation. (Source: adapted from Egyedi, 1996)*

Cases-related Implementation Issues

Different types of standards likely highlight different kind of implementation problems. In the following three clusters of standards are discussed: the interoperability (compatibility) standards of SGML and XML, standards belonging to the family of the OSI reference model, and UML, a modelling standard used for system development. We explore the kind of implementation problems which the cases raise, try to deduce a set of basic implementation problems, and use this to develop an initial taxonomy.

SGML & XML

As the reader may know, the initial idea behind XML (W3C, 1998) was to bring the - structured data exchange - functionalities of SGML (ISO 8879: 1988) to the web (Egyedi & Loeffen, 2001). In the web environment XML was to succeed SGML. The aim was that XML would remain compatible with SGML. However, this was only partly achieved. XML documents could not be processed by SGML (1988) tools. The implementation problems addressed here are set against this background.

To address incompatibility between XML and SGML, two initiatives took place. First, an ISO/IEC SGML working group drew up Technical Corrigendum 2[4] (Cor 2: 1999) that

> "(...) remedies defects revealed by the multiple adaptations of SGML for the World Wide Web (WWW), intranets, and extranets. The annex corrects errors, resolves ambiguities for which there is a clear resolution that does not cause existing conforming documents to become non-conforming, and provides a choice of alternative resolutions for other ambiguities. Although motivated by the World Wide Web, applicability of this annex extends to all uses of SGML." (SGML, 1999; annex K)

Full implementation of the technical corrigendum would make an SGML system XML compatible. However, in practice new software providers and standards implementers had no connection to SGML. They immediately turned to XML rather than implement elaborated SGML.

Second, the XML working group included non-binding recommendations in the standard. Implementation thereof was to allow XML documents to be processed by SGML (1988) software. However, the standard would not guarantee compatibility (i.e. implementation thereof only "increases chances" of interworking). Many XML system designers ignored these guidelines anyway.

The emphasis in the SGML standard has always been on its ubiquitous applicability. XML emphasises simplicity and implementability. Although the SGML standard was successful in many ways and for a very long time in IT-measures, the present popularity of XML suggests that - in a web-based environment - wide(r) implementation requires simplicity.

OSI model

The Open System Interconnection (OSI) model is a standard reference framework well known to Information and Communication Technology (ICT) students. It was initiated to rationalise and integrate standards activities in the merging fields of IT and telecommunications in the 1980s. It identifies ICT services as consisting of a set of functions that are mapped onto seven layers (i.e. physical, datalink, network, transport, session, presentation and application layer). Within these layers generic building blocks are specified, called base standards. Base standards can contain options. Problems arise if two service implementations are based on different options in base standards. This causes problems of interoperability. To avoid this problem, the formal standards bodies (ISO/IEC/JTC1 and International Telecommunication Union's (ITU) CCITT) also standardised sets of specified base standards with fixed options for certain application areas (e.g. world of banking). These are called profiles or functional standards. A *functional standard* is a ' ... document which identifies a base standard or group of base standards, together with options and parameters, necessary to accomplish a function or a set of functions'. (ECITC, 1993)

Taking a closer look at options in base standards, for the transport layer protocol, for example, a compromise of five different protocol classes was defined. This complicated interworking. To alleviate interworking problems, means were developed to allow a certain amount of negotiation of protocol classes. In addition, profiles were developed for specific applications, which defined a fixed OSI protocol stack, including the necessary transport protocol class. For example, the classes of TP0 and TP1 were prescribed for CCITT's message handling recommendation X.400. (Egyedi, 1997)

For the session layer, functional units were defined with overlapping functionalities. According to participants, this was a political compromise. There was no viable technical reason for the overlap. The consequence of the overlap was that implementers of the session protocol usually implemented one or the other combination of functional units, and not both. That is, the session layer, too, gave rise to different OSI stacks (i.e. to fragmentation) - and to interworking problems.

In sum, OSI's objective of implementation- and field-independent standards was ambitious and came at a cost. According to some critics, the costs of implementation were too high. In their opinion OSI standards

[4] The corrigendum contained two annexes. The normative Annex K on *Web SGML Adaptations* and the informative Annex L for *Added Requirements for XML*. Annex K was an optional extension of SGML [N1929].

comprised much overhead, too many options, and complex answers to specific and simple needs. To cut down costs, OSI implementers sometimes omitted functionality's that were intended to be part of the standard.[5] Nominally, OSI-compatible products resulted. In reality, only partial compliance existed. Partial implementations damaged OSI's reputation.

UML

The Unified Modeling Language (UML) was adopted unanimously by the Object Management Group (OMG) as a standard in November 1997 (OMG, 1998). The standard aimed to simplify and consolidate the large number of Object Oriented (OO) software developing methods that had emerged (e.g. Shlaer & Mellor, 1988; Coad & Yourdon, 1991; Booch, 1991); to reduce gratuitous divergence among tools; to encourage widespread use of OO modelling among developers; and to facilitate the development of a robust market of support tools and training "now that neither user nor vendor have to guess which approaches to use and support". (UML reference manual, 1998)

However, there are different types of inconsistencies in UML modelling. In modelling approaches consistency in naming is a well-known requirement for avoiding impedance mismatches. Impedance problems arise when *a unified naming convention is lacking* within and across modelling techniques. UML comprises several complementary and substitutive modelling techniques (e.g. class diagrams, state transition models, activity models, and functional models). Difference in terminology use for similar things between these modelling techniques (a) leads to *misunderstandings between the parties* involved in the system development process (e.g. developers, testers, and users); (b) aggravates the *problem of integrating information* from one model to another during the system design stage – even apart from the problem which this poses during system implementation; and (c) leads to *difficulties in the traceability and re-use of components*.

Related to the latter point, UML does not intend to be a complete development method. That is, it does not include a step-by-step development process. Originally, a companion book for UML-based system development, the Rational Unified Process (RUP) (Jacobson et al., 1999) was proposed. However, it *lacks the necessary vigour and freedom of modelling*, according to experts, and has been challenged by UML-based methodologies such as Select Perspective (Allen & Frost, 1998), Catalysis (D' Souza & Wills, 1999), UNIFACE (2000), KobrA (Atkinson et al., 2000) and CBD/e (Castek, 2000). These methodologies usually produce similar functional solutions. However, they often are *not able to replace one another or allow integration*. Also, they may differ completely in how they implement the system.[6]

UML is more *complicated* than some of its antecedents because it intends to be more comprehensive. It incorporates several kinds of models. Normally, one does not need all UML modelling techniques in each project. Although experts know how to combine parts of UML, newcomers do not. Therefore, UML profiles would be recommendable which indicate the combinations that are useful in certain situations.

Lastly, consistency in and interoperability of the UML-based system also plays at the level of the data model. An example best illustrates what is at stake. Let us suppose that 'Student' is a Class in the UML class diagram. Its real representation in the application area is an instance object with a name (e.g. S. Mohamed) and a number of other instance attributes. In the generic model, the Class 'Student' represents any student. But students may come from different countries. For example, while normally a year has 12 months, the Ethiopian year has 13 months. This poses a problem for representing the date attribute of Class 'Student'. That is, if at instance level the data model is inconsistent, this will obstruct system interoperability. (Stojanovic *et al*., 2001)

[5] For example, there are implementations of File Transfer, Access & Management (FTAM-OSI) that make it impossible to update the same file from different locations although this functionality is incorporated in the standard. Such unintended OSI implementations are cheaper, but they entail a loss of functionality. (Private communication, Eddie Michiels)

[6] Since UML is a graphic modelling language, it lacks the proper means to formalise what is needed to derive executable models and limit the implementation model generation.

Taxonomy

Implementation problems can undermine the goals of standardisation (interoperability, exchangeability, less diversity, etc.). Therefore there is a need to identify and localise such problems. In the following an attempt is made to develop a taxonomy that captures such problems. This, ultimately, to determine whether they can be solved and, if so, by what means.

Temporality: Incidental and Structural Causes

The previous sections illustrated two categories of problems:

- problems of a more *structural* kind (i.e. standards with parallel options and several parameters, overlapping functionalities, internal fragmentation, complexity, ambiguities that result from political compromises); and
- more *incidental, temporary* problems (errors, accidental ambiguities, etc.); these are irritating but usually of a passing nature.

Incidental problems such as errors and ambiguities are usually retrospectively addressed in formal standardisation by means of defect reports, technical corrigenda, etc. Such problems can partly be avoided by developing reference implementations that show how to implement parts of the standard where doubt arises, and/or by making public the rationale that underlies the decisions of the technical committee. Understanding the rationale helps to interpret standards' specifications during implementation (e.g. extension of C++ programming language[7]). See Table 3 for a list of solutions.

Structural implementation problems often result from compromises in standards development. In the OSI example, the participants represented different interests; had to integrate different views on technology (telephony- versus computer-oriented paradigms) and different application areas; and, in addition, wanted to maintain interoperability with earlier standards efforts (e.g. X25). The standards ideal of consensus decision making required that participants would reach a compromise.

The causes of structural problems can be and have partly already been addressed through institutional change. For example,

- in the case of OSI standards, the difficulty of achieving interoperability between implementations with several options has been addressed by installing the Special Group on Functional Standardisation (SGFS) for developing OSI profiles or functional standards. That is, for the purpose of interoperability a two-phased standards process is gone through.
- discussions have been held about whether or not consensus decision-making should be changed into (weighted) majority voting. This reduces the need for political compromises, but is weakens the basis for user support.
- the standards bodies could prioritise implementability in standardisation. Should the focus of the formal bodies shift more purposefully from the standardisation (process), to its outcome (standard) and to standard's use (implementations)? See Table 1. This would imply the systematic inclusion of standards conformance and interoperability testing in the standards process.

Some structural problems, however, should be recognized as fundamental dilemmas that are difficult to resolve. For example, an inherent tension exists between developing implementation-independent standards (company, technology, application etc. independent) and easily implementable standards. The former standards are generic and therefore usually include more options, and are more difficult and expensive to implement. Generic, comprehensive standardisation aims and implementability are difficult to reconcile satisfactorily, as both the OSI and the UML case confirm.

The OSI - TCP/IP debate in the early 1990s externalises this dilemma. In the debate the OSI and the Transmission Control Protocol/Internet Protocol (TCP/IP) family were staged as competing standards trajectories. Exceptions aside, they supported similar communication functions (including e.g. email and file transfer). OSI critics highlighted the lack of workability of OSI standards. OSI standards were too complex and too expensive to implement. The critics contrasted them with TCP/IP standards, which were simple and applicable, and argued for Internet solutions to OSI problems. For example, the incorporation of testing procedures was suggested to address incompatibility between OSI implementations. Moreover, from their vantage point a reference environment would be able to focus the comprehensive OSI approach and narrow

[7] Personal communication with Willem Wakker.

down the set of functions and the number of options - although this, in turn, would have been in direct conflict with OSI's aim of wide applicability. (Egyedi, 1997)

The debate illustrates the dilemma of comprehensive and implementation-independent standardisation versus implementability. It is a recurrent, irresolvable fundamental dilemma. A principled choice must be made with high costs either way.

Standards Implementation Problems	
Causes	**Possible Solutions**
1. Errors, ambiguities, inconsistencies 2. Uncertainty concerning compatibility (XML non-b. recomm.) 3. Inconsistencies within standard (e.g. use of terminology; data model, UML) 4. Parallel options and parameters (options with overlapping functionalities, OSI; alternative modelling techniques, UML) 5. Aim to be comprehensive / complexity (OSI, UML) 6. Unclear status of non-binding recommendation (XML) or companion book (UML) 7. Functional deviation and partial implementation (e.g. superfluous features, too complex, too expensive for intended use; SGML, OSI)	a. Technical corrigenda, revisions, defect reports, ... (1) b. Include decision making rationale in standard (1) c. Unified naming convention (1) d. Functional standards & profiles (OSI & UML) (4) e. Standardisation as a two-step selection process (e.g. funct. specs in 2nd step) (4) f. Reference implementation (1) g. Interoperability testing (1,2) h. Interoperability conformance statement (1,2) i. Reference guide (included in standard) (1) j. Companion book (unsatisfactory solution; UML) (1) k. Negotiation mechanisms between protocol classes (4) l. Weighted majority voting (4) m. Prioritise implementability (5)

Table 3: *Standards implementation problems: main causes and possible solutions*

Locus: State and Process

To further specify and categorise causes of implementation problems, we focus on interoperability, the standardisation aim which featured most prominently in the examples discussed in the previous section. If implementations are not interoperable, despite the best of intentions, this can be due to problems attributable to the different phases leading up to standards implementation. See Figure 1. The figure highlights the three main *states* of a standard: the conceptual idea, the specification, and the implementation. It further identifies two *translation processes* between these states: the standard process and the implementation process. If the standards process leads to ambiguous specifications, no matter how well thought-out the implementation process, interoperability problems among implementations may still arise. That is, together the two processes determine whether or not standards implementations will be interoperable.

The figure includes a set of contextual causes of standards implementation problems for the sake of completeness. But they are not specified in order to keep sight of the main factors. Only the influence of institutional factors on the standards process is explicitly included because of its salience in the case studies and its policy relevance for standards organisations.

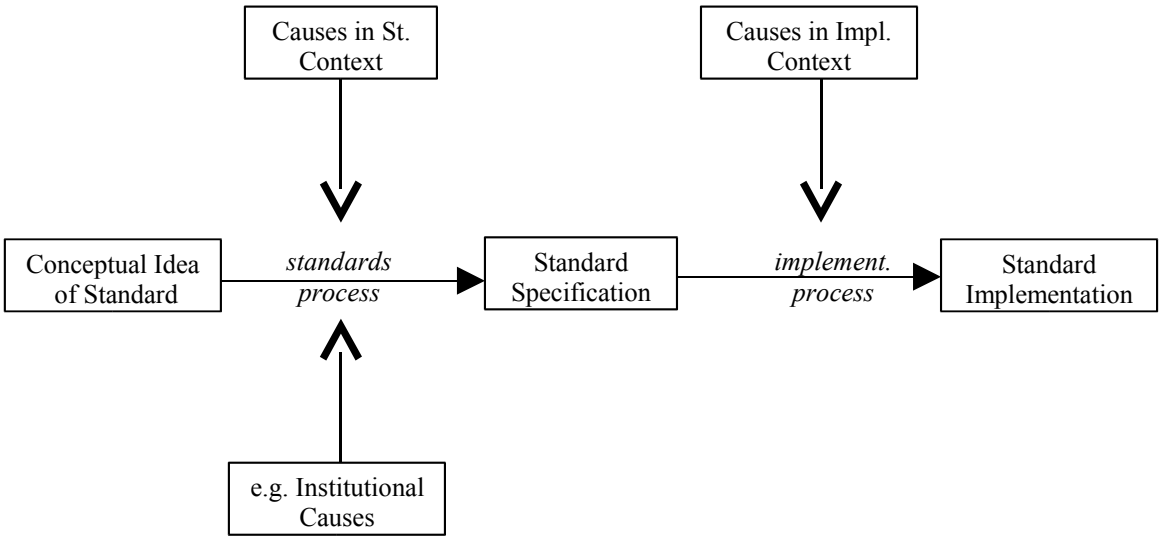

Figure 1: *Schematic representation of the phases leading up to the standards implementation process. It is intended as a means to identify and locate causes of implementation problems.*

Apart from distinguishing more incidental and more structural causes for lack of interoperability, we include the four main categories of figure 1 in our initial taxonomy, i.e.: the two states of a standard (idea and specification), and two main processes (standards and implementation process). The cause of implementation-related problems differs per category.

Conceptual idea of standard. For example, under certain circumstances the conceptual idea that underlies a standard may not work satisfactorily when implemented (e.g. the scaleability of Java and OSI's comprehensiveness), which would be a reason to adapt the standard and jeopardise the interoperability of implementations.

Standard process. For example, the formal bodies' ideal of consensus decision-making and implementation-independence affect the standards process and indirectly the implementability of standards. Consensus and a pressure to deliver quick results sooner lead to political compromises that are technically ambiguous.

Standard Specification. For example, different use of terminology leads to problems of interpretation, implementation and interoperability.

Implementation process. For example, modest user requirements and cost-constraints often lead to partial standards compliance. This creates incompatibility among implementations.

Table 4 summarises the main elements of our taxonomy. The term *locus* does not refer to a material locus but to the states and processes of transition between the initial idea of developing a standard and the standard's implementation.

In sum, with the proposed taxonomy causes of problems are categorised (1) as being located at/in the conceptual idea of a standard, standard's process, standard's specification, or the implementation process (locus), and (2) as having a more incidental or structural nature (temporality). It is a tool to structure discussions about manifest, concrete implementation problems as well as more fundamental issues. As an illustration, in table 4 it is used to locate the fundamental standardisation dilemmas discussed earlier.

Temporality **Locus**	**Incidental**	**Structural**
Conceptual Idea of Standard	-	Comprehensive standard or a simple one?

Standard Process • **Institutional Causes**	-	Consensus or implementability? Implementation-independence or implementability?
• **Other Causes**	-	-
Standard Specification	-	-
Implementation Process	-	Adapt standard to one's own simpler needs or aim for interoperability with other standard-compliant products?

Table 4: *Taxonomy for identifying and localising causes of standards implementation problems. Applied here to locate some main implementation-related fundamental dilemmas that have repercussions for the interoperability of standards implementations.*

Conclusion

One would expect standards to reduce diversity, heighten market transparency and increase interoperability between ICT products and services in the case of compatibility standards. In the case of modelling standards, one would expect them to facilitate discussion about and the integration of subsystems, and the reuse of components. Etc. In reality, matters are not so simple. This paper explores why diversity among and incompatibility between standards implementations arises.

Some features of the institutional setting of standards organisations explain why standards give rise to problems of interpretation and implementation. The formal setting, in particular, has to cope with the dilemmas of the aims of consensus decision-making and implementation-independent standards, on the one hand, and, standards implementability, on the other. The aim of consensus may well lead to political consensus and technically ambiguous compromise formulations; while the aim of implementation-independent standards elicits, as it were, generic solutions with multiple options.

The case studies point to several causes for and types of problems with implementing standards. In the taxonomy which we develop based on these findings, the causes are categorised along two dimensions of temporality and locus. Some causes seem incidental of nature while others seem more structural. This is captured by the term *temporality*. The second dimension is that of *locus*. It refers to the states and processes of transition between the conceptual idea of a standard and the standard's implementation (i.e. conceptual idea of a standard, standard's process, standard's specification, and implementation process).

The taxonomy draws attention to causes in certain loci. In that sense it is an analytic tool. It can be serve as a policy tool where used to structure discussions on likely and unlikely areas of policy intervention for standards bodies and public government. However, it is a 'tool under construction'. Further research is needed to evaluate and elaborated it. For example, a more full-scale inventory of implementation problems should be made, one that includes cases where implementation and conformance testing are brought into the standards process.

The problems demonstrate that standards development and implementation, although conceptually distinguishable, are intertwined in their working. Considerations in both areas cannot meaningfully be separated. That is, a shift in emphasis from pure standards development to the inclusion of implementation concerns is very much needed.

List of Abbreviations

CCITT	Comité Consultatif International Télégraphique et Téléphonique, former standards division of the ITU (now called ITU-TS)
ICT	Information and Communication Technology
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force, develops Internet standards (RFCs)
ISO	International Organization for Standardization
ITU	International Telecommunication Union
OMG	Object Management Group
OO	Object Oriented
OSI	Open Systems Interconnection
RFC	Request For Comments, part of which are Internet standards
SGML	Standard Generalized Markup Language

TCP/IP	Transmission Control Protocol/Internet Protocol
UML	Unified Modeling Language
W3C	World Wide Web Consortium
XML	Extensible Markup Language

References

Allen, P., & S. Frost (1998). *Component-Based Development for Enterprise Systems: Applying the Select Perspective.* Cambridge: Cambridge University Press.

Atkinson, C., Bayer, J., Laitenberger, O., & J. Zettel (2000). 'Component-based software engineering: The KobrA approach.' *The 3rd int. Workshop on component-based software engineering.* Limerick, Ireland.

Booch, G. (1991). *Object oriented Analysis and Design with Applications.* Redwood City, Calif.: Benjamin/Cummings, 1st ed.

Castek (2000). *Component-based development: the concepts, technology and methodology*. Castek company's white paper. http://www.castek.com.

Coad, P. & E. Yourdon (1991). *Object Oriented Analysis*. Englewood Cliffs, N.J.: Yourdon Press, 2nd ed.

Coleman, D., Arnold, P., Bodoff, S., Dollin, Ch., Gilchrist, H., Hayes, F., & P. Jeremaes (1994). *Object-Oriented Development: The Fusion Method*. Englewood Cliffs, N.J.: Prentice Hall.

Dahanayake, A.N.W. (1997), *An Environment to Support Flexible Information systems Modeling*, (Dissertation), Delft University of Technology, the Netherlands.

ECITC (1993). ECITC Guide to IT&T testing and certification. Brussels, Belgium.

Egyedi, T.M. (1996). Shaping Standardisation: A study of standards processes and standards policies in the field of telematic services. Delft: Delft University Press. Dissertation.

Egyedi, T.M. (1997). Examining the relevance of paradigms to base OSI standardisation. *Computer Standards & Interfaces, 18*, pp. 431-450.

Egyedi, T.M. (2001). 'Why Java™ was -not- standardized twice', *Computer Standards & Interfaces, 23/4*, pp. 253-265.

Egyedi, T.M. & J. Hudson (2001). 'Maintaining the Integrity of Standards: The Java Case', in: K. Dittrich & T.M. Egyedi (Eds.), *Standards Compatibility and Infrastructure Development: Proceedings of the 6th EURAS Workshop,* 28-29 June 2001, Delft, The Netherlands, pp. 83-98.

Egyedi, T.M. & A.G.A.J. Loeffen (2001). 'Succession in Standardization: Grafting XML onto SGML', in T.D. Schoechle & C.B. Wagner (Eds.), Proceedings of *'Standardization and Innovation in Information Technology' (SIIT2001) October 3-5, 2001, University of Colorado, Boulder, CO, USA,* pp.38-49. Repr. in *Computer Standards & Interfaces, 24,* pp.279-290, 2002.

Findings of Fact (1999). United States of America v. Microsoft Corporation. Findings of Fact, United States District Court For The District Of Columbia, Civil Action No. 98-1232 TPJ.

IETF/ RFC 2026 (1996). *The Internet Standards Process* - Revision 3; S. Bradner, Network Working Group, October 1996. http://www.ietf.org/rfc/rfc2026.txt

ISO (2001). *ISO Strategies 2002-2004: Raising Standards for the World.* ISO/Gen 15:2001. http://www.iso.org/iso/en/aboutiso/strategies/isostrategies2002-E.pdf

Jacobson, I., Booch, G., & J. Rumbaugh (1999). *The Unified Software Development Process.* Reading, Mass.: Addison-Wesley.

Jakobs, K. (2000). *Standardisation Processes in IT: Impact, Problems and Benefits of User Participation.* Braunschweig/Wiesbaden: Vieweg Professional Computing.

OMG (1998). *Unified Modeling Language Specification.* Framingham, Mass.: Object Management Group, Internet:www.omg.org.

Schmidt, S.K. & Werle, R. (1998). *Co-ordinating Technology. Studies in the International Standardization of Telecommunications.* Cambridge, Mass.: MIT Press.

SGML (1999). ISO 8879:1986/Cor 2:1999.

Shlaer, S., & J.M. Stephen (1988). *Object-Oriented Systems Analysis: Modeling the world in data.* Englewood Cliffs, N.J.: Yourdon Press.

Söderström, E. (2002). *A Certification Instrument for Standards Implementation*, Thesis Proposal, technical report HS-IDA-TR-02-002, Department of Computer Science, University of Skövde, Sweden, April, 2002.

Souza, D.F. D', & A.C. Wills (1999). *Objects, Components and Frameworks with UML: The Catalysis Approach.* Reading, Mass.: Addison-Wesley.

Stojanovic, Z. & Dahanayake, A. & Sol, H. (2001), A Methodology Framework for Component-Based Systems Development Support. Proceedings of the 6th CAISE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design.

Uniface (2000). *Compuware Corp. Uniface products*, http:// www.compuware.com/products/uniface/.

W3C (1998). *Extensible Markup Language (XML) 1.0.* W3C Recommendation 10 February 1998. Editors: T. Bray, J. Paoli & M. Sperberg-McQueen.

Stojanovic, Z. & Dahanayake, A. & Sol, H. (2001), A Methodology Framework for Component-Based Systems Development Support. Proceedings of the 6th CAISE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design.