Issues in Open EDI

Jari Veijalainen

Technical Research Centre of Finland, Laboratory for Information Processing
Lehtisaarentie 2A, SF-00340 Helsinki, Finland§
email:veijalainen@tik.vtt.fi

§Current address: GMD-IPSI, Dolivostr. 15, D-6100 Germany.

Abstract

Electronic Data Interchange (EDI) is gaining more and more importance in business and administration, as big savings are in sight. It is made plausible that autonomy is the key issue in EDI environments. The globally and locally controlled autonomy domains must be separated from each other both conceptually and in implementations. Based on this observation, an examination is made of how the work within EDI environments should be organized. An autonomy preserving EDI systems reference architecture is proposed, the need for which was identified by the special working group of ISO (SWG-EDI). It consists of two-components. It is argued that future standardization work should include automatic scenario development tools, like a machine-executable scenario language, which would be supported by Open EDI systems

1 Introduction

Investments in information technology have increased in almost every sector of the economy and a vast variety of applications, equipment, databases and communication software already exists, representing tremendous economic value. Modern societies are today dependent on information technology.

The existing information systems and existing and emerging new communication facilities together make it possible to exchange data between the computer systems of different organizations or individuals for the purpose of *cooperation*. This kind of activity can loosely be called Electronic Data Interchange (EDI), especially if it happens through a computer network. It has the potential to reduce the cost of managing the *information flows* between different organizations, and indeed, it is rapidly increasing in practice. The trend towards computerized cooperation based on EDI is strong, for instance, in international trade, as the emerging standards and systems for computer-aided trade show. The Commission of the European Community has estimated that the usage of EDI will increase by 120 per cent per annum during 1990 – 1992. EDI will be especially important in 1993, as the uniform European Economic Area with eighteen countries will have been implemented [23]. In the U.S.A., the federal government has already begun a national EDI implementation program with the goal that all Federal agencies will be able to communicate with EDI by 1996. The US Health Care Finance Administration has received congressional approval for requiring all doctors and hospitals to submit federally subsidized medical claims electronically via EDI standard formats [3].

The term EDI has not been too well defined. A special working group of ISO [20].defines the term Open-edi, as *electronic data interchange among autonomous parties using public standards and aiming towards interoperability over time, business sectors, information technology systems, and data types. The data exchanged can be any predefined, structured data, processable by autonomous applications at both ends.* Another definition, slightly modified from that given by ANSI X-12 says: *EDI is the exchange and* ***integration*** *of routine business transactions in a computer-processable format, covering such traditional applications as inquiries, planning, purchasing, acknowledgements, pricing, order status, scheduling, test results, shipping and receiving, invoices, payments, and financial reporting* [3].

The preservation of autonomy is pervasive in the EDI context and causes directly or indirectly the special organizational and technical problems. Thus, autonomy and integration are really the key issues in inter-organizational environments and their inclusion into the definition of EDI is mandatory.

The main goal of this paper is to clarify the aspects and implications of autonomy, and the resulting architectural issues in standard open EDI systems. We discuss autonomy and heterogeneity in section 2. Section 3 is devoted to the essential problems of open EDI systems. Section 4 contains the components of the reference architecture, section 5 the conclusions and recommendations for further standardization.

0-8186-2697-6/92 $03.00 © 1992 IEEE

2 Autonomy in EDI Environments

Autonomy has lately become an issue in distributed systems design. The reason is that distributed information systems have been developed that span several organizations or relatively autonomous units within an organization. These environments require understanding of what autonomy is and how to cope with it. This is especially crucial when designing EDI systems.

In the communication area the approach stated in the ISO/OSI Reference Model [11] and the standardized protocols developed within its framework are an important contribution to the understanding of autonomy. The S.W.I.F.T. network was one of the first value-added networks [21]. It spans indeed the globe, having been operational since 1977. It introduced the first tools, standardized message types, to cope with autonomy and heterogeneity in an EDI environment.

In the database field, autonomy has been recognized as an issue in the context of distributed databases, notable R* [19], multidatabases [15], and federated databases [10]. It was analysed further in Europe at least in the MAP761 project [6, 16, 21] and in the ANSA project [2]. At the same time, autonomy was addressed within some projects in the U.S.A. as reported e.g. in [1, 5, 8]. Autonomy is recognized as an important issue by the ISO/Special Working Group on EDI [20].

When defining autonomy more accurately, we must decide what it is attributed to. Since we need to cope with such different things as organizations, human beings, computers, software modules, databases, etc., we consider autonomy of *entities* [22]. In general, an entity is anything that can be distinguished (by a human being) from its environment – which also consists of entities. An entity has a particular *structure* which determines how it *could behave* during its existence. For instance, in a computer system certain hardware and software are required in order for it to communicate with other computers or human beings in an "open" way; a lion has organs that enable it to hunt other animals and digest their meat, but it never can fly like a bird because its structure simply does not enable it to do so. In this sense the structure is the "internal limit" of the entity.

The structure can be modeled as the set of all *possible futures* of the entity. During its existence an entity then behaves in a particular *actual* way, choosing among alternative possible behaviors.

Some behaviors contain visible actions and are thus *interactions*. The environment might *control* the entity through interactions. In our view, autonomy is the opposite of *controllability* of the possible or actual behaviors of the entity, from the point of view of the entity itself or of the environment. The real *self-controlling potential* of an entity with respect to its own possible futures is called *self-determination*. On the other hand, an entity whose *actually exhibited* particular behavior deviates from the externally determined criteria is called *effectively autonomous*, with respect to the criteria. An entity is *externally controllable* if it is *not* effectively autonomous. – A more detailed discussion of the autonomy model can be found in [22].

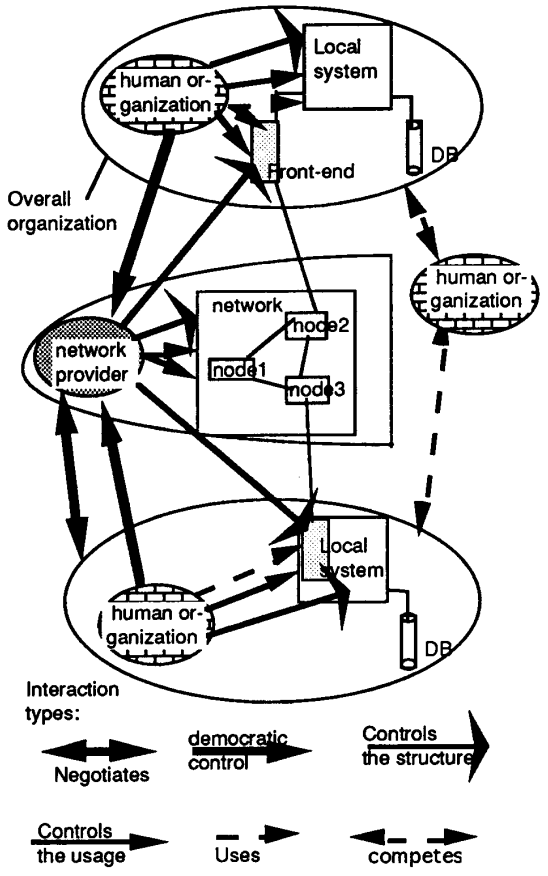

Fig.1. The model of the environment of concern.

2.1 O-autonomy

The main entities and interactions in an autonomous EDI environment are shown in Fig.1. A *human organization* consists solely of human beings. The corresponding *overall organization* consists of the human organization and *computers and telecommunication facilities*, whose usage is solely controlled by the human organization and which support EDI with other organizations. We show in Fig.1 only four types of interactions between organizations, namely *negotiations, democratic control, competi-*

tion, and *repeated cooperation*. These are the most interesting from autonomy point of view in this case.

Any overall organization that can be distinguished from its environment has *organizational self-determination*. It means that an organization has real alternatives to choose from in its daily behavior, even though other organizations try to influence it. This influence can be realized through direct interactions, as is the case between a negotiating or cooperating contractor and subcontractor, or indirectly – e.g. when companies compete in the market place.

There are many different kinds of interactions between organizations and the *degree* of self-determination must be considered with respect to all of them. Organizational subunits have a smaller degree of self-determination than the main organization.

Effective O-autonomy means that an organization, say X, does not behave in a way another organization, say Y, expects it to behave in a single interaction, in a sequence of interactions, or in a situation when Y refuses to interact with X.

2.2 D-, M-, and T-autonomy

EDI systems, like all computer systems, are used by organizations to satisfy their information processing needs. Since organizations are the basis for the existence and usage of computers and all kinds of EDI systems, O-autonomy, is the basis for autonomies attributed to computer systems.

Because computers do not have intentions of their own they do not start activities by themselves. Rather, some part of the human organization controls the overall structure and behavior of computers through two different types of interaction, namely *controls the usage* and *controls the structure*, as shown in Fig.1. The former is an interaction type with the help of which the organization decides who is entitled to use the computer system, what access rights the users have, which services are offered, and how much must be paid for usage. The latter interaction type models the activity where the organization chooses and/or develops the computer systems' hardware, software and applications that meet its needs.

The two controlling behaviors above might be fully self-determined, effectively controlled by another organization, or something in between. The decisions concerning the usage become effective through *controls the usage* – type of interactions. We say that an organization is *M-autonomous* (management autonomous) with respect to a computer system (or part of it), if it has full organizational self-determination as it enforces the usage policy of that system[1]. Usually, M-autonomy requires that the organization owns or has leased the computer system in question.

M-autonomy domain of an organization consists of those computer systems (or their parts) with respect to which the organization is M-autonomous.

Effective M-autonomy means that an organization indeed makes such decisions concerning the usage of its system that are contrary to the expectations of another organization. Note that an organization is normally effectively M-autonomous towards *foreign* organizations, but might never act against the will of the headquarters, concerning the usage policy of a system.

We say that an *organization is D-autonomous* (design autonomous) with respect to a computer system (or a part of it) if it has full organizational self-determination with respect to *controls the structure* –type of interactions (see Fig.1). In other words, a D-autonomous organization has a number of real alternatives at its disposal when choosing the hardware and software of that system, when developing the conceptual and physical schemata[2] for the database system, when developing databases and EDI applications run in that system, and so on.

D-autonomy domain of an organization consists of those computer systems (or their parts) with respect to which it is D-autonomous. An organization may be D-autonomous with respect to one (part of a) system but not with respect to another, even if it is M-autonomous with respect to both of them (leased systems). Moreover, an organizational subunit, like an affiliated company, may be D-autonomous with respect to its computer systems, although its self-determination can be restricted in some other respects by the headquarters. Thus, M– and D-autonomy domain of one organization do not necessarily coincide. Rather, they might be partially overlapping or one could be a subdomain of the other. This is why speaking about autonomy in general is often confusing.

When a cooperative EDI system is set up, the *D-autonomy of a system* plays the main role. We say that a (part of a) computer system is D-autonomous with respect to an organization if the organization is *not* D-autonomous with respect to the system. In other words, an organization must be, for some reason, incapable of fully controlling the structure of such a system that is D-autonomous towards it.

Often, the reason for the D-autonomy of a particular system with respect to an organization is that there is another organization that is D-autonomous with respect to the same system. The latter might refuse to choose or develop a computer system structure that the former organization expects from it in negotiations. The reason for such

[1] In [24] a somewhat similar definition was given but the term used was "organizational autonomy". Within Internet an autonomous system is also mentioned in the context of routing. It is "a group of gateways and networks controlled by a single administrative authority" [4, p. 166].

[2] Litwin et. al., for instance, discuss four sub-aspects of D-autonomy (name independence, data duplication, data restructuring, value type autonomy) under the name "data definition autonomy" [15].

a behavior is usually that the expected structures (e.g. protocol software or hardware, database system changes) are not in the *interest* of the latter, D-autonomous organization, or the required structure is too expensive or otherwise impossible to implement.

In addition to the above interactions with a computer system, an O-autonomous organization often also develops conceptual models, performance models, and other similar models. These are used in the overall system design process to abstractly represent conceptual schemas, database schemas and applications, protocols, and the like. For these purposes, different kinds of software engineering and other *tools* are used. We say that an organization is *T-autonomous* (tool autonomous) if it has full self-determination when choosing and using the tools in the system design and evaluation process.

The *T–autonomy domain* of an organization consists of the tools with respect to which it is T-autonomous. If these tools have computer supported parts then they belong, by definition, to the M-autonomy domain of the organization.They might or might not belong to the D-autonomy domain of the same organization.

2.3 C-autonomy

In computerized cooperation, computers are used as a technical means of cooperation. The information exchange needed in cooperation is (partly) implemented by communicating computers of the organizations. A manifestation of effective M-autonomy is that the organization wants to regulate when its system is reachable through the network, when it is detached, or even shut down. From the network point of view, such a system either communicates or does not communicate at all during a certain period of time.

We say that a system S is *C-autonomous* towards another system S´ if the latter cannot force S to start, stop, or continue communicating[3] with an arbitrary system S". System S is *effectively C-autonomous* towards another system S', if S SOMETIMES[4] refuses to start or continue communicating with S'.

If two systems are not at all willing to communicate at the same real time, they cannot communicate at all, or one or both must adapt their communication behavior. This means actually reduction of C-autonomy (and M-and O-autonomy, which is the actual problem). This can be avoided if at least one system within the environment (network) is not effectively C-autonomous. Such a value–added network, like S.W.I.F.T. I [21] supports the effective C-autonomy of the participating systems and stores the messages until the receiver fetches them. This is typical of EDI environments.

Supporting C-autonomy requires mechanisms that are designed with this goal in mind. The CCITT X.400 recommendation [25] is a good example of such protocols that support maximally C-autonomy of the users. Also, in other ISO/OSI protocols the responding system can always refuse to set up a connection, or break an existing one, by issuing the corresponding protocol message. The latter are examples of mechanisms supporting (effective) C-autonomy. The consequences of, and measures for, C-autonomy are dealt with more thoroughly in [22].

2.4 E-autonomy

The information in EDI is exchanged through individual messages transported between autonomous organizations. The whole repeatable cooperation is an instance of a **business protocol**. Its *definition* contains protocol message types, their business semantics, and their acceptable relative orders at each organization. An expression of effective O-autonomy (and M-autonomy) is that an organization does not have to *execute*[5] all messages it receives, in the same way as it does not have to react positively to a letter or telex it receives. We say that a system has *E-autonomy* (execution autonomy) with respect to an EDI protocol message type if it can choose whether to execute the incoming message instances or not, and how fast they are processed.

Effective E-autonomy means making use of the possibility of not executing a message. An E-autonomous system can refuse to execute an incoming EDI protocol message for a variety of business or technical reasons, like mistrust of the business partner, erroneous information in the message, authorization failure, overload of the system – or because the business protocol does not always require further steps (as e.g. an offer to buy or sell currency). It can also start the execution and even request other systems to completely perform subtasks, but later still come to the decision that it is not possible to execute the message.

The implications of effective E-autonomy become clearer when considering e.g. the 2PC protocol [9]; if a participant system really refuses to execute COMMIT or ABORT message sent by the Coordinator, the protocol does not achieve its goals. The 2PC protocol is also not very viable in autonomous environments because the overall Open-edi transactions, like banking transactions, can last hours or even weeks. Blocking pertinent data of an organization for a long period of time blocks the local users from using the data, which might substantially reduce the self-determination of the local users.

[3]By communication we mean in this context the exchange of protocol messages, defined for an OSI layer, between two computers. The information disclosure aspect [8] is here not of relevance.

[4]This is a fuzzy quantifier, see e.g. [22].

[5]Executing a message means taking all measures necessary to carry out the request conveyed by the message instance.

2.5 Autonomy versus incorrect behavior

The example above brings up the question of what is the relationship between "erroneous" and "autonomous" behavior of a computer system? First, if erroneous means a qualitative deviation of the system behavior from its overall specification (2 + 2 = 5 type errors, visible protocol errors), then the behavior cannot be accepted by the organization using the system; It must be repaired.

Another view on errors is quantitative. Different values of a numerical measure might be criteria for the "erroneous" and for the "autonomous" behavior of a system. A fuzzy measure [22], for instance, can be used. Then, either it holds that effectively (C– or E–) autonomous behavior is erroneous, or there is a degree up to which it is acceptable, according to these criteria. An example of the latter possibility is characterized by a statement "if a system does not communicate for three days it is effectively C-autonomous, but if it does not communicate for a month, then it is erroneous".

The relationships between D–, C–, and E-autonomy and correct behavior have been further discussed in [22].

2.6 Permanence of heterogeneity and autonomy domains

An EDI system consists of many *component systems.* D-autonomy of participating organizations with respect to the *local parts* of component systems managed by them implies that the local systems are D-autonomous with respect to global bodies and other participating organizations. Therefore there is neither single point of control nor coordination when developing them. As a result, the overall EDI environment is usually *heterogeneous* according to many criteria; different local hardware and software, dissimilar database management systems and database schemas, contradicting consistency constraints embedded in application programs, diverse communication software, conflicting naming of data objects, incompatible tools, and so forth.

Because the component systems are controlled by *distinct* D–, and M-autonomous organizations, an overall system supporting EDI *crosses organizational borders.* The O-autonomous organizations will make use of their T- and D-autonomy by constantly developing the component systems, structurally controlled by them, as they like. Therefore, for any EDI environment, *the heterogeneity between component systems is likely to be a permanent state of affairs.* Consequently integration principles, global tools, architecture models, services, and so on, must cope *explicitly* with different kinds of heterogeneities and autonomy aspects in order to be applicable for a longer period of time.

From the technical point of view, EDI requires that different component systems are *interoperable* [15] or that they are somehow "integrated" to form a larger system (c.f. the definitions of EDI in the introduction).Since heterogeneity is permanent "integration" in an EDI environment means at least *overcoming heterogeneity* between component systems, for the *purpose of cooperation.* The only reasonable approach is to base the integration on *a homogeneous global (autonomy) domain* that crosses the borders of the participating overall organizations. Preservation of O– and D-autonomy of the participating organizations requires that the global domain is *globally controlled.*

3 Organizing EDI

3.1 The global EDI designer

What should the global homogeneous domain technically look like in order to preserve the different autonomy domains and to "integrate" them? Historically, means to overcome heterogeneity have been the first target of *standardization* in EDI environments. Indeed, in any EDI system the global domain is "standardized" in some sense. In different approaches both the global domain controlled and the nature of the controlling body are different. The main problem is in any case how *to cross the border* between the local and the global D-autonomy domain. The second question is, what kind of *organizational basis* is needed to successfully develop and implement such a global domain or such global domains.

In an EDI environment the main expression of organizational self-determination is that, in general, organizations *cannot be forced* to accept technical or other solutions pushed by other organizations. Such solutions should rather support cooperation among the participants, should be of benefit to all of them, and preserve maximally their O–, D–, M–, and C-autonomy. Thus, common homogeneous technical solutions can be developed only *in cooperation* by the O-autonomous organizations having interest in EDI. We call this the **cooperation principle.** This cooperation can be organized either directly among the (few) partners, through a separate company (like S.W.I.F.T. with over 2000 partners), or a public (standardization) body which is "democratically" controlled by the interested parties. The latter case is shown in Fig.1. The abstract cooperative body that controls the structure of the homogeneous domain is called a *global EDI designer.*

Let us look at some problems. In the real world there are many separate global EDI designers that control homogeneous domains of different *business areas.* This organizing principle is natural, because it is based on the *common interests* of the participants that for some reason interact with each other. EDI influences, however, also other

interests. Because *conflicting* business and other interests are often involved when setting up an EDI system or designing standards, any global EDI designer is *primarily political* rather than technical in nature.

Whose interests should EDI serve? In general, each participant would like to see global homogeneous solutions that minimize its costs and maximally support its business. From the national economy point of view, the savings in small businesses[6] are especially important, since these firms generate the majority of wages and gross national product [3]. Small savings in a large number of companies help in creating a competitive national economy. In practice, the business interests of users differ from each other and are different from those of the computer manufacturers. Different countries have different needs and interests, teleoperators compete with computer manufacturers, and so forth. Organizations have quite different degrees of self-determination with respect to the homogeneous solutions. Weak organizations (and countries) have to adapt themselves to what the strong ones enforce, no matter whether the EDI solutions are "Open" or not [3]. We call this complex the **interest reconciliation problem.**

It is not evident that the interest reconciliation problem has a unique solution. This would require global homogeneous EDI standards or solutions that satisfy the current needs of *all* business sectors, administration, research, and so on. The interests of different (business) sectors might simply be too far apart from each other. And even if really "general" solutions exist, "global" EDI standards developed by (especially) international, impartial standardization bodies might be rejected by the organizations supposed to apply them if the bodies fail to recognize the real interests of the organizations involved. Owing to the O-autonomy of the parties, ignoring standards is always possible.

Because of the diverging interests involved and the practical impossibility to solve all problems by one body, the existence of several global EDI designers seems unavoidable. This leads to "islands of EDI" [20] which are incompatible with each other. The problem is the same as in the case of the individual organizations, the difference being that the global EDI designers are now D-autonomous as they control the different global autonomy domains.

Because it is not possible to set up a single global EDI designer controlling a single global domain, the most common task is rather to *coordinate* the work of different global EDI designers, where necessary. Then, "the real pay-off in EDI lies in the development of bridges and commonalities in infrastructrue among current islands of EDI" [14]. We call this the **global EDI coordination problem.** The coordinator can be only a *single global standardization body*. It must provide the organizational and technical support, especially suitable standards and tools.

The technology and interests evolve over time and solutions and standards now feasible will probably become obsolete in the future. The only question is whether this happens in the next 10 or 100 years. This means that the EDI standardization is a *continuous process*. The evolution of interests and technology is closely connected to the question of optimal timing when introducing new generation of EDI standards. Standards dealing with interoperability of systems will have an impact reaching decades far into the future. Premature or bad standards will prohibit proliferation of good technical solutions. Too late a standardization tends to lead to a turbulent era with competing solutions and islands of EDI. We call this the **problem of adequate timing.** It is a separate issue of what good standards should encompass at different times. It is discussed below from the contemporary perspective.

3.2 The work of ISO

The Special Working Group on EDI of ISO (ISO/SWG-EDI) can be understood as a first step towards The Global EDI Designer (c.f. 3.1). Its goal is "development of a conceptual model for the identification and coordination of existing and future standards and services for furthering global interoperability of electronic data interchange". The working group addresses all aspects of *Open-edi transactions* [20]. These consist of *edi-messages* between *autonomous parties* involved in such transactions[7]. "Open" in this context means compatibility and harmonization with the OSI model; generic, public, interoperable, non-proprietary standards, testable with objective test criteria, conformance testing and certification; and independent of specific business applications or industry sectors, fostering cross-industry sector EDI utilizing ISO standards [20, p. 26].

The working group identifies several targets for standardization. The first one is the *conceptual model* covering all aspects relevant to Open-edi transactions. The second one is a *reference model* that essentially defines a standard architecture for an Open-edi system, in the same spirit as the OSI Reference Model was defined. The third target are the tools to describe (standard) *scenarios*, i.e. (topological) structures of Open-edi transactions. Fourth, the working group addresses a set of existing and new standards (e.g. EDIFACT) that are needed to describe the information objects exchanged between organizations and the

[6] A company with fewer than 500 employees.

[7] This definition is recursive and quite imprecise; it is not clear whether the parties are within the model or not. What the "edi-messages" are, is not specified. One gets the impression that the focus of the working group is on open services and protocols whose topology is a general graph.

(communication) standards covering their transport through communication networks.

The working group identifies seven work items. They are (1) Open-edi reference model (c.f. above), (2) Business agreement services (full description of universe of discourse for the business in all semantical aspects), (3) EDI support service standard(s) (specification of IT services supporting Open-edi within the Reference Model), (4) Requirements for amendment to and/or addition of non-edi-specific standards (determining, whether and how the above services can be supported by the existing non-edi related standards), (5) Usage specification for non-edi standards (how the existing standards should be used), (6) Requirements for amendment to and/or addition of edi-specific standards (c.f. (4)), and (7) Usage specification for edi-related standards (c.f. (5)).

The strong point in the outcome of SWG-EDI is that the scope of the global domain has been identified to be larger than only the "mechanistic" aspects of interchange (i.e. syntactically homogeneous messages). The working group also addresses the business protocol level in the form of *scenarios*, as well as aspects of organizational autonomy (legal aspects, business-making capability), and the global EDI coordination problem (see 3.1).

The work of SWG-EDI is a big step into the right direction. The main problem seems to be that no aspects of autonomy are directly addressed nor modeled, i.e. there is no clear *border* between the global homogeneous domains and the diverse locally controlled D–, T–, and M–autonomy domains. This affects the target of global standardization, the form of the global tools, as well as the interfaces between the global and local domain[8]. Therefore, e.g. the *border crossing principle* (see below) cannot be formulated nor used.

We will analyze existing domains using the conceptual framework presented above and deduce requirements and principles that could be used in the further EDI standardization work.

3.3 Analysis of some existing global domains

Historically, one of the first homogeneous domains was the S.W.I.F.T. I system [20] (c.f. Fig.1). This domain is worldwide but not global in the sense that it is only used in banking. In the banking environment, the tools for overcoming heterogeneity are the *message types*.

[8]Actually, the approach of SWG-EDI is the same as in the ISO/OSI Reference Model. In it, *the functionality* of a system is specified by the standardization body in terms of its external behavior in message exchanges (or through an abstract state automaton), but the *concrete implementation* is not fixed. The idea is that the degree of local D-autonomy of an organization with respect to the design and implementation of the communication subsystem should remain maximal.

EDIFACT (Electronic Data Interchange for Administration, Commerce and Transport) is the current state of the art in standardized EDI, accepted in 1987 by ISO [13]. It is a part of the Trade Data Interchange Directory, issued and maintained by the United Nations Economic Commission for Europe, UN/ECE. It is a *tool* (grammar) with which minimal homogeneous domains can be defined by different EDI designers in different business sectors.

Using EDIFACT the homogenous transfer syntax of the data to be interchanged can be described. The actual objects of standardization are message types. The smallest identifiable and typed unit is the *element*. Of them, *data segments* are compiled. Several segments form *messages*, and several messages can be included into one *interchange*. In real implementations, an interchange instance corresponds quite naturally to a usual file and a message to a record with a variable length and structure.

EDIFACT alone does not solve the problem of the global EDI coordination (see 3.1): there are several business sectors, that use EDIFACT but the message types are partially incompatible owing to semantically overlapping but syntactically differing definitions of segments [20].

In order for EDI systems to be interoperable with other EDI systems there must be some *real software and hardware* that makes this possible. They can only be *added* to an existing system or, as an alternative, a complete new system is installed. Otherwise the existing local D-autonomy domain would be reduced. This leads to the **EDI architecture problem.**

The architecture must cope with questions like how the interchanges are transported and how the homogeneous data in the interchanges are "mapped" into heterogeneous local environments and vice versa. Conceptually, the standardized EDIFACT messages form a *global abstract specification* of the *interface* between the global and local autonomy domains. The local EDI designers then design the *mapping* between this specification and the input and output formats of the local EDI applications. This as such is a general principle: the interface between the global and local domain is *globally (abstractly) specified*, and *locally (concretely) implemented*. We call it the **border-crossing principle.** It seems to guarantee maximal D– and T-autonomy for both local and global EDI designer.

The architecture problem has been addressed in Finland by defining a simple reference architecture, the "EDI/OVT model" [18]. It distinguishes between four different functional components: the *transport interface*, *translator*, *application interface*, and *management component*.

The management component controls the overall flow of data between the other components. The translator is guided by the individual EDIFACT message descriptions agreed upon by a global EDI designer active in a business sector. The translator is a piece of general-purpose software, isolated from the communication software used and

from the applications by the corresponding interfaces. It checks and transforms the interchange contents from the EDIFACT syntax to a local syntax and vice versa.

The concrete problem is then how the local data and mappings to the global format and vice versa are described for the translator. Because of the heterogeneity of different local systems, there cannot exist only one mapping. The most general idea is to offer a special *mapping language* for describing the mappings between the global standard and local formats [7]. This language does not need to be directly related to EDIFACT. It must be learned and used by the local EDI designer, since only it knows the local data syntaxes used by the local EDI applications, as well as the homogeneous interface – the global message types.

Neither the EDIFACT standard nor the existence of the transport interface fixes the communication service. Thus, the services and protocols used in transporting the interchanges must be negotiated among the EDI users. Different choices for the (OSI) protocol stack lead to numerous "(EDI) communication islands", even if all parties use EDIFACT to describe the interchanges. Thus, the scope of the EDI/OVT model above is too limited. Still, the separation between communication and EDI worlds should be preserved.

The homogeneous part of an EDI component system resides within the global D-autonomy domain. It must be interfaced with the existing software (and hardware), residing within the local D-autonomy domain. In the EDI/OVT model the application interface is a **real interface** between the global and local D-autonomy domain. It specifies, in locally available low-level concepts, *how* the general purpose translator is interfaced with the local applications. Such a real interface is needed, if one wants to develop general-purpose EDI software. If the real interface between the global and local domain is not explicitly addressed by standardization bodies, it varies from component system to component system. This easily causes unnecessary costs, since the local EDI designers have to develop the interfaces and changing them is not easy. The lack of such a standard interface might also make difficult to develop second generation standard EDI products.The real low-level interfacing technique depends on the technology available. It should be based on facilities available in *each* local environment and it should preserve D-autonomy at both sides of the border. Today, only files and databases are used for this purpose.

A further inherent problem with EDIFACT is that it does not support description of the (business)*semantics* of the elements, segments, or messages. The *overall business protocol,* the binding of different message types into a scenario, remains also implicit [20]. Thus, both must be given with a natural language, which easily causes problems for common understanding. The problem is aggravated by the need to describe autonomy characteristics of the business protocols.

The goal of addressing business semantics requires not only external system behavior to be specified, but also *local data and operations* to be addressed in one way or another. Because of the heterogeneity of local systems, they cannot be directly referenced, but rather an *abstract model* must be designed – like "The Virtual File Store" in FTAM [12]. The next problem is defining the relationship between the model and a real system. What is a correct behavior of the real local system with respect to the abstract model? If there is no suitable technical basis on which the global semantics can be based, it is difficult to achieve anything more than with the current means. The border-crossing principle says: the specification of any interface between the global and local D-autonomy domain is a global task but the implementation is a local task. We suggest that the scenarios [20] should be based on the Abstract Local Interface (ALI) [22] which resides between the global and local D-autonomy domain. The abstract global (operational) semantics is attached to the interface objects which are *Abstract Local Operations* (ALO) manipulating abstract data. The global data appears as input and output parameters of the operations. The local real data and real programs are used to implement the abstract operations.

What should be standardized? It seems that both tools to describe the scenarios and the scenarios itselves should be targets. The design of the common tools falls within the domain controlled by ISO, as well as some very general scenario structures (generic business protocols). The different global EDI designers are responsible for developing particular scenarios with these tools. The tool would guarantee interoperability by *automatically producing* the messages and segments on the basis of input data.

The central tool is a **scenario language** that is rich enough to represent the abstract operations above, the computation and control flow, the input data and (generation of) the messages. For compatibility reasons, the EDI messages sent could be syntactically mapped into existing EDIFACT message formats, if such exist. Thus, the old and new EDI systems would remain interoperable. From this it follows that the scenario language should have some relationship with EDIFACT. It should also be implementable in various systems, so that the scenarios would become widely executable.

Nowadays designing new message types is mainly paper work performed by committees. If the scenario language is combined with a (graphical) user interface at each EDI system, the global EDI designers can be *established and supported* by the global EDI system. They can then design new interoperable domains, using the system as a distributed (CSCW) tool. In this way, the process of designing new domains can be automated much further than now, the design process being supported by one or a few

special scenarios. This might also be a solution to the global EDI coordination problem; The scenarios could be sent to ISO, using any standard EDI system, to be tested and registered. They could also be tested by an emulator for internal and external conformance (c.f. [20]).

4 Reference Architectures for Open EDI

As a conclusion from the above considerations we present a Communication Reference Model and a System Reference Architecture. The former represents a pure OSI view on the open EDI systems architecture. The latter is an abstract implementation model that addresses the reconciliation of the D– and T-autonomy domains.

4.1 The Communication Reference Model

The view that an Open EDI system is a distributed, layered system with a certain open [11] service and protocol structure is given in the Communication Reference Model, depicted in Fig.2. We assume that the data transfer service is based on the OSI stack with P1 and maybe P2 protocol on top of it [25], because this arrangement maximally supports C-autonomy of component systems. Any other suitable protocol stack could be used, however, as long as it provides an end-to-end data transfer service. It is certainly a point of discussion which stacks should be allowed, or if any should be excluded. A reasonable requirement for such a stack is that it can be used by several applications, not only by EDI.

The next level service is EDI point-to-point service (EDI-PP-Service). This layer is needed because the quality of (transfer) service differs in different stacks. Also, the data might be transferred through several different networks, and without this level there are no end-to-end acknowledgements. The service provides for a homogeneous naming scheme of partners, and a reliable, homogeneous order-preserving end-to-end data transfer between component systems. The service data elements are only chunks of data with addresses, so it does not matter whether the data to be sent consists of individual protocol messages or interchanges. The protocol providing the service may vary, depending on which stack is used below it. If the stack is that indicated by X.400, then the protocol is most naturally Pedi [26].

The Scenario layer providing the Scenario service is the heart of an EDI architecture. The Scenario service provides for the communication support of the automated scenarios. A scenario is instantiated at one entity, and several other entities, representing the business partners, are usually involved. A scenario entity (S-E) should be capable of supporting the network topology of standard scenario instances envisaged for Open EDI transactions. Currently it is understood how a tree or a more general directed acyclic graph protocols can be implemented [22], but protocols with cyclic topologies require more investigation.The Scenario service offers all the standard scenario service elements for the EDI entities.

The EDI protocols are from the topology point of view identical with the scenario protocols or are their subgraphs. In the latter case not every S-E interacts with the corresponding EDI entity. The protocol data units of different EDI protocols differ from each other, so conceptually there are several protocols, although they might use the same scenario service elements.

One of the protocols within the EDI layer is the EDI *management protocol.* It supports a decentralized definition and installation of new scenarios by a global EDI designer (c.f. above).

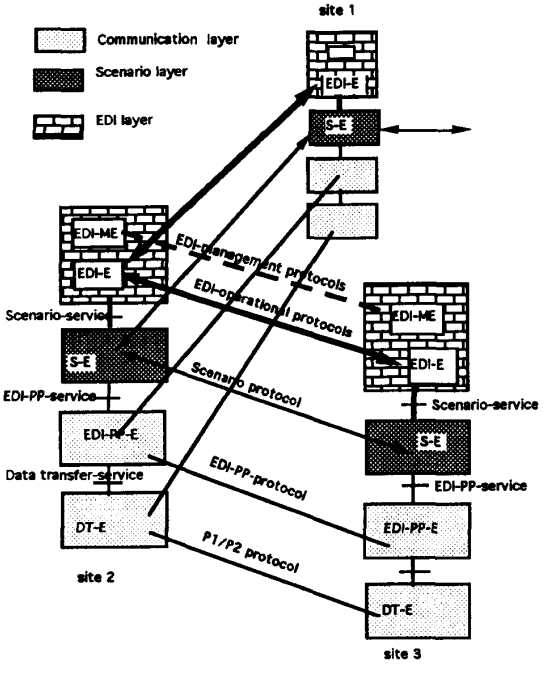

Fig. 2. Abstract Services and protocols in EDI environment.

4.2 The Systems Reference Architecture

The purpose of this architecture (see Fig. 3) is to show the essential components in the EDI environment and their relationships to the global and local D-autonomy domains.

It seems that there are three functionally quite separate parts in an advanced EDI system. Therefore, the reference architecture consists of three layers. They correspond roughly to the three different protocol layers of the Communication Reference Model in Fig. 2. The lowest grou-

ping of boxes represents the *communication subsystem* software, the next higher level the *scenario management software*, and the highest level the *user interface* facilities.

The architecture is hierarchical. It models the idea that new global functionality can be added on top of the existing one. It also incorporates the idea of the OSI model that different architecture layers can be designed and implemented by different global bodies, as long as the service is known and understood in the same way by the standardizing bodies of each two adjacent layers. In the architecture at hand the layers are interfaced by *(abstract) service specifications*. These are produced e.g. by subgroups of the global EDI designer responsible for a special layer.

Following the OSI layering idea, the *functionality* of the global EDI software components is assumed to have been standardized by a standardization body (SE-NW stripes). Each package has then been implemented by a software house (SW-NE stripes, resulting together in cross-ruled boxes). Different packages might have been implemented by different houses. Therefore, they must be separately adapted to each other with additional interfacing components (SW-NE stripes) that obey a standardized*technical interface* (c.f. 3.3). Such interfaces are the Global Communication Service Interface (GCI) and Global Scenario Interface (GSI). One direction could be to specify the real interfaces (c.f. 3.1) within one or several "open" operating system environments, like UNIX™. This is a subject for further study.

To overcome heterogeneity between the global and local D-autonomy domains, some kind of *adapters* are needed. Depending on the functionality of the added software, there is a need for different kinds of adapters. The basic idea of the architecture is that the adapters are interfaced with the same global software interface components as the globally specified parts. Thus, the adapters map both globally specified abstract interface and the globally used real interface into/from a local form. The interaction patterns between LSI and GSI, and LCI and GCI, respectively, are identical.

GL/PPIA (Global/Local point-to-point Service Interface Adapter) offers communication facilities and global/local translator functionality (c.f. 3.2) to a local application. Typically the translator functionality is both ways between EDIFACT – local format, or ASN.1 – local format. Thus, application A1 can be interpreted as a contemporary interchange-based EDI application which accesses the local database producing contents, in local format, for new EDIFACT messages, or processing the translated data arriving in EDIFACT messages. Note that all the different EDI application protocols system interfaces must be implemented in this solution within EDI applications like A1.

The software layer above GCI is called *Scenario Management* (SM). Scenario management is based on the scenario language (SL) that should be specified by ISO. There is an initial formalism in [20], other suggestions including at least [6] and [17]. Automated scenarios are designed using SL and stored by the scenario software. The scenarios are most preferably *programs* executable by the scenario management software, that is essentially a compiler/interpreter of that language. The compiler/interpreter itself can be implemented using any programming language.

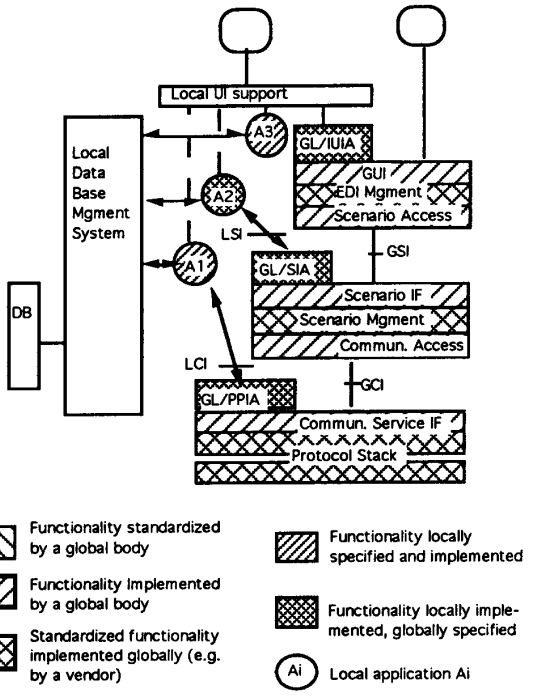

Fig. 3. A schematic system reference architecture.

At the scenario level, the global domain and local domain are interfaced by GL/SIA (Global/Local Scenario Interface Adapter). Most naturally, it offers ALOs for the scenario level (c.f. 3.3). Their name, input and output parameters, and operational (business) semantics are globally specified in the context of one or several scenarios. The semantics of each ALO are locally implemented, using applications like A2. ALOs are only invoked by global scenario software while they perform scenarios, i.e. the *invocation control* is within the global domain, represented by *Abstract Local Operation Manager* (ALOM). It is a subcomponent of GL/SIA.

The difference between GSI and GCI is that the GSI is oriented to support performing individual ALOs, essentially modeling single local steps that belong to larger complete interactions between organizations, as embodied by

the scenarios. GSI thus supports independent invocation of the local, computerized steps of single scenarios, whereas GCI is oriented towards transporting chunks of data between two organizations. At the GCI and LCI, a single interaction consists of handing over a chunk of data in one direction. The overall interaction over the GSI (LSI) is more complicated than over the GCI (LCI). It consists of a *request*, a *response*, and possibly of a *final confirmation*. If it is globally agreed that ALOs should have *transactional properties* then ALOM must have the full functionality of an *Abstract Local Transaction Manager* (see [22]).

A local application can initiate execution of stored scenario programs through the Local Scenario Interface (LSI), provided by a subcomponent of GL/SIA called the Scenario Interface Provider (SIP). The invocation control of scenario execution is within the local domain when the application uses LSI. Note that this architecture allows for local applications that are invoked through GL/PPIA or through ALOM and that further invoke scenarios through SIP. It is a subject for further study whether this is desirable, for instance, when upgrading the current systems to Open EDI systems. The possible interaction patterns between SIP and SM are also for further study.

The top layer then consists of the *Global User Interface* (GUI) facility, Global/local Interactive User Interface Adapter (GL/IUIA), and management functionality. It provides the interfaces for different EDI user categories and facilities to manage the information necessary to control the access to the system. The EDI Management software uses *Global Scenario Interface* (GSI) when accessing the scenarios. Thus, all stored scenarios can be invoked by human beings and new scenarios installed.

It is evident that although the goal of EDI is to minimize the manual work when handling business transactions, it cannot be totally eliminated. Which is then the better way to offer the interface, in the local heterogeneous forms or in a globally standardized form? For instance, people belonging to a global EDI designer can use the overall EDI system as a *design tool*. This interface could well be a global and homogeneous one, i.e. a part of GUI. The use of EDI systems to transmit objects like *electronic forms*, mainly processed by human beings, also requires a user interface.

Should the EDI user interface be homogeneous or heterogeneous? The homogeneous approach would be to specify a *homogeneous generic global user interface* for EDI systems. In this case it should support all kinds of EDI applications. The heterogeneous approach requires GL/IUIA to be designed and implemented. It would save the local users from learning a new interface because of introduction of electronic forms and other new EDI applications. Thus, the local users could use their old tools, like text processing systems or graphical user interface tools. However, before a GL/IUIA can be locally implemented, the above generic global user interface must already exist. Thus, *a global standard user interface for EDI systems should be specified*. As above, there are two aspects that must be taken into account. The abstract user interface specifies for each user category, which interface entities each of them can use and how the entities behave. The real interface describes how the abstract one is technically implemented with certain interface facilities. How both of these sets of interfaces look like, is a subject for further study.

5 Conclusion

This work has been inspired largely by the work of ISO's special working group on EDI. Although autonomy has been understood by the working group to be a major issue in EDI environments, we have pointed out some problems when handling it. The most problematic one is the lack of separation between locally controlled and globally controlled autonomy domains. Our view is that the D-autonomy domains are distinct and reconciled applying the border-crossing principle: something globally standardized or specified is implemented under the local control, i.e. within the local domain. Depending on the architecture and tools available, the reconciliation of the D-autonomy domains takes different concrete forms.

The working group performs a valuable analysis as regards the specification of the globally controlled domain. It is recognized that the EDIFACT is a far too limited global specification tool. The proposed scenarios and means to describe business rules are a long step forward. Still, it is open how these new tools and EDIFACT could be used together. What we propose is to design a complete machine-executable scenario language. Its data part might be compatible with EDIFACT or ASN.1. With this language, the meaning of the scenarios (semantics) could be very precisely given. Nonetheless, some additional tools could be necessary, as required by legal aspects and EDI designer support. The scenario design requires evidently a graphical global user interface.

A further idea presented here is that the EDI environment could be a work bench itself, offering tools for a distributed EDI designer. Behind this is the conclusion that there are and will be a number of autonomous global EDI designers controlling different global domains (business sectors, governments, international organizations, and so on) and standardizing them. If at least the tools were compatible then the scenarios might also be. The distributed tool could even support automatic registration of the results at an international standardization body.

The standardization activity should now concentrate on refining the reference architecture and specifying the necessary scenario language, the tools with which it is handled, and the necessary user/application interfaces.

Acknowledgements

The results presented in this paper were gained in the project *Integrating EDI in open database environments* (EDIT), performed at the Laboratory for Information Processing. The author wishes to thank prof. Esko Heikkilä, for providing funding for the project, and Mr. Tapio Niemelä for chairing the steering committee of the project. The comments of prof. Eero Peltola, Ms. Aija Palomäki, and Mr. Pasi Peuranen for this paper are highly appreciated. The discussions with the team developing electronic forms based on EDIFACT at the Laboratory for Information Processing/Jyväskylä are also appreciated.

6 References

[1] Alonso, R., Garcia-Molina, H. & Salem, K., Concurrency control and recovery for global procedures in federated database systems. IEEE Data Engineering Quarterly Bulletin (Special Issue on Federated Database systems) 10(1987), pp. 5 – 11.

[2] Herbert, A.J. The ANSA Project. In: Mullender, S. (ed.). Distributed systems. USA 1989. ACM Frontier Press. Pp. 391 – 427.

[3] Beltran, C. Small business: the key to the success of EDI in US. Proc. of 3rd Intl. Congress of EDI Users. Brussels, 4 – 6 Sept. 1991. EDI Worldwide. Pp. 398 – 408.

[4] Comer, D. Interworking with TCP/IP; principles protocols, and architecture. USA 1988. Prentice-Hall International. 327 p.

[5] Du, W.& Elmagarmid, A.K. Quasi serializability: a correctness criterion for global concurrency control in Inter-Base. Proc. 15th VLDB Conf. Amsterdam, August 1989. Pp. 347 – 355.

[6] Eliassen, F.&Veijalainen, J., An s-transaction definition language and evaluation mechanism. Berlin(West) 1987. Gesellschaft für Mathematik und Datenverarbeitung mbH, Forschungszentrum für Offene Kommunikationssysteme (GMD-FOKUS). Technical Report "Arbeitspapiere der GMD" Nr. 275/1987. 96 p.

[7] EDISERVER 2.0 Reference Manual. Helsinki 1991. The Finnish PTT/TELE-Edivan. (in Finnish). 273 p.

[8] Garcia-Molina, H. & Kogan, B. Node autonomy in distributed systems. Proc. of IEEE International Symposium on Databases in Parallel and Distributed Systems. Austin, Texas, Dec. 1988. USA 1988. IEEE CH2665. Pp. 158 – 166.

[9] Gray, J. Notes on database operating systems. In: Bayer, R., Graham, R.M. & Seegmüller, H. (eds.). Operating systems, an advanced course. Lecture notes in computer science, Nr. 60. Germany 1978. Springer Verlag. Pp. 393 – 481.

[10] Heimbigner, D. & McLeod,.D. A federated architecture for information management. ACM Transactions on Office Information Systems 3(1985), pp. 253 – 278.

[11] ISO 7492. Information processing systems – Open Systems Interconnection – Basic Reference Model. ISO, 1984. 40 p.

[12] ISO 8571/1-4. Information processing systems – Open Systems Interconnection – File transfer, access, and management. ISO, 1989. ca. 450 p.

[13] ISO 9735. Electronic data interchange for administration, commerce and transport (EDIFACT) – Application level syntax rules. ISO, 1988. 20 p.

[14] Knoppers, J. Transforming cross industry practices into EDI: the business case for scenario modelling. Proc. of 3rd Intl. Congress of EDI Users. Brussels 4 – 6 Sept. 1991. Belgium 1991. EDI Worldwide. Pp. 572 – 584.

[15] Litwin, W., Mark, L. & Roussopoulos, N. Interoperability of multiple autonomous databases. ACM Computing Surveys 22(1990), pp. 267 – 293.

[16] MAP761. Multidatabase services on ISO/OSI networks for transnational accounting, final report. I.N.R.I.A. (ed.). France 1986. 293 p.

[17] MAP761B, Multidatabase services on ISO/OSI networks for transnational accounting, final report of the second phase. S.W.I.F.T. (ed.). Belgium 1989. 216 p.

[18] EDI/OVT recommendations. Helsinki 1991. The Finnish Data Communication Association (STY). (Partly in Finnish). ca 800 p.

[19] Selinger, P. et al. The impact of site autonomy on R*. In: Stocker, P.M., Gray, P.M.D. & Atkinson, M.P. (eds.). Databases - role and structure. United Kingdom 1984. Cambridge Univ. Press. Pp. 151 – 176.

[20] ISO/IEC JTC1/SWG-EDI, information technology – report on the open-edi conceptual model, parts 1 – 2. Technical reports N222-1 and N222-2. Sweden 1991. ISO/Special Working Group on EDI (SWG-EDI). 63 p.

[21] S.W.I.F.T. User Handbook, Volumes 5 and 6 – Standards. Belgium 1984. Society for Worldwide Interbank Financial Telecommunication s.c.

[22] Veijalainen, J. Transaction concepts in autonomous database environments. Ph.D. thesis. Germany 1990. R. Oldenbourg Verlag. GMD-Bericht Nr. 183, ISBN 3-486-21596-5. 358 p.

[23] Weltevreden, P.S. The development of EDI in the European scheme. Stamper, R., Kerola, P., Lee, R. & Lyytinen, K. (eds.). Proc. of IFIP TC8 Working Conference on Collaborative Work, Social Communications and Information Systems (COSCIS'91). Helsinki, Aug. 1991. Netherlands 1991. IFIP/North-Holland. Pp. 261 – 270.

[24] Wiederhold, G. File organization for dabase design. USA 1987. McGraw-Hill. 287 p.

[25] CCITT X.400. Message handling systems recommendations. CCITT, 1984. ca. 300 p.

[26] CCITT X.435. Message Handling Systems. The protocol for EDI. CCITT, 1991. ca 100 p.