Executive summary

The European Commission requested that Microsoft provide specifications to enable third parties to implement “work group server” functionalities both for client/server and server/server interaction equivalent to those of Microsoft work group servers. In response, Microsoft provided the Technical Documentation that is the subject of this Report.

Microsoft work group servers are enormously complex software systems. Tens of millions of lines of source code are necessary to implement them. Their complexity is caused by (a) their large and comprehensive functionality, (b) their required flexibility and reliability in operation and (c) their development history, which makes them systems of enormous size. In addition, in part because of its origin as a proprietary system not intended as an open standard, the Microsoft’s proprietary work group servers implementation uses a particular mixture of open standards and protocols as well as proprietary solutions including extensions of open standards. Typically, for the development documents used inside Microsoft there was no need to separate interoperability information from implementation details.

Producing a rigorous, complete, stand-alone specification for Microsoft work group server functionalities – or any software system of such enormous complexity – that is free of errors and omissions is beyond the state of the art and far beyond industrial practice. The expectation that anyone could do so in only a few increments is simply unrealistic. Rather, normal industry practice is that specifications for such a system are developed and enhanced on an ongoing basis in an iterative process through interaction with engineers skilled in the relevant art and who actually use the specifications.

To make interoperability information for Microsoft work group servers available in any practical manner, the only realistic approach is the pragmatic one Microsoft has taken:

- provide technical documentation (Microsoft has provided more than 12,000 pages of such documentation) that addresses expert users who are experienced developers of client/server system software and familiar with the Microsoft network and programming environment,

- anticipate that users of the documentation will take advantage of the large amount of information about Microsoft work group servers that is already in the public domain, and

- provide live technical support from developers of the documentation to the users of the documentation as the documentation is used and issues with it arise.

Indeed, by normal industry practice specifications for such a large and complex project would always be written for an expert audience and improved incrementally while being
used and made more complete through an iterative process in a dialogue between users and documenters. This is because (1) any Licensee planning to implement such a project would expect to assign persons with the appropriate expertise to the project, and (2) writing specifications that are intended for use by persons not skilled in the relevant art or that sought to address every theoretically possible issue would produce a document of unwieldy size. Such a document would, because of the space needed to explain existing art and address all possible issues, be less usable by persons who were skilled in the art.

From what can be said by a review of the Technical Documentation in the limited time that was available because of the deadline set by the Commission, what I have seen leads me to conclude that Microsoft did the best that could reasonably be expected by normal industry standards to provide the necessary information, as far as it is currently not publicly available, to enable third parties to implement work group servers that are interoperable in the broad sense defined by the Commission – that is, substitutable.

In addition, I considered the Commission’s assertion that Microsoft refused to document anything beyond protocols used “on the wire” in networks, and I found that assertion to be simply not correct. There is absolutely no doubt that the version of the Technical Documentation provided in November 2005, as well as the further release provided on 15 December 2005, went far beyond “on the wire” protocols. Rather, the Technical Documentation also documented internal information of Microsoft servers. The Server Rules and the Explanatory Material provide detailed information about how the provided methods are used and how systems providing them interact with each other.

Moreover, the specification of the relevant protocols together with the additional information provided by Microsoft is sufficient to achieve substitutability, through the normal industry practice described above.

The condemnation of the Technical Documentation by the Trustee, OTR, and Microsoft competitors is not justified under these circumstances. None of them took nearly enough time for a realistic evaluation of the specification of more than 12,000 pages. In particular, the attempt of the Trustee to gather the details for a particular task (introduction of a new user) from the document within a few days is inadequate to make an informed and fair assessment.

A real implementation effort would first derive a more comprehensive design document from the Technical Documentation – a task taking certainly a sizable number of person years. In the end the completeness of the Technical Documentation can only be realistically judged by such a serious effort to develop an interoperable workgroup server. In such a process some of the unavoidable imperfections of a document of the size of the Technical Documentation could be discovered and corrected.

Again, no competitor actually would attempt to rely on Technical Documentation alone. Instead, as matter of industry practice, any serious competitor would rely on and use all available resources to achieve the most efficient the development process. To be able to
develop interoperable work group servers, a competitor would also perform experiments ("interoperability experimentation" and "testing validation") with Microsoft systems in addition to using information as contained in the Technical Documentation provided by Microsoft and information available from Microsoft or public sources. Such measures would always be undertaken in conjunction with documentation review in any serious development effort.
Table of Contents

1  FUNDAMENTALS ON SPECIFICATIONS OF COMPLEX DISTRIBUTED SOFTWARE SYSTEMS ................................................................. 7

1.1  DISTRIBUTED SYSTEMS ........................................................................................................................................................................ 7
1.1.1  CLIENT/SERVER SYSTEMS AND SERVER/SERVER SYSTEMS ......................................................... 7
1.1.2  REMOTE PROCEDURE CALLS AND MESSAGE PASSING .......................................................... 10
1.1.3  PROTOCOLS .................................................................................................................................................................................... 10

1.2  INTEROPERABILITY AND SUBSTITUTABILITY ......................................................................................................................... 11
1.2.1  CLIENT/SERVER INTEROPERABILITY .......................................................................................... 11
1.2.2  SERVER/SERVER SUBSTITUTABILITY .................................................................................. 11
1.2.3  OBSERVABLE BEHAVIOR, INTEROPERABILITY AND SUBSTITUTABILITY .................................. 12
1.2.4  REQUIREMENTS FOR INTEROPERABILITY INFORMATION TO PROVIDE SUBSTITUTABILITY ...... 13
1.2.5  SCOPE OF DOCUMENTATION .............................................................................................. 13

1.3  DOCUMENTATION OF INTEROPERABILITY AND SUBSTITUTABILITY INFORMATION ............................................................. 14
1.3.1  PRINCIPAL APPROACHES TO SYSTEM SPECIFICATION .......................................................... 14
1.3.2  SPECIFICATION DOCUMENTS ............................................................................................. 15
1.3.3  LIMITATIONS OF SPECIFICATIONS ....................................................................................... 15
1.3.4  SUBSTITUTABILITY SPECIFICATION CONTENTS AND METHODS ............................................ 16

1.4  INDUSTRIAL PRACTICE IN SPECIFICATIONS ....................................................................................................................... 18

2  ABOUT THE MICROSOFT SPECIFICATION ............................................................................................................................ 21

2.1  DEMANDS OF THE COMMISSION .............................................................................................................................. 22
2.2  CHARACTERIZING THE SYSTEM TO BE SPECIFIED .......................................................................................................................... 23
2.3  AUDIENCE FOR WHICH THE SPECIFICATION IS WRITTEN ........................................................................................................... 24
2.4  SIZE AND COMPLEXITY OF THE DOCUMENT ............................................................................................................................. 25
2.5  PUBLIC SPECIFICATIONS RELATED TO THE SYSTEM............................................................................................................. 25

3  EVALUATING THE MICROSOFT SPECIFICATION DOCUMENTATION .................................................................................. 27

3.1  THE SCOPE OF DOCUMENTATION PROVIDED ............................................................................................................................ 27
3.2  USABILITY: ARE THE MICROSOFT SPECIFICATIONS USABLE? ........................................................................................................... 27
3.3  IS THE SPECIFICATION STATE OF INDUSTRIAL PRACTICE? ...................................................................................................... 28
3.4  LOOKING AT A SPECIFIC EXAMPLE OF THE SPECIFICATION ............................................................................................. 28

4  COMMENTING ON REMARKS ON THE TECHNICAL DOCUMENTATION ..................................................................................... 31

4.1  COMMENTING ON THE TASK OF EVALUATING THE TECHNICAL DOCUMENTATION .................................................. 31
4.2  THE TRUSTEE’S EVALUATION AND THE SUFFICIENCY EXPERIMENT ......................................................................................... 31
4.3  MISSING DOCUMENTS ........................................................................................................................................................................ 34
4.4  KNOWLEDGE OF THE SYSTEM .................................................................................................................................................. 34
4.5  PURPOSE OF THE SPECIFICATION ................................................................................................................................. 35
4.6  IMPLEMENTATION DETAILS ..................................................................................................................................................... 35

Prof. Dr. Dr. h.c. Manfred Broy 15.02.2006
5 CONCLUSION ......................................................................................................................37

6 REFERENCES.....................................................................................................................38

7 APPENDICES ....................................................................................................................40

7.1 STATEMENT OF QUALIFICATION ..............................................................................40
7.2 DESCRIPTION OF WORKING METHOD ........................................................................41
Fundamentals on Specifications of Complex Distributed Software Systems

Complex distributed IT systems consist of distributed hardware connected by communication lines. Their software systems consist of architectures of software components that interact by exchanging messages over communication channels. The complexity of these systems can be enormous.

In view of the differing opinions on the level of completeness and quality of the Technical Documentation provided by Microsoft, this section gives a short overview of the underlying concepts that are relevant to the assessment of the level of completeness and quality of the Technical Documentation.

1.1 Distributed Systems

A distributed system consists of a family of sub-systems (which we call components) that interact. Their interaction is carried out by exchanging information. The information exchanged is needed to co-ordinate and synchronize the components, to delegate computations, to access and to update their local states.

There are several ways information exchange can be carried out in software systems running on multiple hardware devices, including:

- access to shared memory
- exchanging signals and messages

In a truly distributed hardware system (consisting for instance of a client/server architecture) there is no shared memory between the hardware devices. Therefore information is exchanged by sending messages and signals over the network that connects the hardware on which the software components run. The structure of a distributed system, consisting of the components and their interconnection, is sometimes called the architecture of the system.

1.1.1 Client/Server Systems and Server/Server Systems

A client/server system is a particular architecture that separates client components from server components. Client components send requests for specific services (for example file or print services) to server components, which provide these services. Thus, for a single server and a single client, the architecture can be illustrated as shown in Figure 1. In the real world, there are multiple clients and also more than one server, for instance, separate file and print servers.
In more complex systems, the servers are themselves interconnected, as shown in Figure 2. Here, Server1 may request or share services of Server2, or vice versa. The primary (but not the only) goal of this interconnection is one of data replication. Replication refers to the provision of redundant resources to improve reliability and fault-tolerance: When one server fails, the services demanded by the client may still be provided by the other server. In addition, replication is also used to increase the performance of the services.

While “client/server” is a well-known term, there is no established dedicated term for systems where servers are interconnected. For the purpose of this Report, we simply call them “server/server” systems. Since ultimately, every such system has to provide services to some client, every server/server system is also a client/server system.

In a Microsoft Windows work group server scenario, a computer can act in multiple roles, as shown in Figure 3.
Figure 3 - Computer roles and Active Directory

A computer may be acting as a

1. *client*, requesting services from a server (for instance, file and print services), or retrieving information (for instance, configuration policies) from the Active Directory (AD).

2. *server*, responding to client requests. As a variant, servers may themselves also act as clients, requesting information from a domain controller triggered by a request from a client to himself, with the consequence that some server-server communications can in actuality logically considered client-server communications.

3. *domain controller*. A domain controller is a server that provides authentication and authorization services to clients and servers including other domain controllers. In the Windows architecture the authentication and authorization functionality in a Windows server acting as domain controllers is known as Active Directory. When there are multiple Windows domain controllers within a domain the domain controllers replicate their data so that they can all act from a common data set. Updates to the AD on one domain controller (i.e. a user password change) are forwarded to all other domain controllers to keep the AD data consistent across numerous domain controllers.

This overview illustrates some of the complexity inherent in such systems.
1.1.2 Remote Procedure Calls and Message Passing

A very common way to reduce the complexity of programming distributed systems is to organize the code that represents the software components of a distributed system and their information exchange as remote procedure calls (RPC). The advantage of RPCs is that the interaction between remote (distributed) components resembles the interaction between software components on the same computer device. From a programming perspective, RPC works as follows: A software component that runs on one computer (the client) issues a call (or invokes a method) addressing a software component that runs on a different computer (the server). The call consists of the name (more precisely: a numerical identifier) of the method together with additional information (the “actual parameters”). Then the call is executed on the server. Finally the call returns to the client. When returning, information is carried from the server to the client in terms of the result of the call or via the updated values of the variable parameters.

The mechanism behind an RPC is message passing. For each method call, the client automatically encodes the method identifier and the actual parameters in packets, which are sent over the network wire; the server automatically decodes these packets, and executes the call. Then the return information is automatically encoded by the server, sent over the network and decoded automatically by the client. Because of this automatic encoding and decoding of data, and because of the highly schematic nature of message passing, it is considered easier to program distributed systems with RPCs than directly by raw message passing.

It is common to use RPCs for client/server systems.

1.1.3 Protocols

For the correct interplay of the components in a distributed system, the communication between components is organized by certain rules. Such rules are commonly referred to as protocols. In [MSWSPLA], protocols are defined as follows:

A communications protocol is the set of rules for information exchange to accomplish pre-defined tasks between a client computer and server computer connected via a network. [...] These rules govern the format, semantics, timing, sequencing and error control messages exchanged over a network.

There are many different protocols in use in computer systems. For example, Microsoft distinguishes between RPC and non-RPC protocols:

A non-RPC protocol is a protocol that sends data across the wire in the packets, and includes RPC itself. An RPC protocol is based on a set of methods defined in one or more IDL files. [MSWSPLA].

Prof. Dr. Dr. h.c. Manfred Broy 15.02.2006
The difference between RPC protocols and non-RPC protocols reflects the different communication layers mentioned above. For communication over the network, RPC protocols are based on non-RPC protocols that ensure the transmission of the method calls, method parameters and method responses. A non-RPC protocol sends both the data relevant to the particular operations as well as data about the generic operation itself; whereas an RPC-based protocol carries the data relevant to the particular operation and only an identifier about the generic operation itself. RPC protocols use non-RPC protocols too as a transport from one computer to another. The point here is that protocols are layered.

1.2 Interoperability and Substitutability

The demand of the European Commission is for “interoperability information” [Decision, Article 5] which the Commission contends must include information from which “Microsoft’s competitors can develop products that interoperate with the Windows domain architecture natively supported in the dominant Windows client PC operating system” [Decision, Article 1 and recital 1003]. Microsoft disagrees with the Commission’s interpretation of the Decision, but this Report will accept the Commission’s interpretation of the required “interoperability information” in evaluating the Technical Documentation.

This section clarifies the notion of interoperability and gives an overview over the theoretical principles of interoperability and substitutability. On this basis, the scope and kind of information needed to enable developers to achieve interoperability and “drop-in” substitutability is determined.

1.2.1 Client/Server Interoperability

Looking at the client/server-architecture shown in Figure 1, one goal of the European Commission is for competitors to be able to offer server software so that Windows client PCs can request services in the same manner as from servers running Microsoft server software.

We call this kind of interoperability “client/server interoperability” (and, as noted earlier, this may include some server/server communications). To enable developers to achieve this kind of interoperability, it is necessary to disclose the relevant protocols for the communication that occurs over the network.

1.2.2 Server/Server Substitutability

Looking at the architecture schemas in Figure 2 and Figure 3, we see that in addition to client/server communication there is also communication between servers acting as domain controllers, for example sharing state information to enable the data replication functionality central to the role of domain controllers.

The Commission asserts that Microsoft must disclose the information necessary to enable third parties to replace servers in such a configuration so that a non-Microsoft
server can not only carry out the file, print, and user administration functions of Windows Server 2003, but that it can act also as a domain controller in a domain with Windows domain controllers. In particular, it must be able to participate in the replication of information contained in Active Directory. It does not require the ability to carry out other Windows Server 2003 functions (e.g., firewall), which are outside the scope of the decision.

We call this expanded concept of interoperability “server/server substitutability”, and as defined that server/server substitutability implies client/server interoperability.

To enable engineers to achieve this expanded concept of interoperability, it is necessary to disclose the relevant protocols for the communication that occurs over the network wires between clients and servers, but also between servers acting as domain controllers. The concept of protocols, and thus the kind of information needed, is the same for client/client interoperability and for server/server substitutability.

1.2.3 Observable Behavior, Interoperability and Substitutability

When we try to specify the behaviour of components in a distributed system, we mainly have to describe their interaction. Components in distributed systems are state machines with input and output. Abstracting from their internal states they can be documented as relations between input and output histories.

Issues of observability are of relevance here. The scientific essentials are well-understood by now and summarized in the following:

- Given a distributed system formed by an architecture of components there is a canonical notion of observability for each of the components that consists of the observable histories of communication (these consist of streams of messages that may represent method invocation and return messages) and the causality between the messages in these histories (indicating which message was triggered by which messages). Observable equivalence is captured in the theory of distributed systems by notions of simulation and bisimulation.

- Observability introduces a notion of substitutability as well as concepts of compatibility and interoperability.

- The notion of substitutability is independent of the structure of the internal states of system components. There are systems with completely different internal state spaces that nevertheless are observably equivalent and therefore compatible and thus substitutable.
1.2.4 Requirements for Interoperability Information to Provide Substitutability

The theoretical results above can be rephrased in a more applied manner as follows:

- Protocol specifications that describe the protocol flow “over the wire” capture exactly the information needed to ensure proper interaction, both in the client/server interoperability and in the server/server substitutability sense: When two computer systems have the same behaviour over the network wires, one may be substituted for the other. This is also the approach of the Samba project: “Only wire format is needed for remote interoperability”, [Samba1].

- For server/server substitutability, however, it may be necessary to provide additional information about the replication protocols between the servers, for example, how the replication topology is constructed. Obviously, for the construction of the topology certain information is exchanged between servers, so strictly speaking this is an “on-the-wire” exchange but the information carried over the wire is insufficient to show what the server does with the information. In this case, the global algorithm used for the topology construction needs to be documented.

- While for the proper interoperation of different computers it may be necessary to share state information, the shared information need not be identical (or even similar) to the internal state that is used in a particular server implementation; it is only necessary that the shared information can be calculated by a particular server from its internal state. This means that internal state information need not necessarily be disclosed for substitutability purposes.

There is one area where wire data and server state are in very close correspondence. These are so-called communication handles, which refer to state information stored in the server. Here the structure, and indeed even the nature, of the information itself are irrelevant for the interaction between client and server (or two servers). Such handles are also used in standard programming: File handles, for instance, are nothing but references to more complicated data structures. A handle is obtained by a “file open” command, it is used by “file read” or “file write” commands, it is discarded by a “file close” command – without the user knowing anything about the nature and contents of the data structure referenced by the handle.

In summary, the specification of the relevant protocols (RPC and non-RPC) together with the additional information provided by Microsoft, including information about the topology of server/server interactions, is in fact sufficient to achieve substitutability.

1.2.5 Scope of Documentation

In the proceedings the term “on-the-wire specification” has been used. Our understanding of this term is that it refers to the initial WSPP specification provided by Microsoft, containing documentation for rules of information exchange that govern the
format, semantics, timing, sequencing, and error control of messages exchanged over a network. This information approximately corresponds to the first point of the list in previous section.

This information was declared by the Commission to be insufficient for the production of what has been referred to as “drop-in” substitutable server products. Microsoft then extended the documentation by annexes that, in particular, describe the details of the topology construction and the way this information is distributed among servers. This information corresponds to the second and third items of the list in the previous section.

Thus, all in all, the scope of the Technical Documentation provided does cover the requirements determined above. The matter of appropriateness and completeness of the documentation is discussed in Chapters 2-4.

1.3 Documentation of Interoperability and Substitutability Information

The information for server/server substitutability demanded by the Commission can in principle be provided by software specifications, or by what we will call substitutability specifications. This section clarifies these specification concepts, shows the general limitations of specifications and describes how the necessary information identified in the previous section should be documented.

1.3.1 Principal Approaches to System Specification

There are, in principle, two approaches to specification of interactive distributed systems:

1. Structural Approach: The structure and the elements of an existing system are described in detail. Such a specification describes the software architecture of the system as an interconnected network of components, as well as the interfaces of the components. Each interface consists of a list of methods that can be invoked, including parameters, possible return values, and the effect of a method invocation on the component state. Not all method invocations have an effect on the component state.

2. Behavioural Approach: The interaction of the system on the network is specified. This approach specifies the response a system gives for each request. The responses may depend on the system state. These state dependencies may be specified in an abstract way. There are several ways to specify such dependencies, as described later in Section 1.3.4.

As discussed in the previous section, for substitutability specification purposes, the Behavioural Approach is preferable, as it allows implementations that can be used as replacements for the original system, while giving developers the freedom to choose their own internal software architecture.
1.3.2 Specification Documents

A system specification describes the relevant properties of a software system. In particular, it describes its externally observable behaviour. In contrast, an implementation describes the realization of a system or sub-system by code. Generally, specifications are more abstract than implementations. A specification describes what a system does or what it should do. An implementation describes how it is done. Obviously, an implementation implicitly also describes what a system does, although because the implementation is less abstract, it may be more difficult to understand this description. Implementations that serve as an example for other implementations are called reference implementations.

The task of creating a specification from an existing system is very different from creating a specification for a system to be developed (requirements engineering). In the second case, the system specifications for the new systems are written before the implementation is produced. This is common engineering practice and it is essential if the implementation is to be realized by someone other than the person who envisaged the system. Such specifications often contain design decisions, for example architectural descriptions. These specifications usually, contain additional contextual information to help implementers to better understand their task.

Specifications for existing systems usually have a different focus. Instead of providing design decisions to guide the implementation, here the difficulty is to capture all the details and idiosyncrasies of a system that has evolved and grown over time. Instead of trying to describe additional contextual information to help understand the development task, the focus of these specifications is on providing resource for readers that already have a clear understanding of what they are trying to accomplish in a substitutable system, because the documented system already exists. Thus, the additional contextual information is neither needed nor desired.

Another difference is between software system and substitutability specifications. Software system specifications are examples of the first kind above, detailing design decision and the choice of data structures and algorithms. Substitutability specifications are examples of the second kind.

1.3.3 Limitations of Specifications

The implementation of a complex distributed system is an enormous task, independent of how specification information is provided. No document or set of documents can make this task so easy that it is just a matter of schematic translation of the provided information into source code.

For every major software engineering task, the engineers must have a clear idea – a model – of the system they are going to build. In some, mostly academic examples, this model can be very explicit and very precise, for instance, as a set of mathematical
formulas. In industry, this model will often be mostly a mental model, containing a large number of implicit assumptions and design decisions.

Note, however, that even the best specification will leave some of the information necessary for forming the model implicit, as it is always assumed that implementers have previous knowledge about or access to this information. As an extreme example, in general it is assumed that readers of a software specification know about at least the programming language to be used. More relevant for the proceedings, any implementer of a product that satisfies the interoperability criteria of the commission can be assumed to have relevant knowledge in the art, including knowledge about distributed systems, RPCs as used by Microsoft, the administrative and the client-programming side of the Microsoft server products, as well as the known standards and techniques on which these products are built, such as LDAP and Kerberos.

Thus, for every specification there is additional context information not included in the specification, which must be obtained from other sources. The amount of this context information is dependent on the specification purpose and the intended audience. For substitutability specifications, much more information can be left implicit than for software specifications.

Finally, most specifications used in practice are informal specifications. This means they are written as text in a natural language, possibly augmented by stand-alone diagrams in various (often ad hoc) notations. Unlike formal specifications, written either directly using mathematical or logical formulas, or using notations that have precise mathematical-logical meanings, specifications used in industrial practice are prone to contain omissions, inconsistencies and ambiguities. However, formal specifications can only be produced for a very limited subject matter area far narrower than the subject matter involved here; for larger specifications, one must live and deal with the limitations of informal specifications.

1.3.4 Substitutability Specification Contents and Methods

For a substitutability specification, the protocols by which information is shared must be described. The nature of this description varies with the kind of protocol. Because of the number of different kinds of protocols used in computer science, there is no universal standard (although there are standards for certain specific classes), but only general principles. At least the following information must be provided:

- For non-RPC protocols, a list of packages sent over the network, including the concrete representation of the data fields in each message. Frequently, the representation of the package data is shown graphically; see [MSWSPPCLA, p.32] for an example.
For RPC protocols:

- a list of methods with their identifiers, parameters and return values. This information may be provided as code in a specific interface definition language (IDL).
- The data structures used for parameters and return values shared over the network; this is also relevant for server/server communication, for instance for migration and replication. Typically, this information is specified directly using type-definition constructs of a computer language (usually “C”) or an IDL.
- The encoding principle of the data into packages, including technical issues like byte order or alignment.

The communication addresses that the packages and RPC-methods are directed to; this information is usually provided as a TCP/IP port number or a named pipe.

Both for RPC and non-RPC protocols, the causality between request and response messages must be documented. This point is most difficult. It deals with the dynamics of behaviour. There are several ways to specify causality. Causality may be specified in the protocol descriptions themselves:

- State machines with their state spaces and state transitions.
- Sequence diagrams that show the interaction patterns between communication partners
- Text or pseudo-code for previous history (“Once a file has been opened, it may be accessed…”)
- Implicit by parameter constraints (e.g., a communication handle must obviously be obtained before it can be used)
- Schematically by patterns (e.g., the sequence within an RPC handshake is reused for higher-level communication)

There are a number of quasi-standards to describe such behaviour, and in particular protocols, such as message sequence diagrams or variations of state machines. However, there are no standards that cover the complete range of protocols used in practice.

The causality can also be specified in a separate description, which gives an overview of the global interactions within the system. This is the approach chosen by Microsoft in the extension to the WSPP specification, for instance in the descriptions of the replication protocols.
The statements of the Commission contain several references to [Wirsing], where it is stated that specifications should also describe “dependencies” (besides “behaviour”, for which there are several approaches, as outlined above). The term “dependencies” is somewhat ambiguous here, and can be taken to mean several things:

- Causal dependencies between requests and responses; these are behavioural aspects, as discussed above, and should certainly be part of a substitutability specification document.

- Dependencies on other (third party) software products. Such software products should, where applicable and relevant, also be referenced in the specification document. Example for such software might be proprietary, non-disclosable encryption systems.

- Dependencies on other documents. For instance, if a protocol is described elsewhere in a generally accessible document, it is sufficient to reference that description.

A perennial issue in protocol specification is the specification of error return values. In principle, it is desirable to precisely know the causality between erroneous situations (or, simpler, erroneous parameters) and the error codes that arise in that situation. In industrial practice, it is not unusual that only a list of relevant error codes is provided, and it must be inferred from the names or descriptions of the error codes which one is relevant in which situation.

Finally, mentioned in Section 1.2.4, to achieve server/server substitutability, it may also be necessary to specify how the topology for server/server communication is constructed. The WSPP specification contains a description of the spanning-tree algorithm used. Algorithms may be specified in a number of mathematical-logical approaches, but in industrial practice pseudo-code (a non-standard, but somewhat structured way of writing highly simplified commands in a natural language like English) or a reference implementation in a real programming language is used more often. Sometimes, various approaches are combined, for instance as a fragment of code where some parts are replaced by English language, and others by mathematical formulas.

### 1.4 Industrial Practice in Specifications

It is one of the Holy Grails of computer science and software engineering to provide precise formal complete and tractable specification of distributed systems and their protocols. Much has been achieved in science along those lines. Today we have a well worked out theory for that and a number of experiments that show that such specifications are in fact feasible for small system instances. However, a formal comprehensive complete specification of systems of a size of millions lines of code is not state-of-the-industry. Here typical for pragmatic issues a mixture of informal specifications are needed that rather form a patchwork of information providing protocol
information about the meaning of certain messages explaining the structuring of the messages and their purpose when exchange between the components of systems.

Due to the inherent difficulty to give a specification and comprehensive documentation of the size of the WSPP in practice a pragmatic approach is used everywhere. It consists as described above, and as reflected in the Microsoft documentation, of a mixture of information about the type and structure of the messages exchanged, their sequences of interaction, their effects described in terms of abstract state models and reactions including error messages. Much of this information is given implicit in the way described in Section 1.3.4.

Typically the system behaviour is not described in all detail in industrial interoperability or substitutability specifications. The audience for those documents consists of experts with extensive background knowledge. Information that can be assumed to be known to the target readers is not always contained. This holds also for information that can be inferred from the provided information, or which can be obtained from public sources, or which is obvious.

Nevertheless, considering the size of the WSPP documentation, it is to be expected that open issues remain. It is impossible to produce a document of that size with no errors or ambiguities. In industry, it is therefore common that the information provided is clarified in an iterative process with the developers of the documentation. Here Microsoft’s offer of technical assistance (helpdesk) is state of industrial practice and the appropriate way to close unavoidable gaps, errors and misconceptions.

Finally, it is also common that some information is obtained by tests and experiments during the implementation process. OTR, one of the parties evaluating the WSPP specification, determined that:

“in order to develop an interoperable product, a competitor would have to perform a considerable number of experiments using functioning combinations of Windows work group servers and PCs.

[OTR-D1]

This statement has been erroneously interpreted as meaning that the Technical Documentation is only useful when accompanied by reverse engineering of the protocol.

As the Samba project has shown, reverse engineering of protocols at a very low abstraction level is a major problem for the implementation of interoperable solutions [Samba2]. Much of the reverse engineering effort of the Samba project deals with the discovery of the RPC methods and the parameter formats used on the wire. Because of the high cost of reverse engineering, it has not been considered as a viable solution to produce alternative server products.

However, the Technical Documentation provided by Microsoft contains all the low-level information that the Samba project is sought to obtain through reverse engineering. In
particular it contains the relationship between network messages and RPC formats including the RPC method identifiers and the high-level data types used in the RPC parameter and return value descriptions. Thus, the Technical Documentation is the “missing link” needed to make reimplementation feasible for competitors without this act of reverse engineering.

This does not exclude that for implementation of a non-Windows server that can replace a Windows server, a certain degree of interoperability experimentation may be needed. Having a “functioning combination of […] servers and PCs” [OTR-D1] in order to test and validate an implementation is a necessity for any real-world network software development project; using it to experiment with the documented API in order to get a deeper understanding of implementation issues is just plain common sense and definitely established industrial practice.

The situation here is not different from that of programming against any other library, for instance, against the Sun Java Swing library for graphical user interfaces. Typically, a developer learning about the library (that is, building a mental model of the library information) will build a system incrementally, checking at every step that method calls have the desired effects and return values.

It should also be noted that while computer science has well-established notions of both specifications and the demands on content and structure put on specifications, industrial practice in this regards has no well-established practices to ensure that even specification documents of limited size are consistent, complete and non-ambiguous. Some issues with published specifications are well known. For example:

- The implementations of HTML (mentioned as a good example for specifications in [Wirsing]) and CSS vastly differ in practice. This leads, as mentioned by Wirsing, to the development of the Mozilla “quirks” mode. (Incidentally, this is also an example of the use of interoperability experiments.)

- Compiler implementations, in spite of large-scale standardization efforts (also mentioned as a good example in [Wirsing]), are also different in many subtle and not-so-subtle points; this can be readily seen from the fact that there are several dedicated compiler validation suites on the market, which are, for instance, used by any serious embedded software maker.

- The on-going problems with connecting Bluetooth phones to automotive systems (demonstrated by the “compatibility lists” of any major car vendor) show that interoperability is still a problem in spite of open standards.
2 About the Microsoft Specification

Documentation describing complex systems like Microsoft work group servers is built up in different layers with every layer having a different focus and addressing users with a different background. The different layers are built on top of each other and provide all the information needed to understand the system only in sum.

![Figure 4 - Levels of documentation](image)

Besides this structuring of documentation, Microsoft also provides documentation and specification for different kinds of users. There are two important groups:

- Administrators and IT-architects
- Developers and software-architects

Administrators and IT-architects need to know how products work, how they interact with each other (Overview Layer), how they are maintained and repaired in case of an error (Howtos). Therefore, they need an understanding of the structure of the systems and their dependencies. Most of Overview and Howto documentation can be found on Microsoft TechNet (technet.microsoft.com) and is available to the public. The TechNet site also contains whitepapers describing the concepts the applications are built on (Concepts/Best practices). Concepts/Best practices layer describes the concepts needed to have an understanding of the inner structure of a system. Best practices are a set of guidelines for improving system stability, performance, and security. Other information sources include Microsoft Windows NT/2000/2003 Resource Kits, which cover the Overview to the Concepts layer with a focus on the ’How to’ layer. In addition, Microsoft provides newsgroups that provide a forum to ask questions and receive answers. These forums are viewed by users and Microsoft employees and the answers to the questions are provided by both other users and often by the Microsoft employees that monitor the newsgroup. The aforementioned documents cover all the conceptual documentation layers except API and the implementation details layer.

Prof. Dr. Dr. h.c. Manfred Broy 15.02.2006
Developers create software applications that use the services provided by the operating system. In addition to the information needed by administrators, developers need to know and understand the interfaces they can use to access the services provided by the operating system. The documentation that provides this information is called the Application Programming Interface (API). Microsoft provides API information on the Microsoft Developer Network (MSDN) website (msdn.microsoft.com). In addition to the API information, developers need to understand the concepts that lie behind the APIs. To gain this understanding developers use the whitepapers provided by Microsoft on its TechNet website. If a software developer wants to build an operating system or software component that provides the same services that another operating system or software component provides then that developer needs to understand not only the APIs and the concepts that lie behind the APIs, but also, must understand the implementation details of the operating system or software component that he is trying to emulate (Implementation Details).

The following sections will focus on is the Technical Documentation provided by Microsoft through the “Workgroup Server Protocol Program” (WSPP). The Technical Documentation provides the Implementation Details documentation layer. Again, as stated in the opening of this section, a developer needs to be familiar with all the documentation layers, including the Implementation Details layer.

2.1 Demands of the Commission

Article 5(a) of the Decision requires Microsoft to

\[
\ldots \text{make the Interoperability Information available to any undertaking having an interest in developing and distributing work group server operating system products} \ldots
\]

The Commission asserts that the objective of this article is elaborated in recital 1003 of the Decision:

\[
\ldots \text{to ensure that Microsoft’s competitors can develop products that interoperate with the Windows domain architecture natively supported on the dominant Windows client PC operating system and hence viably compete with Microsoft’s work group server operating system. Microsoft should thus allow the use of the disclosed specifications for implementation in work group server operating system products. The order to supply is therefore not limited to disclosing specifications but also encompasses authorising the implementation of such specifications in work group server operating system products [Decision, recital 1003]}
\]

The term “drop-in” has been used for that expanded concept of interoperability. In practice a “drop-in” replacement would require two steps; the first step is adding a non-Windows domain controller into a Windows domain and transferring all necessary state

Prof. Dr. Dr. h.c. Manfred Broy 15.02.2006
information to it using the documented protocols and implementation. The second step would be to remove the target Windows domain controller, transferring all its clients to the non-Windows domain controller in a way that would basically go unnoticed by the clients\textsuperscript{1}.

Sections 1.2 and 1.3 elaborated on what we would expect of substitutability specification documents for this purpose, and stated that the scope of the Technical Documentation is sufficient for this purpose.

2.2 Characterizing the System to be specified

Following at the demands of the Commission, Microsoft would have to specify multiple systems and protocols. Most of them are part of Windows Server Systems (Windows NT Server, Windows 2000 Server and Windows Server 2003\textsuperscript{2})

- **Microsoft Active Directory (AD)**
  The Active Directory is a distributed directory service, containing information about computers, groupings of computers (implemented by domains), their configuration (group policies), users and their rights and resources. AD is extensible and can be used as a foundation for other applications\textsuperscript{3}. Because AD is based on the X.500 ITU standard [ITU500], it can interoperate with different technologies:

  o Lightweight directory access protocol (LDAP), a standard protocol for information retrieval from directories. Microsoft makes additional information available for LDAP [MSLDAP], information can also be easily found on the internet [CPLDAP] and [CSVDE]

  o Active Directory Service Interface (ADSI), a Microsoft standard to access the AD. ADSI is very well documented on the MSDN website and additional information can very easily be found on the Internet.

  AD is built up from one or multiple domain controllers running the domains and additional AD functions like global catalogue server and infrastructure master.

- **Domain controller functions and domains**

\textsuperscript{1} The way this is done is documented in the “File Replication Service further Explanatory Material and Server Rules”.

\textsuperscript{2} And their different variants (Standard, Advanced Server, Enterprise Server, Datacenter Server).

\textsuperscript{3} Microsoft Exchange is a product from Microsoft that incorporates AD as directory service and stores information in AD through schema extensions.
Microsoft Windows domains are a concept already found in the first releases of OS/2, jointly developed by Microsoft and IBM. Windows NT 3.1, 3.51 and 4.0 enhanced these functionalities by providing single-sign on, new encrypting algorithms and trusts between domains. Domain controllers are the master servers in a domain, providing the domain services.

Starting with Windows 2000, domain controllers are also the foundation for the Active Directory. The service is provided in a multi-master mode, meaning that every server can read and write to the data in the directory. It is kept in a consistent state by using replication between the servers.

- Member Server
  A member server is a server integrated into a domain and providing services to clients and servers of this domain. This server does not provide Active Directory services or domain services. In the work group server scenario this is a file- or print server.

Besides of these functions, a Windows Server System can provide additional functions like web server, firewall, or media streaming services. These are not included in the Technical Documentation and are also not requested by the Decision.

The Technical Documentation used in conjunction with public standards and the available published art is designed to disclose the information necessary for a third party to create a non-Microsoft server to interoperate and interact with a Microsoft operating system at a client/server, server/server, and domain controller level.

2.3 Audience for which the Specification is written

The Technical Documentation is written for software engineers attempting to implement a work group server that is able to act as domain controller. It therefore assumes a deep knowledge in network technology (network communication, communication infrastructures) and the concepts behind it (directory services, user and system management, security). To protect against overly large unwieldy size, the Technical Documentation assumes a basic level of developer skill, including knowledge of Microsoft systems and the programming concepts and methods behind them. The concepts of what a work group server is and the structure of an Active Directory are NOT explained in the Technical Documentation. These concepts can be readily found in external locations such as TechNet. Explaining everything a programmer would need to implement a work group server operating system in one document is unreasonable and may not be possible.

---

4 Starting from Windows 2000-releases as Active Directory integrated DCs
2.4 Size and Complexity of the document

The protocols and systems making up Windows workgroup systems are among the most complex systems in the industry providing much more functionality out of the box than other directory services. Windows work group servers have evolved through the years becoming more am ore complex while still maintaining compatibility with early versions of Windows server and work group clients.

It is important to understand that the Technical Documentation was written after the implementation of the Windows work group servers (partly long after the implementation). It is not a specification that has been used as a basis to produce code. Therefore much of the documentation process required close inspection of the source code and to develop the Technical Documentation.

Because of the complexity of the system, the broad functionality it provides, and the backward compatibility, the Technical Documentation is unavoidably large and complicated. This is another reason not to overly burden the technical documentation with information that is readily available elsewhere.

2.5 Public specifications related to the System

As stated previously the Technical Documentation is one layer of multi-layered larger specification landscape. The other layers include documentation such as the information available on the extensive Microsoft Developer Network, TechNet, and publicly documented standards. An example of a publicly documented standard is Kerberos V5. The specification must be read in conjunction with published standards like RFC 1560. The document also states the important fact that the specification is subject to change and will be altered and adopted on a regular basis.

Microsoft provides to the public a vast amount of information that correspond to the different document layers identified in Figure 4:

- Overview Layer
  - Microsoft is providing whitepapers on the website, showing the different functions of a system in an overview.

- Howtos Layer
  - Howtos explain how to implement and use the systems. These documents contain the administrator’s documentation (Windows Resource Kit), but also Howtos on special topics not needed for every installation.

---

5 Besides security settings done by an administrator.

6 Every month.
• Concepts / Best practices Layer
Microsoft is providing solution accelerators and best practice guides helping to implement a Windows work group server. There are also operation guides, showing how to maintain a system and deployment guides to help an administrator to roll-out a system. These deployment guides also show the dependencies between components of a system.

• API Layer
Microsoft is documenting most of its API on MSDN. The website can also be ordered on DVD and currently has about 3 GB in size.

• Implementation Details Layer
Microsoft is not providing documentation on all the implementation details. Microsoft has books, called Inside X (Windows 95, NT, 2000,…) that explain the concepts, how these systems work internally. But implementation is up to the implementer.

Open standards like Kerberos, LDAP and RPC are also documented by their owners. These documents have also to be taken in account.
3 Evaluating the Microsoft Specification Documentation

This chapter is based on the Alternate Format Version of the Technical Documentation as provided to the Commission on 15 December 2005. I see no evidence that the Trustee or any of the other critics of the Technical Documentation ever even read the 15 December version. I have also reviewed the materials that were added in November 2005 to the Technical Documentation.

3.1 The Scope of Documentation Provided

The Commission maintains that Microsoft refused to document anything beyond protocols used “on the wire” in networks. This claim is inaccurate. The version of the Technical Documentation provided in November 2005 provided much more than “on the wire” protocols and supplied substantial information relevant to achieving substitutability. Likewise, the 15 December version also goes well beyond “on the wire” protocols.

3.2 Usability: Are the Microsoft specifications usable?

The specification is provided as an Adobe PDF file protected by Windows Rights Management (WRM). There are two components, which have to be installed on a computer to be able to access the documents:

- Windows Rights Management Client
- Add-in for Adobe Reader for Windows Rights Management

The document itself is a PDF file that can be used with normal Acrobat Reader functions. Extraction of content and printing is not enabled as a security measure to protect Microsoft’s intellectual property and as stated in [Com2512] mechanisms to implement these are accepted by the Commission. On balance the current security measures and the technology used for enforcement seem to strike a reasonable balance between usability and security. Acrobat PDF is readable and a widely accepted form for documentation presentation. Problems stated by the Trustee in [TReport2] are features or behaviours of Acrobat Reader and not related to the substance of the Technical Documentation\(^7\). The version of the documentation I reviewed contained working hyperlinks and I am therefore unable to find a basis for the Trustee’s complaint that there were no hyperlinks [TReport2].

OTR stated their comments to be based on a specification based on another file format and that the actual PDF form is considered to be useable by them [OTR1]. The search

\(^7\) Beside copy-paste and print-functionality, but these are functions not made available on purpose, see above.
facility of Acrobat provided information on the whole document within one minute\(^8\), in our opinion a good response time for 12,000 pages.

The use of the widely accepted PDF format for the Technical Documentation should in our view put to rest any question regarding usability.

3.3 Is the Specification State of Industrial Practice?

As already stated, the Technical Documentation is for a very complex system. The documentation therefore covers a lot of material.

Ultimately, the completeness of the documentation can only be judged by a serious and realistic attempt to develop a work group server. To a large extent information is not set forth that either can be concluded by common sense or found elsewhere or assembly state-of-the-art and therefore can be included by experts implicitly.

The use of a common file format like PDF is state of practice. Microsoft often uses word documents for whitepapers, not having done so here can be seen as an accommodation to the Commission not forcing the use of a Microsoft product.

3.4 Looking at a specific example of the specification

To analyze the specification, we focused on one limited example. Our limited focus was a product of the limited time we were afforded to do this review. The example we choose was the Windows Update Service protocol. The Windows update protocol is used by work group server to either provide updates to other computer or to obtain updates from other computers or services such as Windows update.com.

As background information, as illustrated above, Windows Clients can retrieve software updates from

\[^8\] 3 year old notebook, newer systems should provide even better performance
- a website (update.microsoft.com) that the user accesses. The provision of updates is done by interaction with the website. No local server is needed.

- a client application feature integrated within Windows XP. This application feature is running in the background and retrieves information from the website. (update.microsoft.com)

In the enterprise, IT administrators often prefer to control updates by placing a server between the client and Microsoft’s update service. Note that the update service can be from any provider and is not limited to Microsoft. In common parlance this intermediate server is called a Windows Update Services server (“WSUS”). The system Microsoft provides for Windows Update Services is documented on at [WSUS]. A WSUS server can be used for caching and control over what updates are provided and installed on the clients in the network. This system can also be extended to scan for updates for other products than the ones provided by Microsoft.
A detailed description on the API and programming applications based on the service is provided through the MSDN Website at [WSUS3].


Using the knowledge we already had from running a WSUS infrastructure at our site, and using the documentation for developers and administrators, we started analysing the WSUS system and the protocol. We identified two channels that our WSUS server would have to communicate in:

- First, our server must be able to communicate with a software update site and request updates for the clients in our network.
- Second, our server must be a web server showing a website to the user and provide software updates to the clients in our network.

Analyzing the source code of the websites and the documentation provides us the knowledge that there is an ActiveX control embedded in the Website handling communication with the update.microsoft.com website. The MSDN documentation provides a comprehensive description on how to develop an application that is using this control on the client. It also shows the sequences by which information is transferred. The ActiveX control acts as a proxy, hiding the real network traffic that is generated from the developer. To implement our own server, we need to know more about how traffic transferred over the wire.

Using the Technical Documentation, we achieve an understanding, how messages are built up. The Technical Documentation references to functions used and provides hyperlinks to the MSDN website. Example code in the Technical Documentation demonstrates how the messages that are transferred over the network.

We learn that SOAP⁹ messages are transferred between the client and the update website. This is something we already would have expected. We learn, how the server gets the information concerning the system configuration of the client it.

Using the ActiveX control on a client and our implementation we could now start to implement a server. By using the control on the client we can send messages to our implementation to check whether it is working correctly and the messages transferred are correctly recognized by the control running on the client.

---

⁹ SOAP is a standard used to transfer messages between web services.
4 Commenting on Remarks on the Technical Documentation

In this section we comment on the Trustee’s so-called sufficiency test and the remarks by the competitors.

4.1 Commenting on the Task of Evaluating the Technical Documentation

Providing documentation of WSPP is a major effort. It is obvious that Technical Documentation contains both some systems designs produced a number of years ago as well as some quite recent design information. It is also clear that the disclosed information was available within Microsoft sometimes rather implicitly in documents often deeply buried in information that is proprietary and need not be disclosed under the Commission's Decision. Therefore putting together the Technical Documentation is for Microsoft a major effort of reverse engineering, re-documentation and deciding on which information has to be included and which not.

Judging from the size and complexity of the Technical Documentation, producing the Technical Documentation took according to our estimation far more than fifteen person years. The documentation also had to be written within a relatively short time period. Therefore already just the coordination of the team and the information to be captured by the team cause major difficulties. One has to keep in mind that reading through the documentation of about 12,000 pages requires 200 hours –five weeks of working time – if one studies each page for one minute only.

What applies to the production of the document carries over also to the evaluation of the document. Evaluating a specification of this size in only three days is impossible. All the conclusions reached by that approach are quite speculative and certainly not firmly grounded.

4.2 The Trustee’s Evaluation and the Sufficiency Experiment

The Trustee claims that there is insufficient introductory and explanatory material. As already shown, if the documentation is understood being a specification document for use by experienced developers, in accordance with industry practice, such information is neither expected nor necessary.

Furthermore the Trustee complains that a considerable amount of reverse engineering would still be needed even for a programmer with appropriate skills. It is true that an undertaking would need to perform test and experimentation on his product to make sure that it works in practice; this is not, however, "reverse engineering" in the sense required by, for example, the SAMBA project (as discussed in Section 3.4). Rather, the Technical Documentation, like any specification, must be combined with common engineering practices such as experiments ("interoperability experimentation" and "testing validation") with Microsoft systems, which provide enough information to fulfil the
requirement to enable implementation of an interoperable system. This, however, is different from reverse engineering, as explained in Section 3.4.

In addition, taking into account the size of the Technical Documentation and the little time the Trustee had to review the Documentation, his limited findings cannot be taken as reliable conclusions.

In the sufficiency experiment [Sufficiency] the Trustee tried to trace all the information needed for the implementation of two functions.

- Add a new user
- Propagate a password change

These functions seem to be quite simple and isolated. Having a closer look at the AD, the way it is acting as a system and taking in account all the subsystems that are involved in this task reveals that this is not true. These functions

- interact with the client accepting the change request
- have to store the information in AD
- have to replicate the data to other domain controllers

Trying to do an experiment on a system and a specification of the size of a server by trying to implement a particular function, which in addition requires substantial functionality, on which it is based, is not trivial. To carry out such an attempt without solid background within a few days is again quite impossible and frankly speaking naive. Such an experiment needs a deep understanding of the system to be considered, and it certainly needs a longer period of preparation and research into the documents in order to gain that understanding.

We will evaluate the experiment and show how better background knowledge would have provided a more positive outcome. We do not consider our experiment as finished, due to a lack of time, but will show that the efforts of the Trustee could have been successful.

The Trustee is obviously not fully aware of the way the AD operates. AD data is not centrally located, but instead resides on all domain controllers. These domain controllers replicate the AD data between them to keep the directory in sync. When domain controllers interact with clients to provide a service, such as changing a password, the user does not know which server he is interacting with. The domain controllers that happens to be interacting with the client implements the changes to their local copy of the AD, e.g. changes the users password. This information is subsequently propagated to other domain controller servers by using replication mechanisms. Making the updated password information available to all domain controllers. This also explains, why the Trustee did not find the function he searched for in the Active Directory specification, but
in the file- and print-server specification, because it consists of a client/server interoperable function. The replication mechanisms take place after the password change has been done locally on the domain controller that the client is interacting with.

Additional documents that can be found on the Internet explain how a password change works and how it is replicated to other servers in the AD. As stated in [PWChange]:

[...] When a domain controller carries out a password change, the change is forwarded to the primary domain controller (PDC), which performs the PDC emulator operations master role (also known as Flexible Single Master Operation or FSMO). This type of change isn't an urgent replication but is instead a separate communication that notifies the PDC emulator master outside of regular replication connections.

When a user enters an incorrect password to initiate an authentication request, before failing the authentication, the domain controller that received the authentication request asks the PDC emulator master to verify the password and confirm whether a new password is in use. If the password has changed, the PDC emulator communicates the password to the domain controller outside of normal replication cycles. This communication for verifying incorrect passwords is for any domain controller in the domain, not just those within a local site. [...] 

Searching for “FSMO” in the specification results in 616 references. Querying for PDC emulator shows that this is a pre-AD function provided for Windows NT Domains. Therefore we know that this can be done by using the functions a Windows NT domain controller provides. Due to the fact that Samba can act as a Windows NT domain control to Windows Clients, this function can be considered to be common knowledge or is retrievable from the Samba sources. 

Summing up our analysis we find that:

- The Trustee was, at least at the time of the experiment, not sufficiently familiar with the system

- The Trustee made errors in his experiment, like not using the right parts of the specification (see the use of samr than NETLOGON API above), although he mentions these APIs in his report.

10 We did not further investigate, whether this information is available in the CIFS specification, what we would expect.
• The Trustee in his evaluation did not take other relevant document sources into account.

4.3 Missing documents

One of the complaints often mentioned by the competitors is missing documentation on specific topics. For example,

• “[…] Active Directory-based Domain Objects can be managed using LDAP, […] documentation describing how to do this is incomplete” [IBM]. This claim is not appropriate. There are lots of documents from Microsoft and from other sources available describing how to interact with the AD using LDAP. Some of them are intended to be used by Administrators [CSVD] to script every day operations or do bulk operations, others are intended to be used by developers [CPLDAP]. Microsoft also comments on how to configure LDAP to access it from other Systems than Microsoft Windows [MSLDAP]

As we also argued above, the claims have to be evaluated and additional documents have to be taken into account that are not part of the specification.

4.4 Knowledge of the system

Complaints of the competitors also show that the individual evaluators were not always fully familiar with the topic and the system they had to deal with:

• [IBM] states that AD Replication by using SMTP was not clearly documented. “IBM’s evaluators saw discussions of extensions to the SMTP protocol…but it was not clear how this applies to replication”. The comments of Novell state that to them the use and specification was [Novell] clear. The replication mechanism and in which scenarios it is used is also documented in [ADREPL]

• Further IBM mentions the lack of the description of AES. Because AES is not used in this scenario, a documentation of it is not necessary.

• [IBM] states that the secure DNS update protocol is not specified. „This feature is particularly important because Microsoft’s implementation is assumed to be proprietary.“ Even if the protocol is not specified in the document, there are documentations available that do show how to use non-Microsoft DNS Server with Active Directory. The inclusion of this protocol is therefore not necessary. [BIND09-1] and [BIND09-2]

• OTR states that it is not documented where public documents may be found. This lack of knowledge about which standards are used and where their specification may be found implies a certain lack of knowledge of the whole system.
4.5 Purpose of the specification

Also the character and purpose of the specification and what to expect from it is not always clear between the competitors:

- [IBM] complains that there are neither test cases nor a test suite for compliance checks. This is claimed to be state of the art when doing a specification. As a matter of fact, compliance check test cases are, in industrial practice, not part of a specification. To produce test cases is the responsibility of the party implementing a system. Perhaps there is some confusion with the notion of acceptance tests, when implementation is delegated to a contractor; the situation here is completely different. Test cases are part of an implementation, one was never requested by the Commission.

- [ORACLE] states that the Windows Update Service specification is not sufficient. The Windows Update Service uses an ActiveX Control to do updates on the client; according to Oracle, that is not provided and also runtime information is not available. This is only partly true. On the client side, there is an application running and requesting updates from a Microsoft Website. In enterprise scenarios, this location can be altered to use a local server. The protocol the client is talking to that server is part of the documentation. An independent Windows Update server could therefore be implemented (see Section 3.4). The ActiveX control is used on a website Microsoft provides. It is an offer Microsoft is providing additionally and not needed to provide updates to a client via the Windows update mechanism.

- [ORACLE] states that there is no documentation of the .NET framework. Although this is not necessary, because besides some of the administrator’s applications, no component in a work group server scenario is using the framework, Microsoft has an implementation provided in source code [ROTOR]. There is also an open source implementation of the .NET framework [MONO].

4.6 Implementation details

Some of the statements of the competitors refer to implementation details that are not needed to do an implementation.

- [Novell] states that it is important to know whether a delete file operation results in an immediate deletion of the file on the disk or whether the deletion will take place at a later time. This information is needed, according to Novell, to know, whether an application requesting space on the disc sees the space as freed or not. Looking at the actual and former implementations of Microsoft Windows we can see that this is implementation detail need not be fully specified, because even Windows 9x and Windows NT/XP handle this issue differently. The space information is updated, when the user presses refresh in Win9x. Windows NT/XP
passes this information active to the client by the means of RPC as do new implementations of Samba.

- In [TReport2] and [Sufficiency] the Trustee complains about missing information about the internal structure behind communication handles (in this special case, the handle “SAMPR_HANDLE”). As discussed in Section 1.2.4, these structures are references to the internal server state and neither need nor should be disclosed in a substitutability specification document.

The Trustee notes that knowledge about such a data structure is needed for an implementation. However, this is part of the intellectual effort of a reimplementation and is not a necessary part of a substitutability specification.

The competitors as well as the Trustee complain in several places about what they say is the lack of information about proper sequencing within the protocols. State transition diagrams are mentioned as one way to present this sequencing. While such diagrams can be useful they are by no means required. This can be evidenced by the lack of such diagrams in several standards based Internet protocols specified in Requests for Comments (“RFCs”). These RFCs are distributed as plain-text ASCII documents and do not contain diagrams. The salient point here, however, is not that of graphical vs. textual representations, rather is that the sequencing messages can be specified in various ways and sometimes must be taken in context. For example, RPCs are not that much different from the procedure calls in standard programming methodology. In both cases it is common practice to derive the proper sequencing from the naming of the procedures (for instance, any programmer would expect an “open” or “create” to come before a “close”), and in particular from the dataflow which is implicit in the method parameters and return values. For instance, to use a function that expects a file handle (such as “write”), this handle must be obtained – generally, such a handle is the return value of an “open” function.

In summary, in standard programming it is quite common to derive sequencing information from the description of the programming interface. The same approach will be taken by an experienced developer for the RPC interfaces in the Technical Documentation.

If there are found to be protocols in the Technical Documentation that have a non-trivial sequencing which cannot be derived using the method described above, these parts can be improved in cooperation with developers working with the specifications through the free Microsoft Technical Assistance that has already been offered.
5 Conclusion

Looking at the size, the heterogeneity, the long standing history and all the side conditions, the Technical Documentation provided by Microsoft is certainly what is considered normal industry practice and what was in fact the best Microsoft reasonably could have been expected to do facing the complexity of the task. One has to understand that such a specification has to be iteratively improved over time and through use by developers.

It is obvious that such a specification could only be proved adequate (or on the other hand shown to be insufficient) if someone is prepared to do a substantial implementation based on the documentation and if this experiment is done by the appropriate staff with the required background and competencies. I would expect that in such an experiment issues would arise that would have to be addressed by Microsoft experts. But this is exactly what is provided by Microsoft’s offer of 500 hours support from Microsoft engineers. Furthermore such technical discussions between a potential implementer using the Technical Documentation and the Microsoft engineers corresponds to normal industrial practice and will lead to improvements in the Technical Documentation step by step.

One criticism is that protocols are documented at different levels of details. First, for a Technical Documentation under what is considered industrial practice this is not unusual. Second, some protocols are easier to understand, or more obvious than others. Some protocols are even documented in more detail elsewhere. Third, some documentation pre-existed, in some form within Microsoft while other documentation had to be constructed or reconstructed from existing documents or from existing software, thus creating a differences in the documentation.

In fact, there is no single standard for protocol documentation, because of the varied types of protocols, which ask for different descriptions. There are some standards around but those standards are also very much influenced from the domain where the protocols were developed.

There were some complaints about whether the Technical Documentation is up-to-date. The concern can be assuaged because Microsoft has stated that it will update the documentation in a revised form every month based on feedback and further work.

From what can be said by our evaluation of the Technical Documentation, Microsoft did what is considered normal industry practice and what was in fact the best it could to provide the information (as far as not publicly available) to enable third parties to implement interoperable work group servers.
6 References

[ADREPL] Active Directory Replication, 

[BIND09-1] Planning for Windows 2000 DNS, Part 1, Microsoft Website, 
http://www.microsoft.com/technet/community/columns/profwin/pw1200.mspx#EDAA

[BIND09-2] Bind 9 DNS Server mit Windows 2000 Active Directory ohne MS DNS, 
Network4You GmbH, Daniel Heinze, 

[Com2512] Commission Decision of […] imposing a periodic penalty payment pursuant to Article 24(1) of Regulation No 1/2003 on Microsoft Corporation

[CPLDAP] Querying Active Directory using .NET classes and LDAP queries, 
http://www.codeproject.com/dotnet/activedirquery.asp

[CSVDE] CSVDE Example, 
http://computerperformance.co.uk/Logon/Logon_CSVDE_Bulk.htm


[MONO] http://www.mono-project.com/Main_Page

[MSCPPFAQ] Microsoft Communication Protocol Program Frequently asked questions, 
http://members.microsoft.com/consent/Info/faq.aspx

[MSLDAP] How to configure Active Directory to allow anonymous queries, 
http://support.microsoft.com/default.aspx?scid=kb;en-us;Q320528

[MSWSPPPLA] MICROSOFT WORK GROUP SERVER PROTOCOL PROGRAM 
LICENSE AGREEMENT FOR DEVELOPMENT AND PRODUCT DISTRIBUTION


Prof. Dr. Dr. h.c. Manfred Broy 15.02.2006


[PWChange] Flash Tip: How password changes are communicated between Active Directory sites,
http://www.microsoft.com/technet/abouttn/subscriptions/flash/tips/tips_060805.mspx

[ROTOR] Shared Source Common Language Runtime,

[Samba1] Andrew Tridgell. Testing MSRPC. Presentation slides,

[Samba2] Andrew Tridgell. Network Analysis Techniques. Presentation slides,

[Sufficiency] WSPP Documentation Sufficiency Test. Attached to Letter from Cecilio Madero Villarejo to Ian Forrester, 17.01.2006


[WSUS1] Microsoft Windows Update Services,
http://www.microsoft.com/windowsserversystem/updateservices

[WSUS2] Windows Update Server Services, TechNet Docu,
http://www.microsoft.com/technet/itsolutions/midsizebusiness/prodtechnol/WSUS/default.mspx

[WSUS3] WSUS Object Modell, MSDN Docu,
7 Appendices

7.1 Statement of qualification

Prof. Dr. Dr. h.c. Manfred Broy
Fakultät für Informatik
Technische Universität München
80290 München

Manfred Hans Bertold Broy studied Mathematics and Informatics at the Technische Universität München. He graduated in 1976 with the Diplom in Mathematics and Informatics. 1976 - 1980 he worked as a research assistant at the Institut für Informatik. In February 1980 Manfred Broy received his Ph. D. (Doktor der Naturwissenschaften) with the thesis "Transformation parallel ablaufender Programme" at the Faculty of Mathematics at the Technische Universität München. After that he continued to work as a teaching assistant at the Institut für Informatik of the Technische Universität München. A main research topic was the formal foundation of distributed, reactive real time systems and their specification and verification.

In 1982 he completed his Habilitation Thesis: "A Theory for Nondeterminism, Parallelism, Communication and Concurrency" at the Faculty for Mathematics and Informatics at the Technische Universität München.

In April 1983 he became a full professor for informatics and the founding dean at the Faculty of Mathematics and Informatics at the University of Passau. His research focused on the systematic development of distributed interactive systems.

In October 1989 he became a full professor for computer science at the Faculty of Informatics the Technische Universität München (former chair of Professor F.L. Bauer).

His research interests cover software and systems engineering comprising both theoretical and practical aspects. This includes system models, specification and refinement of system components, specification techniques, development methods and verification. He is leading a research group working in a number of industrial projects that apply mathematically based techniques and to combine practical approaches to software engineering with mathematical rigor. There the main topics are ad hoc networks, software architectures, componentware, software development processes and graphical description techniques. In his group the CASE tool AutoFocus was developed.

Professor Broy is the organizer of the Markttoberdorf Summer Schools in foundations of programming. He published a four volume introductory course to computing science (in German), which was also translated into Russian. He served as the main editor of Acta Informatica and is editor of Information and Software Technology, IEEE Transaction on Software Engineering, Science of Computer Programming, Journal of Universal

Professor Broy is a member of the European Academy of Sciences and a Member of the Deutsche Akademie der Naturforscher „Leopoldina“. In 1994 he received the Leibniz Award by the Deutsche Forschungsgemeinschaft and in 1996 the Bundesverdienstkreuz. From 2001-2003 Broy was a member of the IEEE Computer Society Board of Governors.

Professor Broy has been working for many years as a scientific advisor for various institutions, including major national and international companies, among them Siemens Business Services, Siemens, Daimler Chrysler, EADS, IABG, ESG, CA, BMW, Allianz, and Bosch. In numerous international software engineering projects he served as an advisor and referee.

For this report, Professor Broy collaborated among others with two members of this scientific staff:

- **Dipl.-Inf. Norbert Diernhofer** is leader of the Microsoft Competence Center at the Technische Universität München. He is an expert in Microsoft client and server technologies —including Active Directory— and software architecture in general. Norbert Diernhofer directed and coached numerous student projects and gained industrial experience in more than 10 years of working as an IT-architect and consultant for companies like Audi, Bechtle, Bayernoil, sd&m and BMW.

- **Dipl.-Inform. Jan Philipps** has in various research projects and publications contributed to the state of art in interface specification of distributed and embedded systems. In several scientific and commercial projects in the domain of design and test of distributed systems —ranging from embedded automotive systems to smart cards applications—, he gained a deep insight in the state of industrial practice in specification.

### 7.2 Description of working method

Based on the specification we focused on three different directions:

- We analyzed the different terms used by the Commission and in the specification and tried to link them with each other (plug-in versus on-the-wire)

- We compared the Microsoft specification with industrial practice and state of the art

- We focused on other material available on work group servers from Microsoft and other sources.

Going into these three directions we did the following steps:
• Comparison of the Technical Documentation with industrial practice from different areas (Automotive, Telecommunication, Business Application Systems, Smartcard applications)

• Comparison with publicly available documents

• Examination of the Trustee’s experiment and the competitor’s reports in view of industrial practice and under consideration of publicly available documents

• Analysis of the Windows Server Update Protocol in deep

• Exemplary valuation of the WSPP Technical Documentation (version of 15.12.2005)

Many of the different resources we used are mentioned in Chapter 6.