

Design and Future Architecture of an
Information System for
Virtual Office Management
NQA Hyperwave

Diplomarbeit der Telematik

MARTIN MELCHER

Technische Universität Graz
Institut für Informationsverarbeitung und
Computergestützte neue
Medien (IICM)

Fertigstellung:	Graz, März 1999
Begutachter:	o.Univ.-Prof. Dr.phil. Dr.h.c. Hermann Maurer
Betreuer:	o.Univ.-Prof. Dr.phil. Dr.h.c. Hermann Maurer DI.Dr. Richard Messnarz, ISCN Ltd., Ireland

To Nasti and Kasi

Abstract

To stay competitive on the global market it is necessary to set up win-win based agreements in cost sharing projects in which partners from different countries share the risk and the effort and jointly exploit ideas, products, and services. The NQA system (Network Based Quality Assurance) was designed to meet these requirements. A network based project and quality management systems makes the co-operation of home-working or on-the-road-working team members possible. The co-operation model of NQA is not restricted to the members of a team only. Thinking large, the co-operation of whole companies would be possible over a single NQA system or in the future over a NQA network building a *Virtual Enterprise*. Easy configurability and adaptability make NQA an attractive web based application developed on top of the Hyperwave Information Server system.

Aspects of software quality and a look on new work arrangements are building the basement for an in-depth discussion of the NQA system in this thesis. A tutorial-like introduction to the Hyperwave system and a discussion of all Hyperwave-programming-techniques including real-world programming examples show the technical part of the system.

This thesis covers the commercial release 1.0 of NQA Hyperwave. This release was field-tested by *Austrian Web Application Center (AWAC)*, Austria; *Bosch*, Germany; *Daimler-Benz*, Germany; *Forschungszentrum Informatik (FZI)*, Germany; *Hyperwave R&D*, Austria and *SZTAKI*, Hungary.

Um am globalen Markt überlebensfähig zu bleiben, ist es notwendig, mit internationalen Partnern finanziell wie auch in der Produktentwicklung zu kooperieren. Das NQA System (Network Based Quality Assurance) wurde so entworfen, um diesen Anforderungen gerecht zu werden. Ein netzwerkbasierendes Projekt- und Qualitätsmanagementsystem macht die Zusammenarbeit von Heim- oder Außendienstmitarbeitern in einem Team möglich. Das NQA Kooperationsmodell ist aber nicht auf die Mitglieder eines Teams beschränkt. Global gedacht, wäre die Zusammenarbeit ganzer Unternehmen über ein einziges NQA System denkbar. In Zukunft wäre der Aufbau virtueller Unternehmen über ein NQA Netzwerk vorstellbar. Die Flexibilität und Anpassbarkeit des NQA Systems zeigen die hohe Attraktivität einer als Hyperwave Server Anwendung entwickelten Lösung.

Aspekte der Software Qualität und ein Ausblick auf neue Arbeitsformen wie Teleworking oder Telecommuting bilden die Grundlage für eine tiefgreifende Diskussion des NQA Systems in dieser Diplomarbeit. Eine leicht verständliche Einführung

in das Hyperwave System und eine Behandlung aller Hyperwave Programmiermöglichkeiten, mit Beispielen aus der Praxis, beleuchten den technischen Teil des Systems.

Diese Diplomarbeit behandelt die kommerzielle Release 1.0 von NQA Hyperwave. Am Field Test für diese Version haben teilgenommen: *Austrian Web Application Center (AWAC)*, Österreich; *Bosch*, Deutschland; *Daimler-Benz*, Deutschland; *Forschungszentrum Informatik (FZI)*, Deutschland; *Hyperwave R&D*, Österreich; *SZTAKI*, Ungarn.

Contents

1	Introduction	7
1.1	Organization of this Thesis	7
1.2	The History of Hyperwave	8
2	Aspects of Software Quality Management	10
2.1	What is Quality	10
2.2	Defining Software Quality	11
2.3	Quality Costs	12
3	The Virtual Office	14
3.1	Decentralized Working - Telecommuting	14
3.2	Forms of Telecommuting	16
3.2.1	Home Offices	16
3.2.2	Telework Centers	17
3.2.3	Mobile Offices	18
3.2.4	Hoteling	19
3.3	Telecommuting and Technology	19
3.4	Telecommuting and Security	20
4	NQA	22
4.1	The Scope of NQA	22
4.2	NQA's General Concepts	23
4.2.1	Development by Configuration	24

4.2.2	NQA Product Strategy	24
4.3	NQA in the Software Development	26
4.3.1	The Planning Activity Flow	26
4.3.2	Planning Documents Overview	26
4.3.3	A Role Play for Planning	28
5	NQA Features	31
5.1	Creating and Deleting Projects	31
5.2	Handling the Phases of a Project	32
5.3	An NQA Project Phase	36
5.4	NQA Document Management	38
5.5	Email Transactions	43
5.6	Linking Reports and Documents	46
5.7	The Interproject Link Generator	46
6	Hyperwave	51
6.1	Hyperwave Features and Benefits	51
6.1.1	Problems in the “First Generation” World Wide Web	51
6.1.2	Hyperwave Structural Elements	52
6.1.3	Hyperwave Metadata	52
6.1.4	Hyperwave Programmability	53
6.1.5	Other Features	53
6.2	Hyperwave Programming Paradigms	54
6.2.1	PLACE	54
6.2.1.1	A PLACE Model	54
6.2.1.2	Basic Place Syntax	56
6.2.1.3	Applied PLACE Programming	59
6.2.2	Hyperwave and CGI	62
6.2.2.1	What is the CGI	62

6.2.2.2	Perl and CGI.pm	64
6.2.2.3	CGI Scripts and the Hyperwave Server	66
6.2.3	The Hyperwave API	67
6.2.3.1	The HWJS Command	68
6.2.3.2	JavaScript in the WaveMaster Templates	69
6.3	A Hyperwave Programming Example	71
6.3.1	When should the HTML form be displayed?	73
6.3.2	Getting User Input and Calling the CGI Script	77
6.3.3	Doing the Work: The CGI Script	82
6.3.4	Working with Server Side JavaScript	86
7	Conclusion	88
7.1	The Future of NQA Hyperwave	88
A	A Complete List of Placeholders	90
B	Hyperwave Tools Used in the NQA System	103
B.1	General Information	103
B.1.1	Default Values and Environment Variables	103
B.1.2	Return Codes	104
B.2	hwinsdoc	105
B.3	hwinscoll	108
B.4	hwdownload	110
B.5	hwmodify	112
B.6	hwci	114
B.7	hwco	116
B.8	hwdochistory	118

C A NQAHW Roadmap for Programmers	119
C.1 Scope of the Roadmap for Programmers	119
C.2 Replacing Document Templates in NQAHW	120
C.2.1 How Document Templates are handled in NQAHW	120
C.2.2 The Replacement Procedure	120
C.3 Replacing Form Templates in NQAHW	121
C.3.1 How Form Templates are handled in NQAHW	121
C.3.2 The Replacement Procedure	122

List of Figures

2.1	Quality Costs	13
4.1	Development by Configuration	25
4.2	NQA Product Strategy	25
4.3	The Planning Activity Flow	27
4.4	Planning Documents Overview	27
4.5	A Role Play for Planning	28
5.1	The Project Creation Dialogue for Not Authorized Users	32
5.2	The Project Creation Dialogue for Authorized Users	33
5.3	Listing of all Projects in the NQA System	33
5.4	Dialogue for Deleting NQA Projects	34
5.5	NQA Project Sheet	34
5.6	NQA Projects Workflow Log	35
5.7	Additional Documents Collection	36
5.8	An NQA Project Phase	37
5.9	Adding a Document to a Project	37
5.10	Adding a Report to a Project	38
5.11	Document/Report Header for Read Access	39
5.12	Document/Report Header for Read/Write Access, Checked In	39
5.13	Document/Report Header for Read/Write Access, Checked Out	41
5.14	Document/Report Header with Changed Document State	42
5.15	Report Header for a New Created Report	42

5.16	Report Footer for a New Created Report	43
5.17	The Form for Editing the Recipients List	43
5.18	Document Header with Entries in the Recipients List	44
5.19	Extended Document Header	44
5.20	Email Submission Feedback	45
5.21	Generation of the Recipients from the Access Rights	46
5.22	Creating a Review Report linked to a User Requirements Document	47
5.23	Header of a Linked Review Report	47
5.24	Footer of a Revised User Requirements Document	47
5.25	Concept of the Interproject Link Generator	48
5.26	Interproject Link Generator Bar in the Header of a Document . .	48
5.27	The Interproject Link Generator Dialogue	49
5.28	The Interproject Link Generator Header Bar with Link Destina- tions Listed	50
5.29	The Interproject Link Gnerator Footer Bar with Link Sources Listed	50
6.1	A PLACE Model	55
6.2	PLACE in Combination with JavaScript	61
6.3	Form Interaction with CGI	63
6.4	The Project Creation Dialogue	71
6.5	NQA Project Collection Structure	72
6.6	Example Overview	72
7.1	NQA Update/Maintenance Strategy	89
C.1	How Document Templates are handled in NQAHW	120
C.2	How Form Templates are handled in NQAHW	121

Chapter 1

Introduction

1.1 Organization of this Thesis

NQA stands for *Network based Quality Assurance*. The first chapter deals with the term “*Quality*”. What is quality and how can especially software quality be defined? This chapter should demonstrate the need for quality assurance systems and was intended to be a theoretical base for the chapters 4 and 5 where the practical aspects of the NQA system are described.

One topic of this thesis is the use of information systems, NQA can be seen as such, in the *virtual office*. Chapter 3 explains the principles behind decentralized working and some manifestations of virtual offices.

Chapters 4 and 5 finally give a complete description of the NQA system. While chapter 4 discusses the more or less platform independent principles behind NQA, chapter 5 explains in-depth the use of NQA Hyperwave version 1.0. A complete role play in chapter 4 and vivid screenshots in chapter 5 make this part a valuable reference for NQA Hyperwave.

The last chapter of this thesis covers the Hyperwave Information Server and especially the way to program it. The first part gives a tutorial-like introduction in using PLACE, CGI and Server Side Javascript. In the second part a real-world example, taken out of the NQA system, shows how the theory presented in the first part is used to solve complicated problems.

Finally the appendix gives a complete list of PLACEholders and a detailed description of the HWTools used in the NQA development. This might be a valuable resource for Hyperwave developers from that point of view, that this information was not available in printed form until now. An excerpt from the *NQA Programmers Guide* shows how the principle of *Development by Configuration* is applied in the NQA system.

1.2 The History of Hyperwave

The Institute for Information Processing and Computer Supported New Media (IICM) of the Graz University of Technology, where Hyper-G was conceived, has had long experience in design, implementation and operation of large online hypertext services for fairly large user communities. In particular, the institute (which was then called IIG, Institute for Information Processing Graz) played a significant role in the introduction of Videotex in Austria and parts of Europe starting in 1982 [25].

Around 1989, a small team consisting of Hermann Maurer, Ivan Tomek (now at Acadia University in Canada) and Fritz Huber (now with Anderson Consulting) gathered some requirements for ‘the optimal large-scale hypermedia system’, code-named Hyper-G. Of course, the designs of other similar systems were also taken into account, most notably Intermedia, NoteCards and Xanadu¹ [25][35].

After the Austrian Ministry of Science agreed to fund a prototype development phase of Hyper-G in January 1990, Frank Kappe began to consolidate the requirements and came up with an architectural design of Hyper-G in his Ph.D. thesis in 1991. A very small group of two (!) programmers implemented the first generation of the server (Gerald Pani) and the VT100 client (Frank Kappe) now known as HGTV [25].

In January 1992, the system was put into real use as the University Information System (TUGinfo) of the Graz University of Technology, one of the first such systems world-wide (it has been up and running for seven years now). The success of this system, together with the increasing popularity of the Internet and simple Internet-based information systems, made it evident that Hyper-G could really be useful for a wide range of applications. This made it possible to acquire funding from various sources for a second phase, in which the prototype was to be transformed into a real product, including graphical user interfaces for MS-Windows and UNIX/XWindows [25].

Phase 2 began in summer of 1992. From then on the project has been carried out by IICM in cooperation with the Institute for Hypermedia Systems (HMS) of JOANNEUM RESEARCH, an Austrian non-profit research institution. An important milestone was the adoption of Hyper-G by the European Space Agency (ESA) for its ‘Guide and Directory’ system in late 1992. An early version of Amadeus was released in June 1993, a year later version 0.84 of Harmony appeared [25].

In 1997 the name of Hyper-G was changed into Hyperwave and the server became commercial. A company was founded with research and development in

¹<http://www.xanadu.com.au>

Graz, Austria and marketing in Munich, Germany² [10]. The Hyper-G native clients, Amadeus and Harmony, were replaced by the direct server access over a Web browser. Version 2.6 of the Hyperwave Information Server was amongst the winners of the *European IT Prize* and got a *Byte's Magazine's Best of Show Award* on the CeBIT'97.

In 1998 version 4.0 and 4.1 were released and Hyperwave became a full-programmable platform with it's own API, currently implemented for JavaScript.

²<http://www.hyperwave.com>

Chapter 2

Aspects of Software Quality Management

2.1 What is Quality

In the daily life, quality is often associated with the way a product is designed, with materials that are used and the products look and appearance. We expect quality products to have a higher price and a better market position than comparable products.

From the business point of view, other quality aspects are of higher importance. “Fitness for purpose” or “does it do what it is supposed to do?” are phrases which make a better approach to the meaning of software quality.

“Quality is the totality of features and characteristics of a product or service that bears on its ability to satisfy stated or implied needs.” That’s how the International Organization for Standardization (ISO) defines quality. Michael G. Jenner brings the role of the customer into the ISO quality definition:

“In other words, quality means satisfying the customer. There is no room in this definition for high quality or low quality. You either satisfy the customer completely or you do not. [...] It does not matter what we think, it matters only what our customers think. This is the challenge we must all face. And what adds to the excitement is that customer’s perception are constantly changing. They are demanding more and better products and services, and it is up to us to make sure our organization is fast enough and flexible enough to respond to the challenge presented by customers whose sophistication is increasing daily [20].”

From this point of view quality is defined through two main categories [4]:

Product Requirements

- The product-specific functional and operational requirements, intended to meet the purchaser's identified needs.

Standards

- Standards which define the 'goodness' and effectiveness of the particular product (e.g, operational reliability, user response times);
- General standards applicable to the type of product or to its operational requirements (e.g. conformance to industry test requirements, compliance to industry operating procedures).

The degree of a product's quality is given in this way as extend to which it:

- complies with its requirements
- is fit for its intended purpose.

2.2 Defining Software Quality

Quality management principles don't tell us directly how to specify a software product's quality or how to show, that all the requirements are met. Attempts which have been made over the last few years how to define software quality and how to asses it, looked at the operational user of the product. Another way to define software quality is to look at the way the software was produced.

That's the point where the NQA system (discussed in chapters 4 and 5) comes into the play. NQA controls the way how software (or what product ever) is produced and guarantees the accordance to a specific standard.

If quality is derived from the operational use, the assessment of product characteristics is involved. These characteristics are a result of viewing at the software product while

- using it as it is (product operation)
- changing it for maintenance or modification (product revision)
- and re-using it for other requirements (product transition)

In the following table, the gained characteristics are listed [4]:

Using the software as it is	
Usability	Can the software be used for its intended purpose?
Correctness	Does the software do what is required?
Reliability	Does the software continue to do what is required?
Efficiency	Does the software do what is required with minimum resources, and within time and cost requirements?
Computability	Does the software work (as relevant) with other required systems or software or to other external requirements?
Integrity	Is access to the operating software controlled and does this access only allow usage that is relevant?
Maintaining and modifying the system	
Understandability	Is the function and the structure of the software clear?
Modifiability	Can the software be changed?
Testability	Can the software be tested?
Re-using the system	
Portability	Can the software function in a different environment?
Re-usability	Can the software be used for different requirements?

2.3 Quality Costs

A good starting point for a quality improvement program is the potential cost saving. It is often stated that between 20% and 30% of a company's turnover is wasted by poor quality and its consequences, such as rework, inefficiency and delayed deliveries. The payback from quality improvement comes from progressively reducing and eliminating this 'waste'. It is still not unusual to hear of IT managers who claim to spend as much on maintenance and support as on their development [4].

As a company handles quality more effectively, the quality costs which it recognizes will move through four categories (see figure 2.1):

Failure Costs: Failure results in costs of lost opportunity, since much time is spent on corrective work and this prevents the personnel from being productive.

Appraisal Costs: When the extent of failure costs is realized, it will lead to more time and effort being spent on appraisal, for example by performing more inspections, testing and other forms of quality control. The result is the early detection of failures which are cheaper to correct.

Prevention Costs: Eliminating errors before they happen decreases appraisal costs and increases productivity.

Improvement Costs: One of the aims of an improvement program can include altering the balance of quality costs - instead of incurring uncontrolled and recurring failure and opportunity costs, the money and resource is invested in appraisal and prevention.

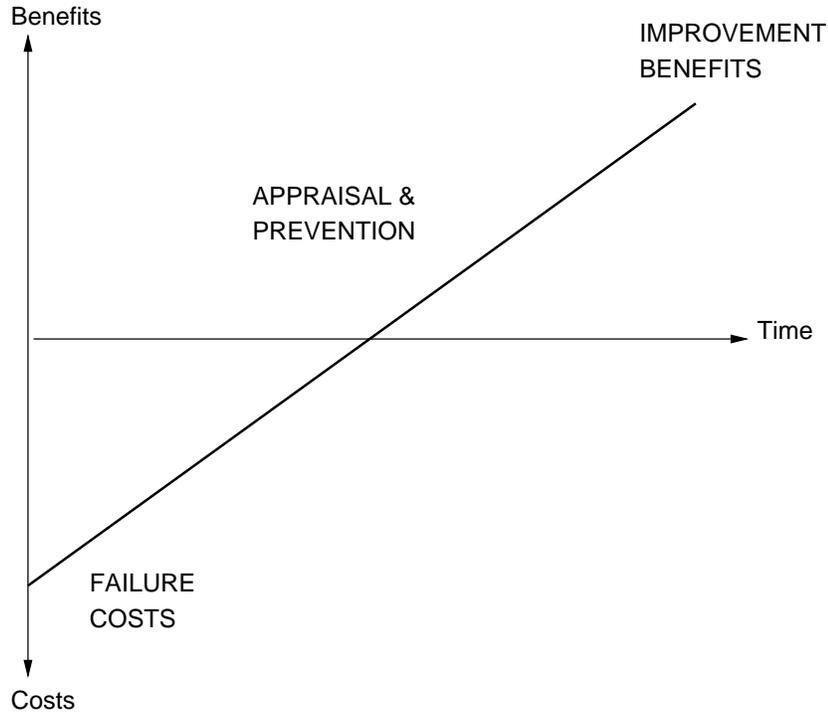


Figure 2.1: Quality Costs

Chapter 3

The Virtual Office

3.1 Decentralized Working - Telecommuting

For people in Europe it is not always clear, what the difference between teleworking and telecommuting is. A reason for this might be, that there is no translation for telecommuting in the most European languages [29]. There would be a translation into German, but the word “*Telependeln*” is not used and doesn’t reflect its meaning. The Oxford English-Reader’s Dictionary [9] subscribes to commute as: “travel daily , esp. by train or car, to and from one’s work in a city, etc.”

Jack M. Nilles tries to give a definition [29]:

Teleworking: ANY form of substitution of information technologies (such as telecommunication and computers) for work related travel.

Telecommuting: moving the work to the workers instead of moving the workers to work; periodic work out of the central office, one or more days per week either at home or in a telework center.

Based on this definition, when Europeans are talking about teleworking, they are meaning telecommuting in reality. In the United States, where the ideas of decentralized working were born, telecommuting is a subset of teleworking. But whatever you call it, the operating principles are the same.

AT&T, as one of the leading companies as far as teleworking and telecommuting is concerned, gives it’s own set of definitions¹ [1]:

¹AT&T has it’s own telework guide online:
<http://www.att.com/telework>

Teleworking can mean a regular agreement where you work from home on specific days each week or from a satellite office closer to home. Instead of commuting to a central location, your workplace may find you teleworking from home, traveling across the country or temporarily located in your customer's office. Telework is an umbrella term for a wide range of alternative office arrangements including telecommuting, virtual/mobile office, hoteling, satellite office and telework center.

Telecommuting: Working from home one or more days a week during normal business hours.

Since the industrial revolution started early in the 19th century the trend in industrializing countries has been to centralize workplaces. The reason is simple: industries, factories and assembly plants, needed centralization. To run efficiently, they needed to be located near source of materials, supplies, and production workers: cities and their transportation hubs [29].

Since the middle of this century the percent of people working in the manufacturing industry is declining. More than 50% [29] of the Americans are working in the information industry nowadays. Their job is the creation, collection transformation and/or dissemination of information. One interesting point of the product information is, that it isn't bound to a location. The tools for working with information are movable and so it is always possible to move your work and not yourself.

There are other aspects which make telecommuting a very interesting topic:

- The commute to (and from) work is more than normally dangerous and/or takes more than forty minutes a day round trip. During that commute you are not doing something either pleasurable or useful [29].
- There are too many meetings that are insufficiently focused. The time wasted in such meetings could be spent in productive working.
- The working environment produces high levels of stress and productivity loss. Jack M. Nilles talks about the *tree crucial I's*: interruptions, interruptions and interruptions [29].
- You are running out of office space. To many people in a office are inflecting the moral and productivity.
- The commuter's cars on the road are increasing the air pollution.

June Langhoff points to the importance of the last item:
Passage of the Clean Air Act by the U.S. Congress in 1992 has spurred companies

to find alternatives to commuting for their employees. So naturally, one of the largest beneficiaries of telecommuting is the environment. With fewer commuters on the road, traffic congestion is reduced. If we all worked at home only once a week, we'd cut traffic by 20%. Energy is conserved, demand on transportation infrastructure is reduced, and air pollution is cut significantly [23].

The location-independence, the enhancements in telecommunication and the enormous power of modern computing makes telecommuting an attractive solution for more and more managers. Fiber optics, digital phone networks and cellular phones are acronyms of instant electronic togetherness. In the computer industry one generation of information power plants follows the other in incredible short periods. And often you have more computing power in your sleeping room than in your office.

3.2 Forms of Telecommuting

All kind of offices described here fall under the category *virtual office*. So the virtual office is the place where telework or telecommute takes place. The most common types of virtual offices are home offices, telecommuting centers, mobile offices and hoteling. The drawbacks and benefits of the different arrangements are discussed here.

3.2.1 Home Offices

One big advantage of having the office at home is that it makes it easier for people to run their lives. The time normally spent with commuting can be used for the family, for hobbies or even for extra work.

Working at home also means fewer meetings and interruptions. Not surprisingly, this often translate into productivity gains. The study by AT&T Home Business Resources found that 80% of work-at-home entrepreneurs say they are more productive at home than they were in a traditional office [2].

Most of the home office's drawbacks are a result of the geographic combination of work and private life. Working in a home office forces you to manage your family and other personal relationships in a special way. Alice Bredin [2] mentions some telecommuting aspects which are very interesting from the psychological point of view:

- How to sell the idea of a home office to the family? How will this office inflect the daily life?

- How to explain children what's going on? Preschool children might not understand that one doesn't have the time to be with them if one is physically present?
- How to make clear for friends and relatives that you are really working and not just sitting around at home?
-

Even if one manages all the troubles listed above, the downside of working at home is that it can be lonely and stressful. Many teleworkers are looking for the interaction they had in the office. So a common solution is to work only two to three days a week at home. The rest of the working week will be spent in the traditional office where meetings can take place and conversation with colleagues is possible.

Home can be an effective base for telecommuting, allowing significant cost reduction for both employer and employee, allowing employees access to jobs that otherwise might not be available, allowing employers access to people who otherwise would not be available, providing significant productivity gains and a host of indirect benefits to society (energy conservation, pollution reduction, etc.). The air pollution reduction aspect of this is a major incentive for many organizations to be involved in telecommuting, generally in response to increasingly strict environmental regulations. For most employees, home-based telecommuting works only as a part-time option [29].

3.2.2 Telework Centers

There are two types of telework centers, the satellite telework center and the local telework center. A satellite telework center is an office building, or part of a building, that is wholly owned (or leased) by an organization, to which its employees regularly report for work. A local telework center is almost the same as a satellite center. The difference here is that the building may house employees from several different organizations. Otherwise, each organization's set of offices are arranged as if it were a satellite center [29].

A telework or telecommuting center offers the possibilities of a regular office on one hand and the advantages of working in a home office on the other. You have colleagues to interact with and computer networks and technical equipment not available at home. Telework centers are situated in regions where people live, so it's often not necessary to commute or even to use the car.

Many multi-employee centers have had difficulties finding tenants for their centers, and some have closed because of a lack of interest. Some of the centers

that are making it at this point are in a “demonstration phase” and are being subsidized by federal, state, or local governments.

Also in Austria first tries are made to establish telework centers. The project “Televillage Bruck an der Leitung” shows the direction of future developments as far as teleworking in Austria is concerned:

In October 1994, the regional government of Lower Austria invited a consortium of the Center for Social Innovation (CSI/Laboratory G.I.V.E.), the Research Institute for Social Economy at the Austrian Academy of Science, and the IBM Consulting Group to conduct a feasibility study regarding the implementation of a telework center as a pilot-project within the framework of regional telematics initiatives.

The project’s first step was to find out about the conditions and possibilities to place a telework center in a housing project. A carrier organization should be installed and supplied with necessary information and support by experts, future users, and officials in order to design, build and run the telework center.

The telework center in Bruck an der Leitha, Lower Austria, shall serve as a model case for further telematic initiatives throughout the country. At the present state of planning it will consist of one large (separable) multi-purpose conference room, approximately 20 full-time office-workspaces, one large room for temporary users, as well as a small kitchen and communication areas. in the basement a kindergarten, a multi-supply shop, the administration and technical support will complete the building’s multifunctionality² [33].

Information about teleworking and telecommuting in Austria and Germany can be found in [21], [22], [24] and [32].

3.2.3 Mobile Offices

The term *mobile office* describes a car, or sometimes even a briefcase, used by people who spent a lot of time on the road. These “road warriors” have all the technology and other tools they need in their mobile office to complete their work without returning to the central office. They complete their work in their mobile office, at a client site, or in a home office, and go to the corporate office only for meetings, to pick up mail, or for support services that are not available elsewhere [2].

The mobile office is the main paradigm for virtual office arrangements. There is no office building, no desk no watercoller or coffee kitchen. A consultant or a

²The final report of the feasibility study for the project “Bruck an der Leitung” is currently being completed. Up to date information can be found under:

<http://obelix.soe.oeaw.ac.at/telework>

traveling salesman equipped with a portable computer, high tech peripherals and a mobile phone is often the first impression of a virtual office.

Enabling mobile workers to spend more time with their clients through technology is appealing to employers. The same technology that allows mobile workers to keep in touch enables them to improve productivity and spend more time with clients.

It must not be ignored, that working in a mobile office arises a bulk of social problems. Consultants might spend more than 40% of their working time abroad or away from home. It can't be neglected that social relations suffer under these conditions. According to Dr. Daniel Kuna, psychotherapist and leadership coach for business owners and upper management in Toledo, Ohio, it is normal for people who are frequently on the road to feel isolated. He stresses the importance of maintaining contact with people with whom you can be yourself and have close, personal interaction. Time spent during the day with clients, with whom there is an agenda aside from interaction, cannot take the place of contact with family and friends. In [2] some tips from Dr. Kuna are listed how to deal with such a situation.

3.2.4 Hoteling

Although technically not an office, hoteling is a catch-all term for work arrangements where office space is shared on a drop-in basis by employees. Employees either reserve space in advance or drop in to use a cubicle on an as-needed basis. Other names for similar but not identical concepts include *unassigned offices*; *free-address*; *nonterritorial offices*; *Just in Time* and *hoteldesking*.

3.3 Telecommuting and Technology

A substantial amount of telecommuting can be accomplished effectively with only a telephone, paper and pencil as the relevant technology. Yet, application of more sophisticated technology generally makes life easier, increases the amount of telecommuting one can do, and makes telecommuting available to more people [29].

Jack M. Nilles lists 6 rules for using technology as telecommuter or as entrepreneur who employs telecommuters[29]:

1. If a certain form of information technology is available today, but costs twice as much as you think you can afford to pay, wait a couple of years; it will be down to your price threshold. If it currently costs ten times as much as you think you can afford, wait about seven years.

2. Always buy the best technology available to accomplish a certain job. Even if it stretches your budget slightly. It is at least a partial guarantee that the technology will still be usable in three years. Don't count on more than three to five-years useful lifetime for computers or telecommunication interface equipment (such as modems), for other than very routine information tasks.
3. The absence of a particular technology, beyond the fundamentals, is rarely a reason (or excuse) not to telecommute. Almost everyone can telecommute at least part of time without any form of "advanced" technology. However, improvements over the fundamentals may enable both significant qualitative and quantitative improvements in telecommuting.
4. Given equal human and economic resources, the person who has the technology best suited for the job wins. If you are able to do the work faster, with higher quality, at lower costs, or with less strain than your competitor, then you have a competitive advantage. The key question is: Is the cost of additional technology less than the value of the increased competitive advantage? It is, then the expenditure could be warranted.
5. Telecommuting generally decreases the start-up costs of adoption of a new technology, computer-based technology in particular.
6. The technology needed for full-scale successful telecommuting is roughly the same as that required in the principal office - plus some more telecommunication.
7. Telecommunication networks are the freeway of telecommuting.
8. There is no substitute for uniform company technology standards.

3.4 Telecommuting and Security

Computers and networks are an integral part of the technology used and needed for teleworking or telecommuting. Today no discussion about computer networks is complete without speaking about security.

Actual cryptography methods and enhancements in network security make it easy to protect sensitive company information as long as it stays inside of the company. Stored on office computers with no access allowed from outside all the data can be considered safe from misuse. It is not possible to totally exclude outside access to the data while operating in a teleworking environment.

There are two main strategies to protect sensitive data. The first is to block unauthorized access, the second is data-encryption. Several layers of access protection can be build into e.g modem access such as all or some combinations of [29]:

- having the telecommunication server not directly connected to the main-frame or LAN
- using “smart cards” that display a password that changes every 30 seconds or so, in synchronism with a password identifier in the computer called (the telecommuter gets out a smart card, dials up the company machine, and enters the password appearing on the card at the moment)
- using a call back system - assuming all the password routines are completed correctly (many notebook computers have them built into the communication software, which is why they are desirable theft objects), the central computer dials the telecommuter’s home or other prearranged phone number
- requiring a positive identification of the caller, such as a retinal scan or a fingerprint or hand shape detector

A security method very often stressed and discussed in the last time is the encryption of data. Especially public key algorithms are very useful in company-telecommuter data transaction.

The concept of public key cryptography was invented by Whitfield Diffie and Martin Hellman, and independently by Ralph Merkle. Their contribution to cryptography was the notion that keys could come in pairs - an encryption key and a decryption key - and that it could be infeasible to generate one key from the other [30]. In this way the company could use the encryption key and only the telecommuter for whom the data was encrypted could decrypt the content with his key. In combination with cryptographic signature methods, the telecommuter can also be sure that the sender of the data was really his employing company. A very profound discussion of computer cryptography can be found in [30].

Chapter 4

NQA as Quality Management Tool in the Virtual Office

4.1 The Scope of NQA

Network based Quality Assurance

To stay competitive on the global market it is necessary to set up win-win based agreements in cost sharing projects in which partners from different countries share the risk and the effort and jointly exploit ideas, products, and services. Through effective and distributed collaborations organizations can cut down their risk significantly (e.g. sharing the development cost with other partners) and can reach a much larger market (e.g. selling the product then in more regions of Europe through VARs - Value Added Re-sellers).

However, the key problem is that distributed collaboration needs effective coordination of the work of the different partners. And old conservative means such as direct supervision, local meetings, large local and not distributed teams, do not work any more. The decomposition into smaller competence teams with clear cooperation interfaces supported by new and effective communication systems is needed. This includes a virtual office on the net with project archives and document management, configuration management, guide-lines and computer support for project documentation, network and computer supported information flows, and appropriate security mechanisms assuring privacy of the materials exchanged and produced.

NQA is developed in co-operation with Hyperwave a leading information management system, and places the required functionality of such a virtual office on top of it. The system has been field tested in leading German and Scandinavian industry in the software co-operation and out-sourcing sector. Most of the large companies (e.g. Siemens re-organizes again and centralizes more, Daimler Benz and other manufacturers require a much more closer co-operation between the teams of suppliers and their internal teams) start to question the simple out-sourcing concept. They think about a system that will create a virtual office through which teams from the supplier and the manufacturer work together as if they are one team in the same office, thus solving the missing-control problem of out-sourcing [27].

NQA was designed with the *Virtual Office* in mind. A network based project and quality management systems makes the co-operation of home-working or on-the-road-working team members possible. The co-operation model of NQA is not restricted to the members of a team only. Thinking large, the co-operation of whole companies would be possible over a single NQA system or in the future over a NQA network building a *Virtual Enterprise*.

4.2 NQA's General Concepts

The NQA system provides the following functions through a Hyperwave Information Server:

- A online manual formatted in HTML
- Work scenarios for planning, design, acceptance test, maintenance including:
 - Role plays and work instructions
 - Document flows
 - Activity flows
- A set of templates (HTML documents and forms) for planning, quality control, design, acceptance test and maintenance
- An automated project administration and workflow facility based on Hyperwave functionality:
 - Create and delete projects
 - Automated document-project phase correlation
 - Automated document status management

- Email based document submission
- Offline document manipulation

A description how these features were implemented in NQA Hyperwave 1.0 can be found in chapter 5.

4.2.1 Development by Configuration

Looking in the past it seems that society develops into 10% technology specialists and 90% users who (as non-technicians) have to continuously adapt themselves to new technocratic developments. To test another option some ISCN¹ [19] consortia (sub-groups sharing budgets for cost shared initiatives) designed a new development model, called "development by configuration". User Interface and functionality is configured from a pool of ready-to-use objects and data can easily be configured without any change of code. Such easy to configure systems allow non-technicians to become meta programmers who do not have to code but just configure to get their system [27].

This way of work creates larger design effort for the 10% technical specialists but in the long run this will make the 90% users to kind-of-programmers and will allow higher acceptance of technical solutions throughout the society (not only in specialised groups).

This paradigm bases on the fact that functionality is to be separated from data, and that data can be assigned with functionality by the user through configuration. NQA concepts are developed according to this principle and allow each organisation to insert their own documentation or result templates, and the NQA system then automatically generates (with the creation of objects from the templates) the functionality to the created objects. This way users can insert and maintain document or result templates and adapt the system to their own specific documentation requirements without any change or customization of code (just by configuration of data).

Figure 4.1 demonstrates, how data is linked to a defined set of functions. The user interface reflects the users view and adapts itself dynamically according to the chosen data and functionality.

4.2.2 NQA Product Strategy

The NQA product strategy is based on the concept of Development by Configuration presented in the last subsection. Instead of selling a ready-to-use application

¹ISCN WWW Homepage:
<http://www.iscn.ie>

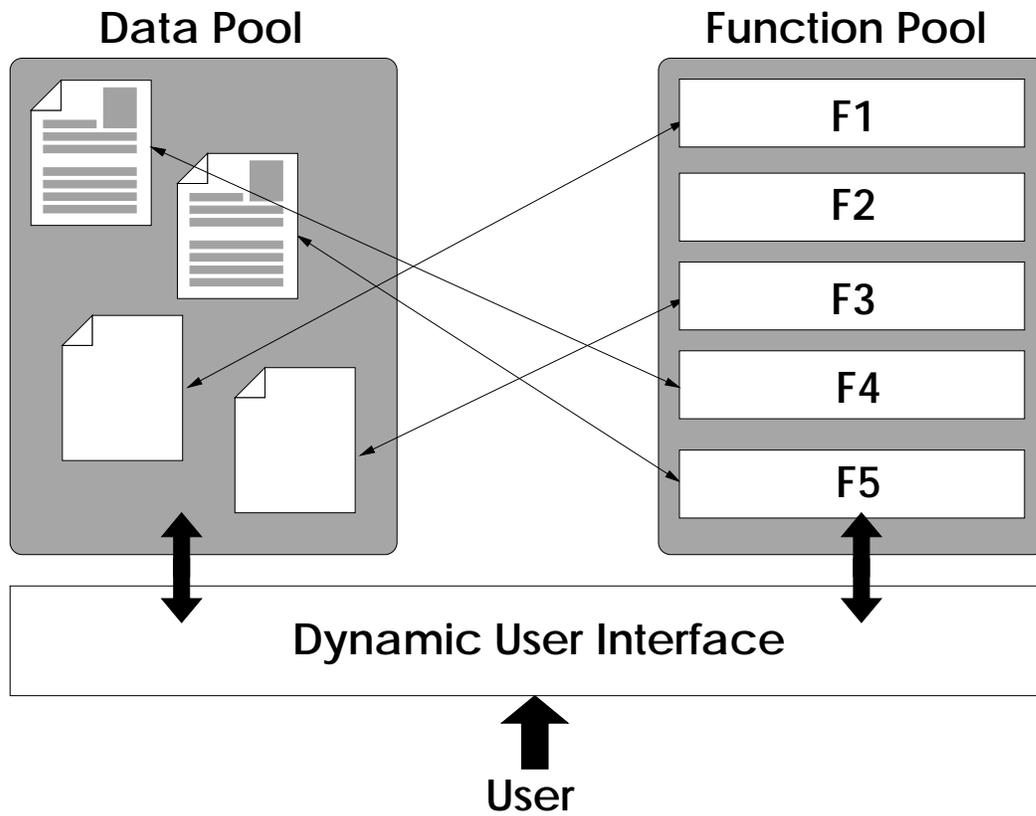


Figure 4.1: Development by Configuration

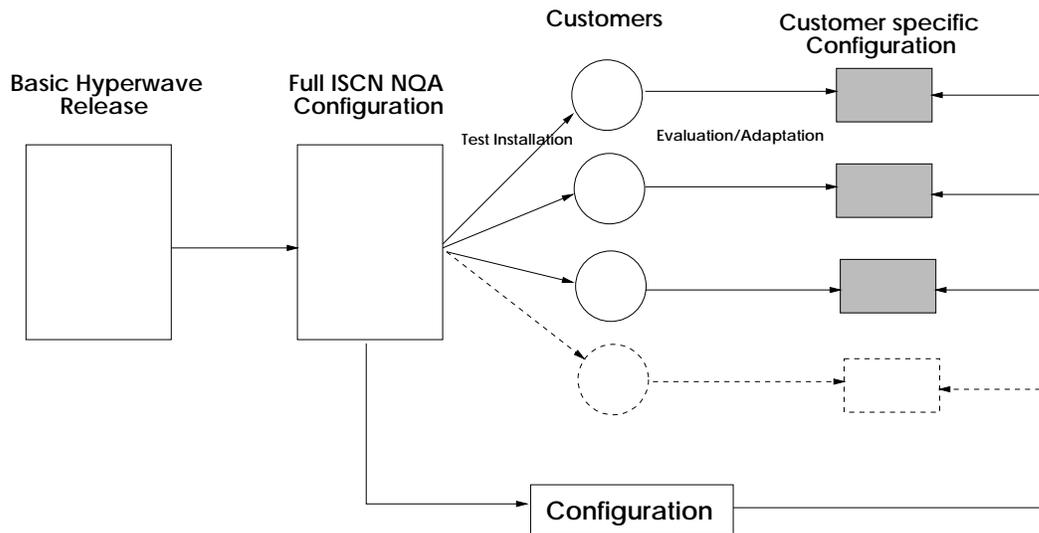


Figure 4.2: NQA Product Strategy

, ISCN sells a completely configured system, designed to meet the customers requirements.

As soon as a customer shows interest in the NQA system a so called standard installation for the customer is prepared. This standard installation was designed for a customer with no or just a weak quality management system and no or a not-reproducible work flow. Goal of the NQA system in its standard installation is to reach a ISO9001 certification [20][4] in a time as short as possible.

Most of the customers will already have a well established quality management system. So while the customer evaluates the standard installation, ISCN prepares a configuration which integrates well into the customers quality and work flow system. When the evaluation period ends the specific configuration is sold to the customer.

This strategy is the mean between a distributable product as the Hyperwave server itself is and a solution developed for a single customer as most of the Hyperwave applications are.

4.3 NQA in the Software Development

This section demonstrates how NQA could be used in the software development process. For demonstration purposes this example is restricted to the planning phase of a software project only. The standard installation of NQA Hyperwave 1.0 is configured exactly after the model presented here.

4.3.1 The Planning Activity Flow

The activity flow for the planning phase is presented in figure 4.3.

The activity flow for a project phase is predefined by the NQA system. In the Hyperwave implementation of NQA the project or team members just have to follow the given project structure top-down. In this way a standardized flow of activities can be guaranteed.

The sketches used in this chapter are taken from the online *Quality Manual* which comes with the standard, non customer specific, installation of NQA Hyperwave 1.0. The author of the *Quality Manual* is *Dr. Richard Messnarz*, Director of ISCN Ltd., Ireland.

4.3.2 Planning Documents Overview

An overview of planning documents is given in figure 4.4.

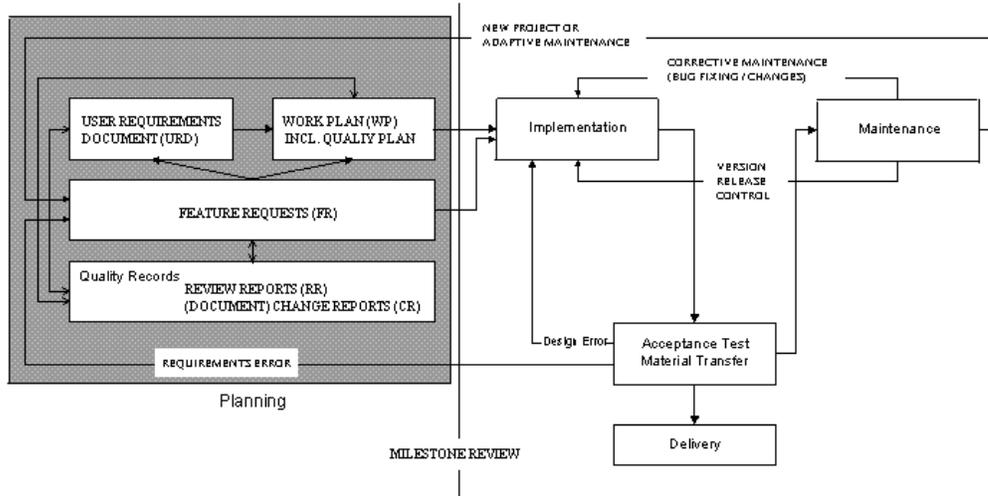
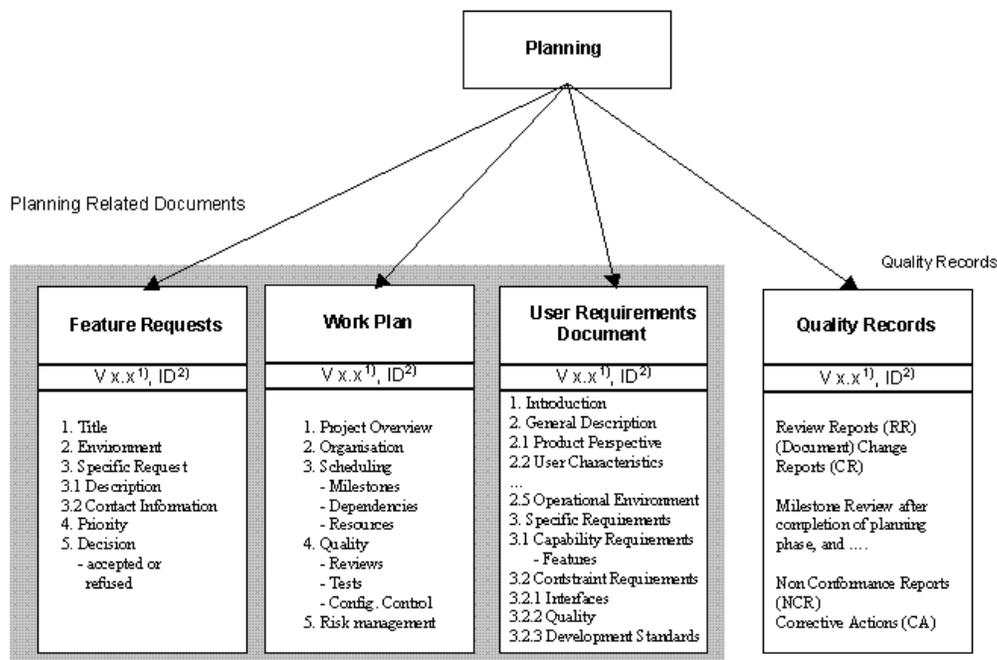


Figure 4.3: The Planning Activity Flow



1) Standard Information: Version of Document or Product, Authoring Team, Distribution List, Copyright, Status of Document
 2) Identification of Quality Procedures Used : Quality Manual Version, Checklist Version

Figure 4.4: Planning Documents Overview

The correlation between project phases and documents required in the actual phase is automatically created by the NQA system. The NQA system offers a set of template documents or forms on a per project phase base. The order of their creation is determined by the activity flow (see 4.3.1).

4.3.3 A Role Play for Planning

Behind each work scenario there is a set of roles to be played by the team members. Each role is described by responsibilities and by its interface to other roles (played by team members). This way the initiation of a project becomes the assignment of people to defined roles and interactions between the roles. Figure 4.5 shows the proposed role play for the planning scenario [28].

In the Hyperwave implementation of NQA the role play has not such a strong manifestation as the activity flow (see 4.3.1) and the relationship between documents (see 4.3.2). In version 1.0 there is a simple email submit function included, which should be used to create a simple workflow which in turn is the base for the assignment of roles. This email function in combination with the Hyperwave user administration can be used to simulate the role play proposed by NQA.

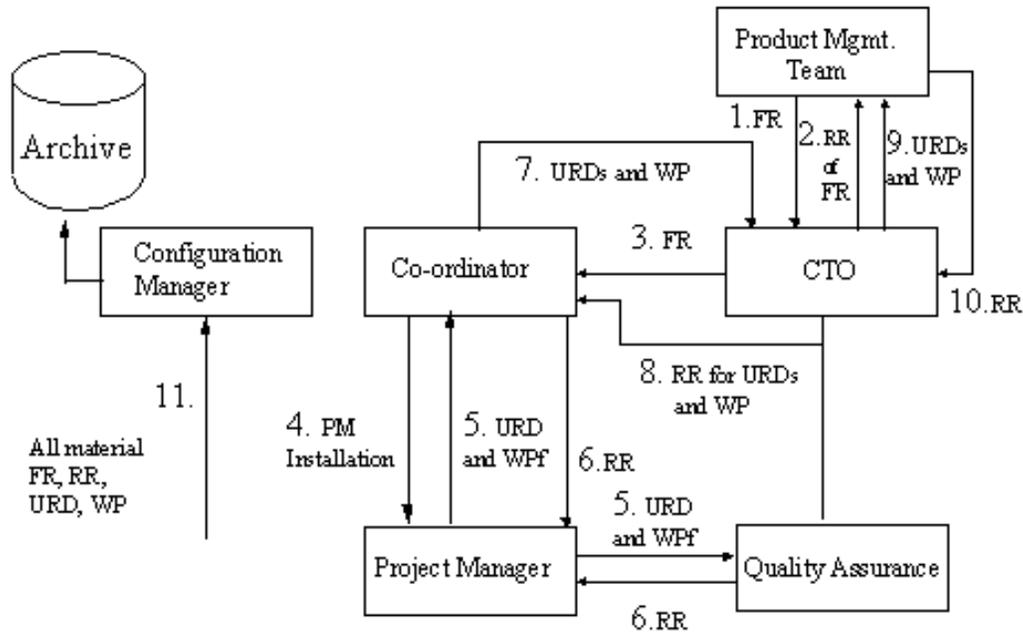


Figure 4.5: A Role Play for Planning

Work instructions for the planning scenario presented in figure 4.5:

1. The CTO receives new requirements, new ideas, and problems requiring a new architecture (thus the creation of a new project is needed).
2. The CTO forwards these system and schedule requirements to a responsible project manager either in form of a draft URD or in form of a RR done together with the customer.
3. The project manager refines the user requirements and establishes a draft WP and QP which both are reviewed by the CTO.
4. The project manager provides the quality assurance with draft URD, WP and QP.
5. The quality assurance reviews URD, WP and QP and documents all findings in RRs. The project manager refines the URD, WP and QP according to the quality assurance's RRs and achieves a test status *Reviewed* for the planning documents.
6. The project manager forwards the URD with test status *Reviewed* to the customer.
7. The project manager forwards the WP with test status *Reviewed* to the Customer.
8. The customer reviews the URD and WP and documents findings in RRs which are sent back to the project manager. Based on this feedback the project manager refines the URD and WP until acceptance by the customer is achieved.
9. The project manager provides the URD and WP which have been accepted by the customer to the COO who then makes a review of these agreed versions. He checks the project from the cost and contractual point of view and can initiate step 2 again.
10. If the COO accepts the URD and WP, all planning documents achieve a test status *Approved* and then the COO is writing a proposal (as basis for a contract) to the Customer.
11. The configuration manager is responsible for creating a project archive with directories for planning documents, development documents, quality records and maintenance documents and to archive all versions of all documents.

Abbreviations used in the work instructions:

Abbreviation	Meaning
CTO	Chief Technical Officer
COO	Chief Operating Officer
URD	User Requirements Document
WP	Work Plan
QP	Quality Plan

Chapter 5

Description of the Main NQA Features

All NQA Systems installed so far are different in some way. The reason for this is, that ISCN proposes a project structure and a separation into project phases. But in fact the customer will already have his well established project structure, so before selling and even installing NQA, the system has to be adapted to the customer's needs.

So what I will describe here are the NQA features which are the same for every customer and every installation. For demonstration purposes only I will use the phase model proposed by ISCN which represents the standard version of the NQA system.

5.1 Creating and Deleting Projects

The **Project Administration Table** is the entrance to the NQA system. Only Hyperwave users with administrative rights, that means that they have to be member of the Hyperwave user group *system*, are allowed to create or delete projects (fig. 5.1).

If the current user is recognized as member of the *system* group, the form for creating projects is displayed (fig. 5.2).

The field **Project Type** may change from customer to customer while **Project Name**, **Project Number** and **Access Rights** are part of every distribution. The **Project Name** is a free form string and will be mapped to the Hyperwave *Title* attribute. **Project Number** will be used to identify the project inside of the NQA system. This attribute is equivalent to the Hyperwave *Name* so the set of allowed

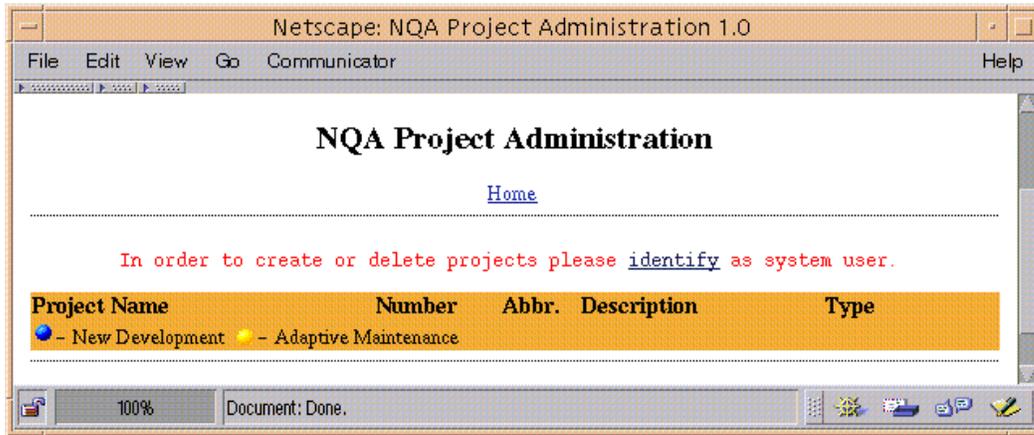


Figure 5.1: The Project Creation Dialogue for Not Authorized Users

characters is restricted to A-Z, a-z, 0-9 and _,-. It is not possible to create two projects with the same **Project Number**. The **Access Rights** are an integral part of the NQA security concept based upon of the Hyperwave user and group management. The access rights are also used to automatically inform selected users about changes in the project. I will talk about this when presenting the document management features.

The **Project Administration Table** is the central point in the NQA system. All project can be accessed from there (fig. 5.3). The selection of a project is done when following a link on the bottom of the table. As expected, only projects which can be accessed by the currently logged in user, appear in the table.

The Button for deleting projects is only displayed for *system* users. Projects to be deleted can be selected over checkboxes (fig. 5.4). There is no way to restore a deleted project (except of restoring the whole Hyperwave database from a backup file). All project entries and possibly existing sub projects are lost.

5.2 Handling the Phases of a Project

The **Project Sheet** is the overview page for a project. From here all project phases can be reached (fig. 5.5). The structure of this page depends on the customers requirements. Only the header and the entries for **Additional Documents** and **Subprojects** can be expected in every distribution.

The header simply shows the attributes given during the project creation process. The **Project Log** button opens a page of the form shown in figure 5.6. As described later, users or groups of users can be notified whenever a document

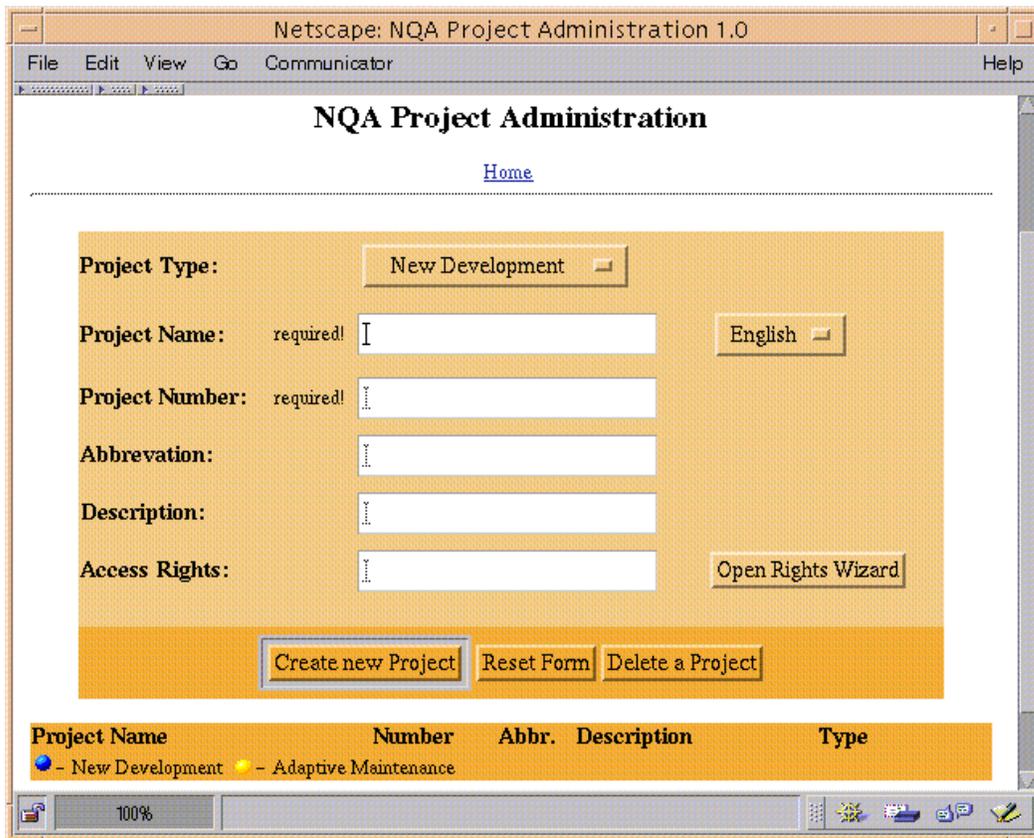


Figure 5.2: The Project Creation Dialogue for Authorized Users

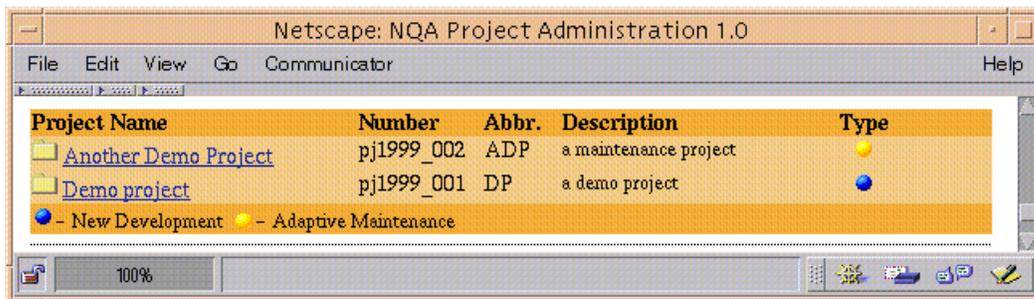


Figure 5.3: Listing of all Projects in the NQA System

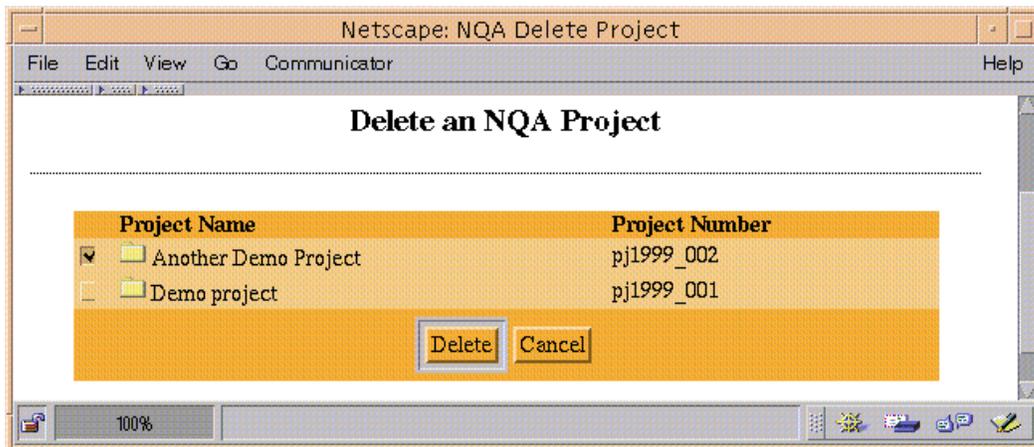


Figure 5.4: Dialogue for Deleting NQA Projects

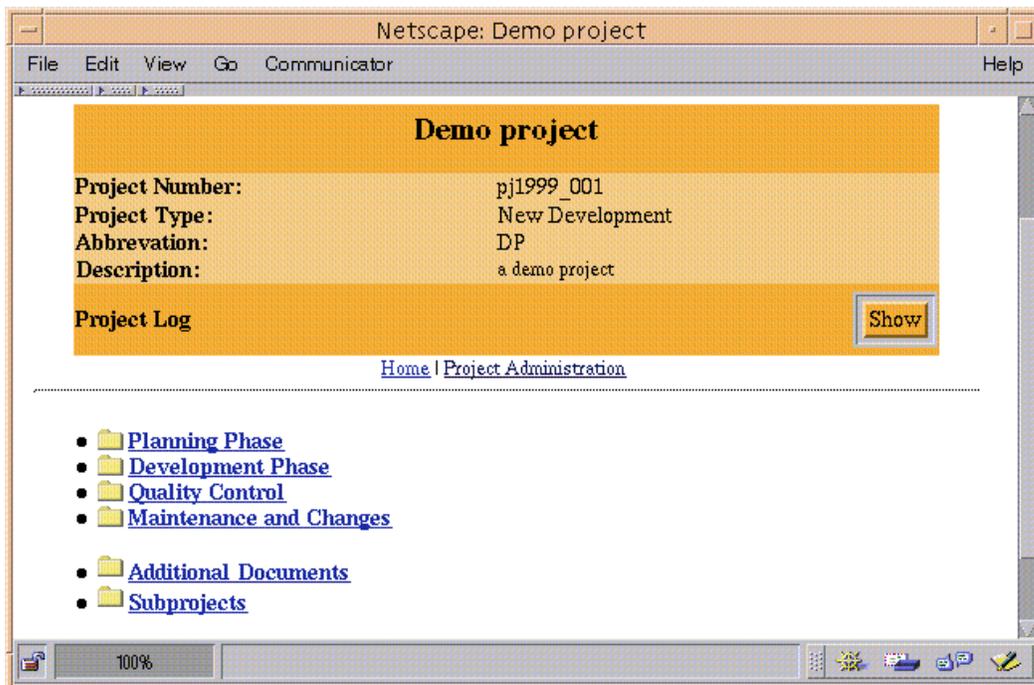


Figure 5.5: NQA Project Sheet

or report in the project changes. This notification is done via email. To guarantee trackability, each of this email transactions is logged separately for every project. The **Project Workflow Log** page is simply a graphical interface to this email log file.

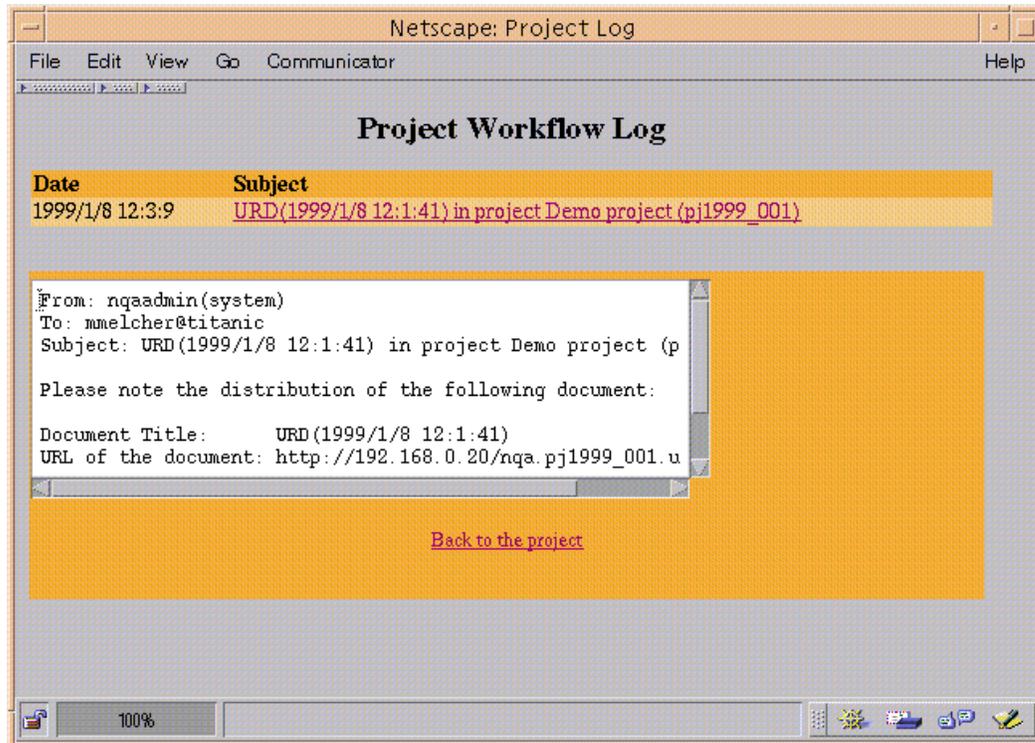


Figure 5.6: NQA Projects Workflow Log

When creating a project a Hyperwave collection for additional documents is created automatically. This is the place where extra information for a project should be stored. For the NQA system, extra data is information which is produced outside of the regular project's workflow or is add-on information to data included in the workflow. When opening this **Additional Documents** collection the user is confronted with the standard Hyperwave look and feel (fig. 5.7). So moving, copying or linking from outside of the NQA system into this collection is possible. A tool for integrating data into the NQA system is the so called *Interproject Link Generator* which is presented later.

NQA allows to structure a project hierarchically. Each project contains a subproject collection. In this collection projects can be created and deleted in the same way as *top level projects*. The difference is that every user who has write access to the project itself can create a subproject. The depth of the project tree produced in this way is not limited by the NQA system.

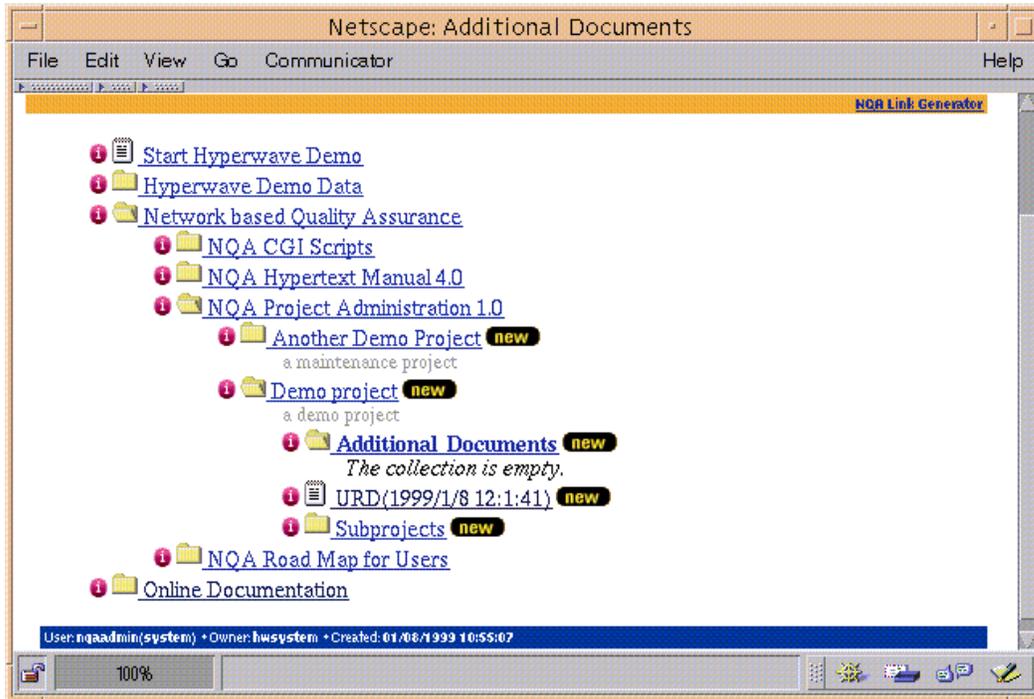


Figure 5.7: Additional Documents Collection

5.3 An NQA Project Phase

As mentioned above the project phases may vary from NQA customer to customer. Every project phase consists of a defined set of documents and reports. In the NQA system a *document* is a Hyperwave object which occurs exactly once in a project. A object which can have more than one occurrence is called a *report*. An example for this might be a *User Requirements Documents* which is unique to a project but possible customers can request a bulk of features. These requests are then handled as reports. Figure 5.8 shows this scenario.

A *Work Plan* is another example for a document. As long as there is no *Work Plan* created for the project a button would enable to do so. After importing the document this button is replaced by a link to the *Work Plan*. The changes from figure 5.8 to figure 5.9 show this process.

When adding reports to the project the interface for adding additional reports stays intact. The NQA systems assigns a unique number to each new report of the same type to distinguish them. While documents get a fixed name by the system the user can assign a custom name to reports. Figure 5.10 shows the situation after adding a report to the situation in figure 5.8.

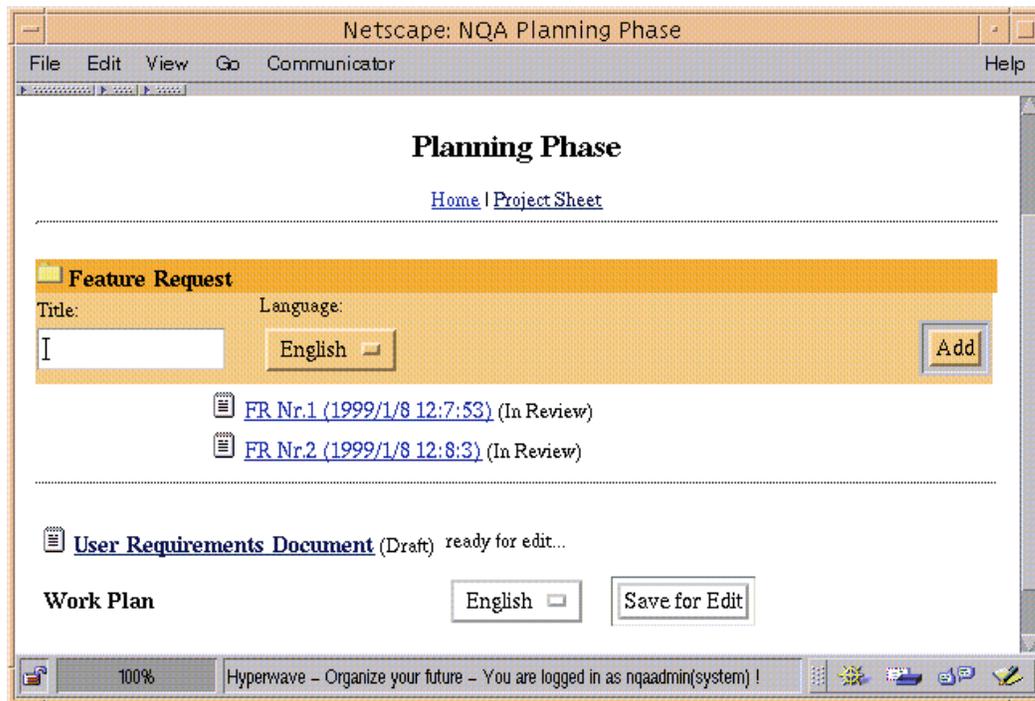


Figure 5.8: An NQA Project Phase

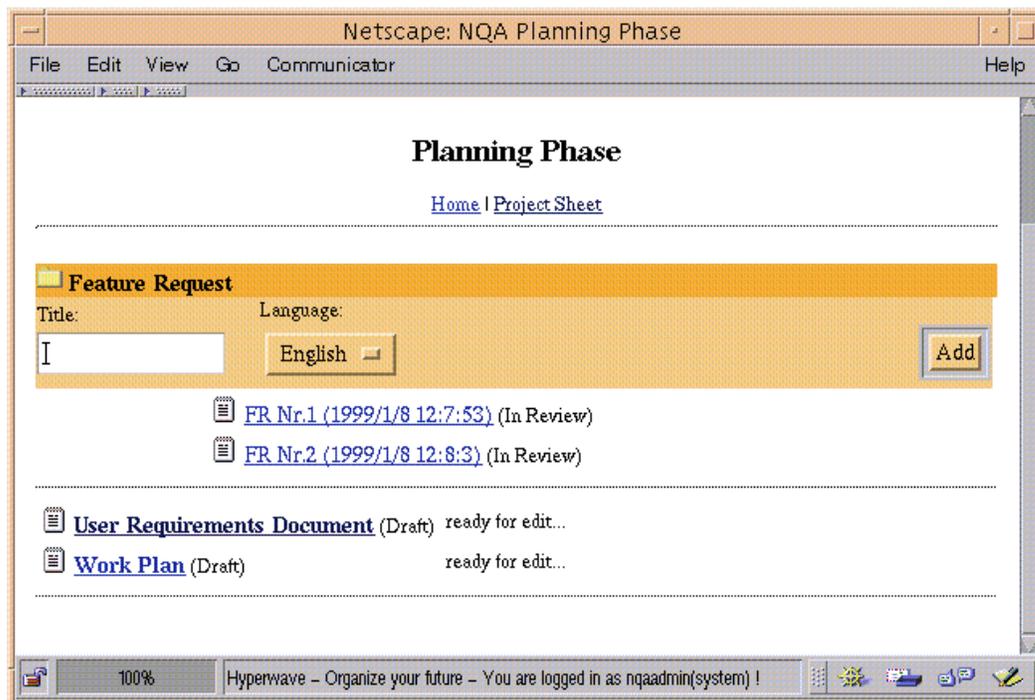


Figure 5.9: Adding a Document to a Project

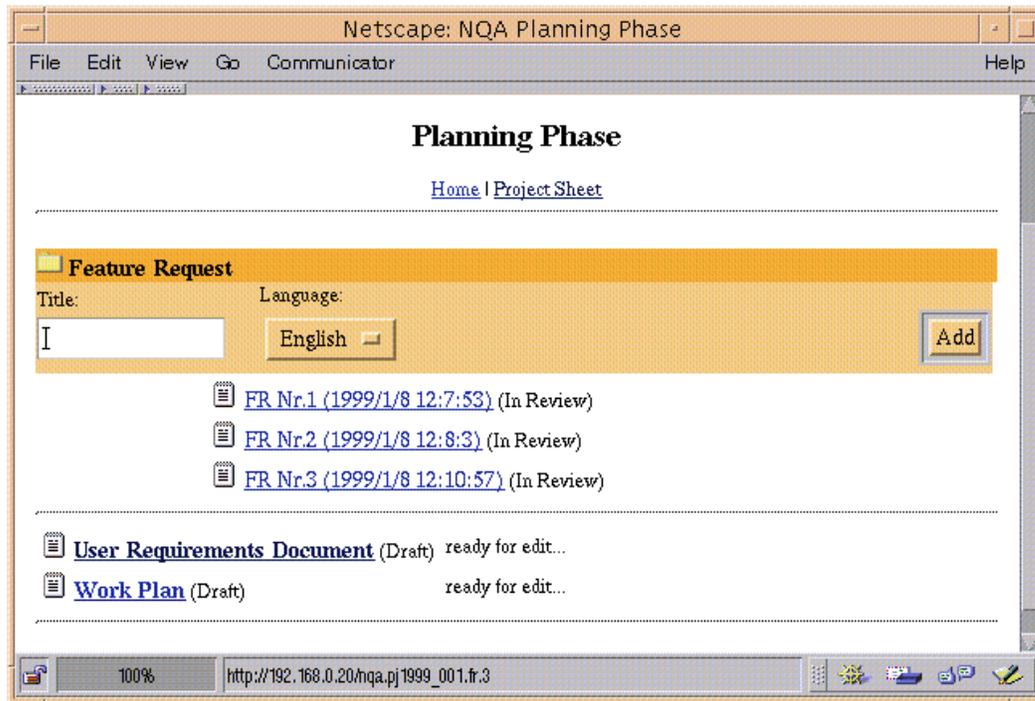


Figure 5.10: Adding a Report to a Project

5.4 NQA Document Management

The NQA document management functions consist of the following points:

- Administration of a recipients list for each report or document
- Notification of the listed recipients if the report or document changed
- Document or report version control
- Online and offline document and report modification possibilities
- Document state management

These functions are collected in the header of each document or report. Depending on the access rights and the state from the point of view of the version control, the header has as different appearance. In figure 5.11 a header for a document is shown where the currently logged in user has only read rights. When reading and writing is allowed the header will look like shown in figure 5.12.

It would be possible there to submit the actual document but I will talk about this special function in the next chapter. Consequently no modification

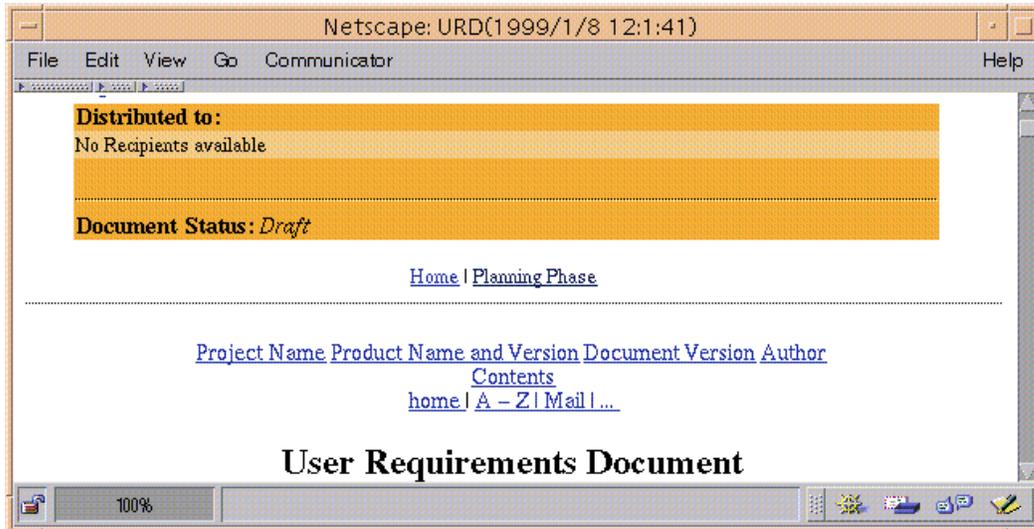


Figure 5.11: Document/Report Header for Read Access

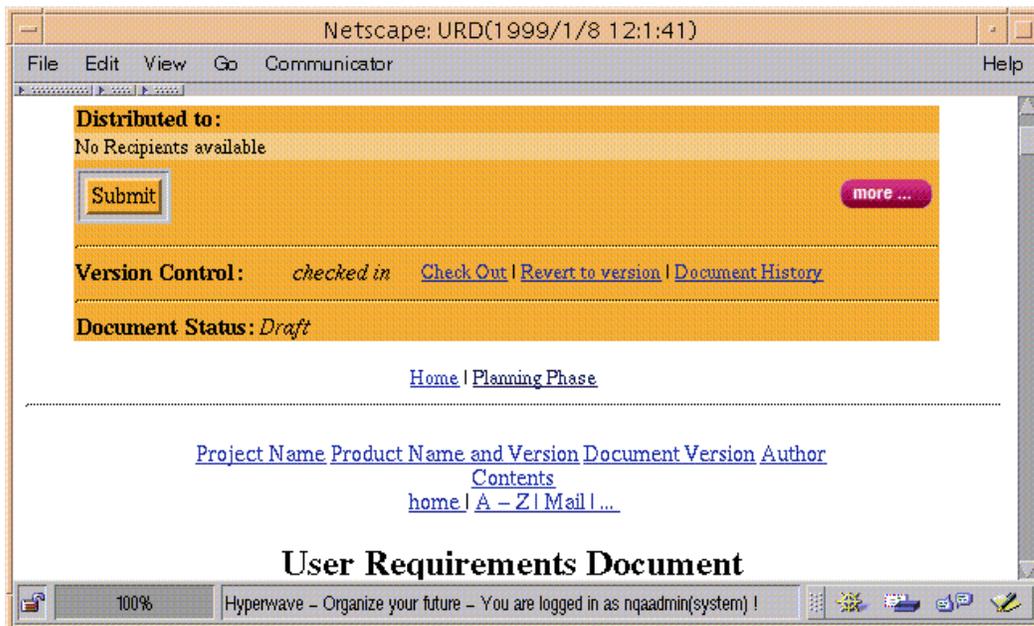


Figure 5.12: Document/Report Header for Read/Write Access, Checked In

possibilities are provided because the document is checked in. Depending if the document is checked in or out the version control bar in the header will change. In figure 5.12 the links **Check Out**, **Revert to version** and **Document History** are present. These links are leading to the standard Hyperwave version control dialogues, the use of them is described in depth in the *Hyperwave User's Guide*¹ [16].

When checking out the document the full set of modification functions becomes visible (fig. 5.13).

Edit Online simply leads to the Hyperwave document edit form described in [16].

Replace Document opens the Hyperwave dialogue for replacing objects on the server (see [16]).

Prepare to Download calls the Hyperwave *header and footer off* action. The document or report is then presented with no additional information. That means that nor a Hyperwave neither a NQA header or footer are displayed. In this form the *Save* function of the Web Browser can be used to store the document locally. After the offline manipulation it can be brought back on the server with the **Replace Document** function mentioned above.

Interproject Link Generator will be treated in its own section later on.

Throughout the lifetime of a project a document or report can change its state several times. How these states are named depends on the customer who uses the NQA system. In the NQA header a dynamic button represents these states. When clicking the button the next possible state is assigned to the actual document. The transition to a new state is shown in figure 5.14.

All documents and reports in NQA are strictly version controlled. Documents are automatically checked in when they are brought into the system and get a version number 1.0. Reports are checked out until they are modified, if text documents, or filled out, if forms, for the first time. Figure 5.15 shows the header, figure 5.16 the footer of a new report form. When the user fills out the form and pushes the **Save** button in the footer, the report is checked in, a standard NQA header is added to the report and the **Save** and **Reset** button in the footer disappear.

¹An online version is available under:
<http://www.hyperwave.de/user>

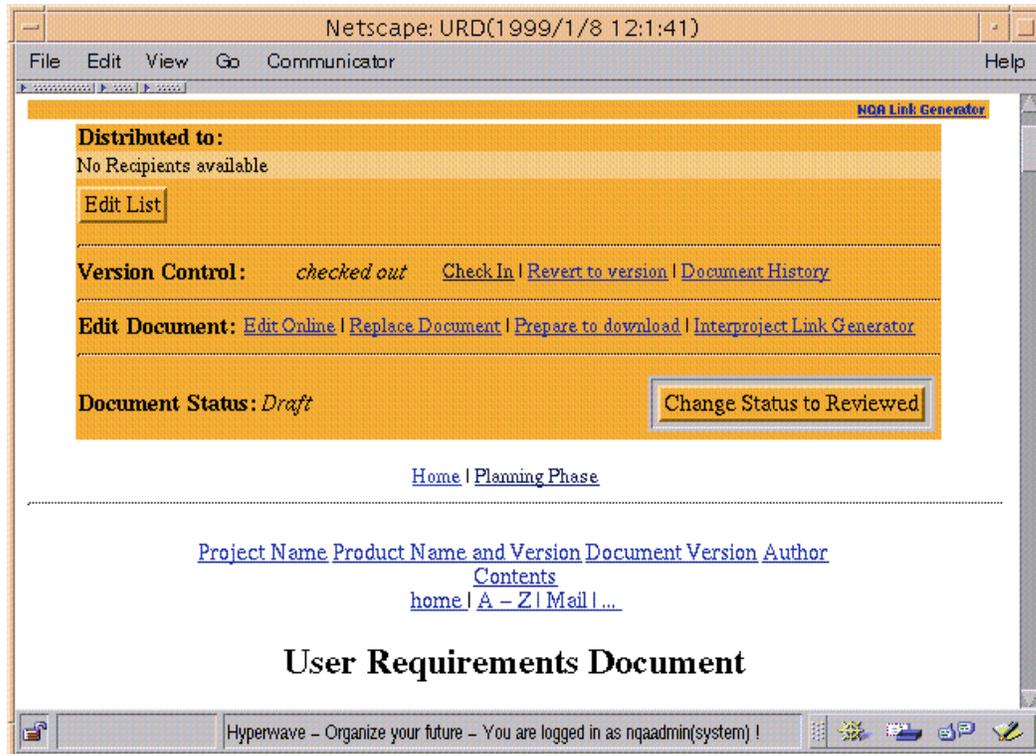


Figure 5.13: Document/Report Header for Read/Write Access, Checked Out

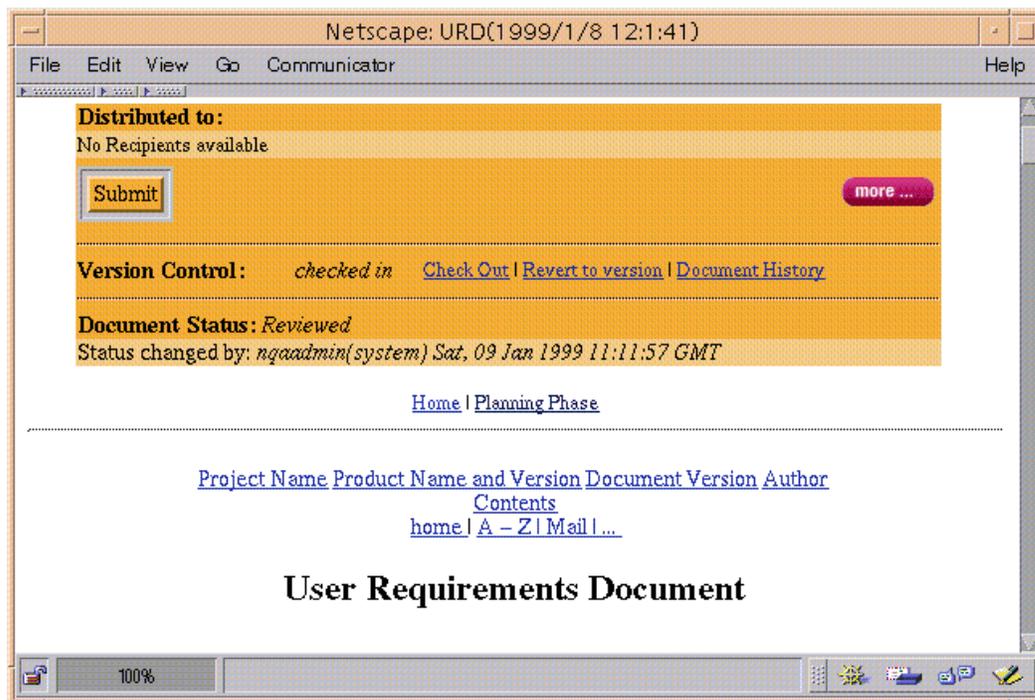


Figure 5.14: Document/Report Header with Changed Document State

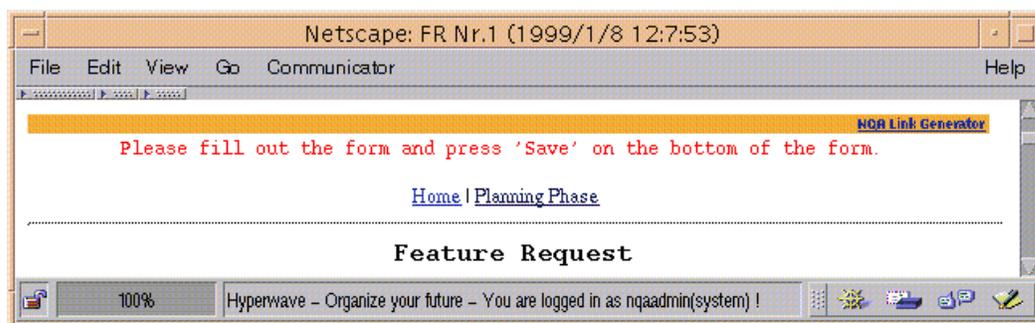


Figure 5.15: Report Header for a New Created Report

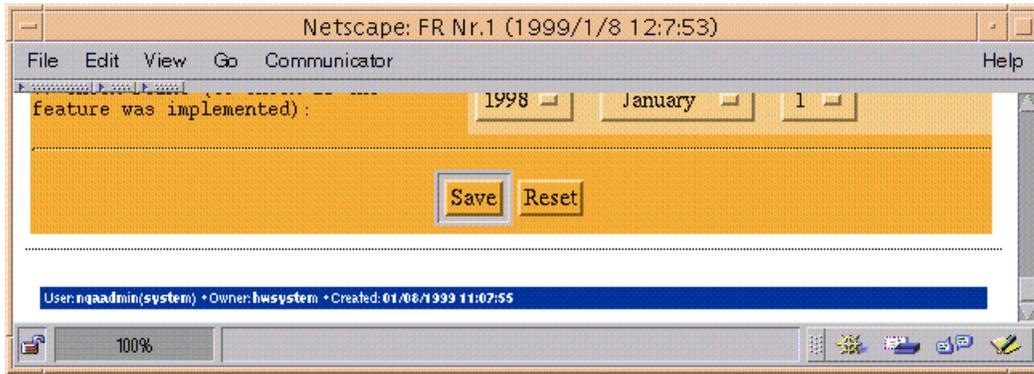


Figure 5.16: Report Footer for a New Created Report

5.5 Email Transactions

Checked out documents or reports which the currently logged in user can modify have a button **Edit List** in its NQA header. When pushing this button a dialogue for editing or creating a recipients list for this document or report is presented (fig 5.17). A recipients list is a list of email addresses to which a short message is sent when the **Submit** button of a checked in document or report is clicked.

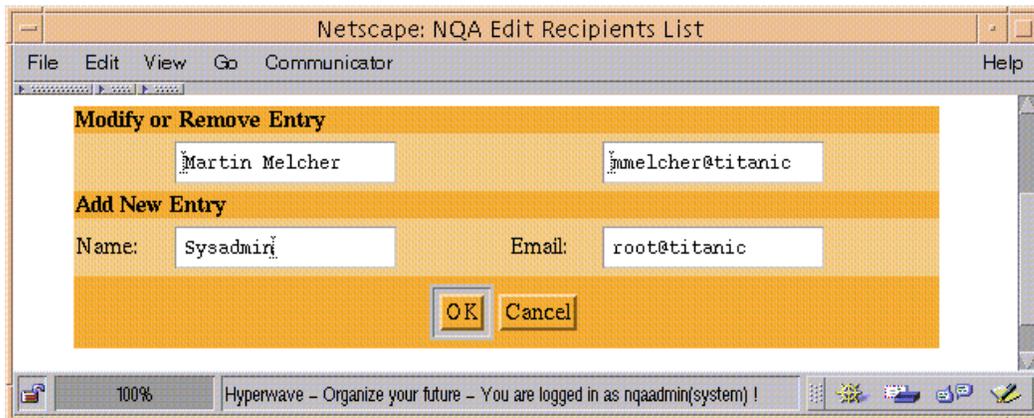


Figure 5.17: The Form for Editing the Recipients List

After editing the recipients list all people who will be notified are listed in the header (fig. 5.18). The **more** option extends the header in that way, that the sender of the email can be changed and an optional comment can be added. Figure 5.19 shows such an extended header.

After checking in the document or report the **Edit List** button becomes a

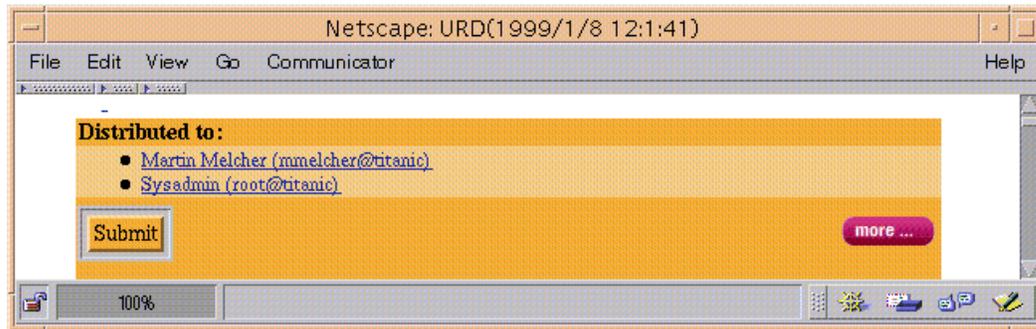


Figure 5.18: Document Header with Entries in the Recipients List

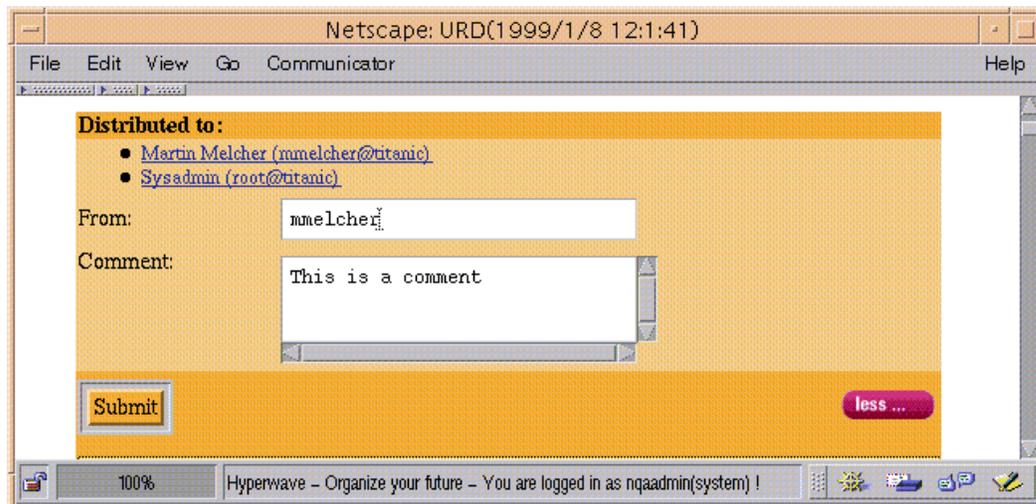


Figure 5.19: Extended Document Header

Submit button. Now it is possible to make a distribution of the current document. The feedback message (fig. 5.20) contains the following information:

- Who modified the document or report (**From**)
- Who was notified about the changes made (**To**)
- What document or report was changes (**Subject** and **Title of document**)
- Where can the document or report be found (**URL of document**)
- What is the new state of the document or report (**Document state**)
- Additional, optional comments

Exactly this record is also stored in the **Project Workflow Logfile** mentioned in section 5.2.

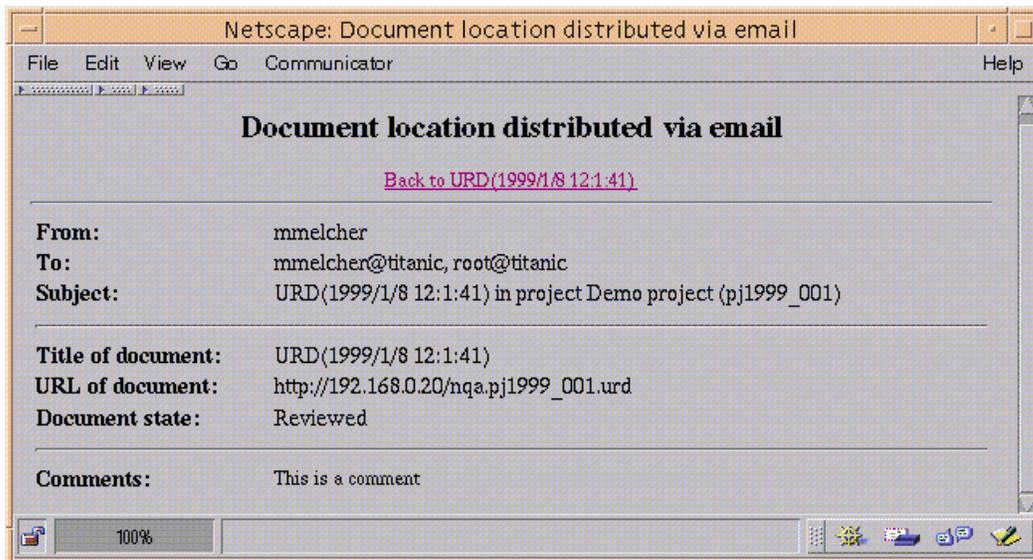


Figure 5.20: Email Submission Feedback

When creating projects, the Hyperwave *Rights Wizzard* can be used to assign access rights (see section 5.1). NQA is able to use these access rights to generate a recipients list automatically. NQA evaluates who has read access to the current object and forms a list of all concerned users. Precondition is that the Hyperwave user objects have a custom attribute *email* or *Email* set which is used here. If a Hyperwave group is used all users in this group are listed.

Figure 5.21 is an example with a group **nqa**. The users **nqaadmin**, **sysadmin** and **mmelcher** are members of this group. Group **nqa** has read rights for the current document. Checkboxes indicate who finally gets an email notification.

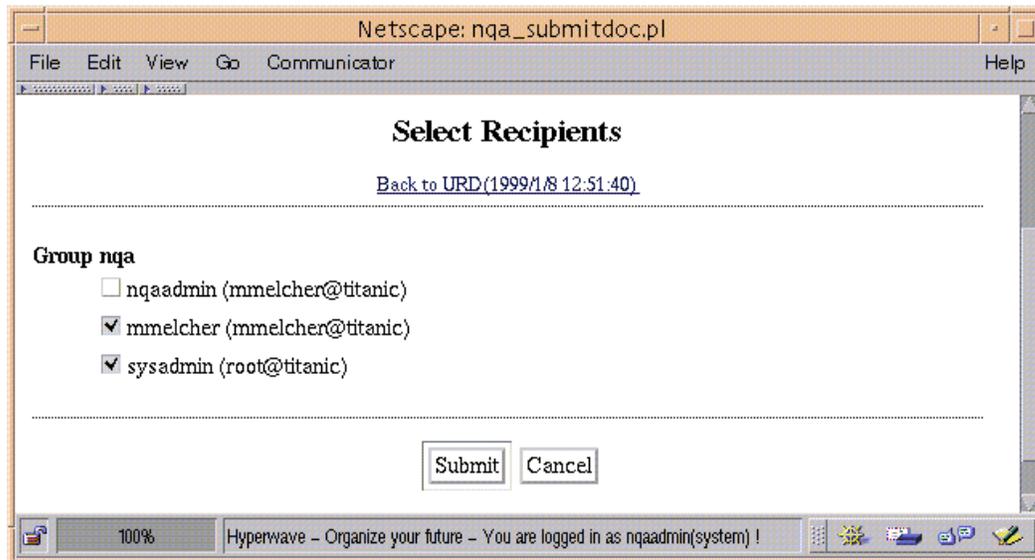


Figure 5.21: Generation of the Recipients from the Access Rights

5.6 Linking Reports and Documents

In a typical project some documents and reports are belonging to each other. In NQA these relations are marked with links. A very common example might be a *User Requirements Document* which is revised several times throughout the project's lifetime. For each revision a *Review Report* has to be inserted into the system.

In the figures 5.22, 5.23 and 5.24 exactly this scenario is presented. In figure 5.22 a new *Review Report* is created which should belong to an existing *User Requirements Document*. As figure 5.23 shows, the *User Requirements Document* is then listed in the *Review Report's* header. The *Review Report* can be reached through a link in the *User Requirements Document's* footer (fig. 5.24).

5.7 The Interproject Link Generator

The linking concept presented in the last section only works inside of an NQA project. But often it is required to make links to other projects in the NQA system or even to objects on the same Hyperwave server but not in the NQA system.

The *Interproject Link Generator* closes this gap. It allows the linking of two Hyperwave objects on the same Hyperwave server, it is not necessary, that these

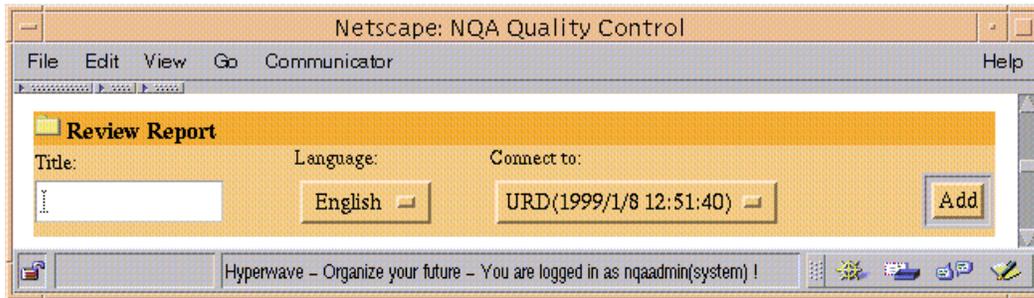


Figure 5.22: Creating a Review Report linked to a User Requirements Document

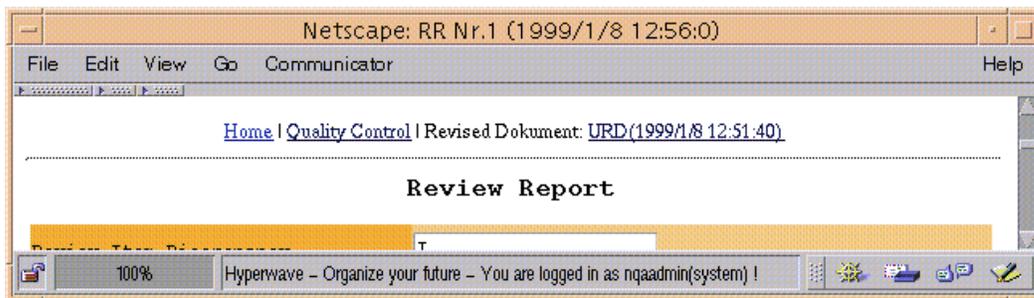


Figure 5.23: Header of a Linked Review Report

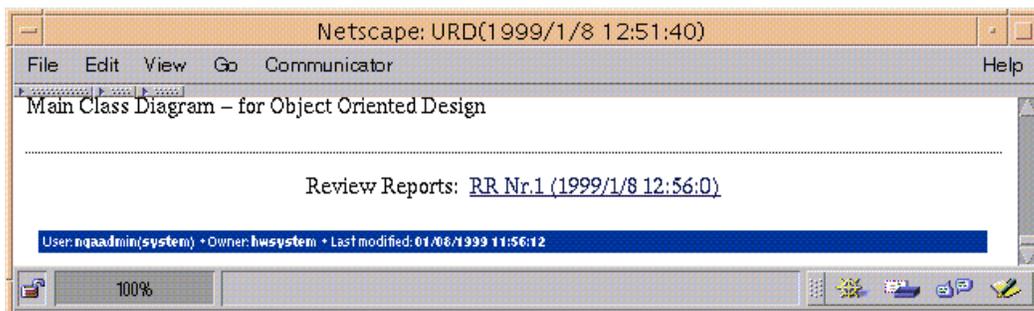


Figure 5.24: Footer of a Revised User Requirements Document

objects are in the same NQA project, they even don't have to be NQA documents or reports. A link created with this tool represents a relation between the objects involved in the link. Figure 5.25 demonstrates this concept:

There is a dedicated *source* and *destination object*. In the header of the *source object* a link to the *destination object* is established. We call this link the *forward link*. The *backward link* is the link leading from the footer of the *destination object* to the *source object*.

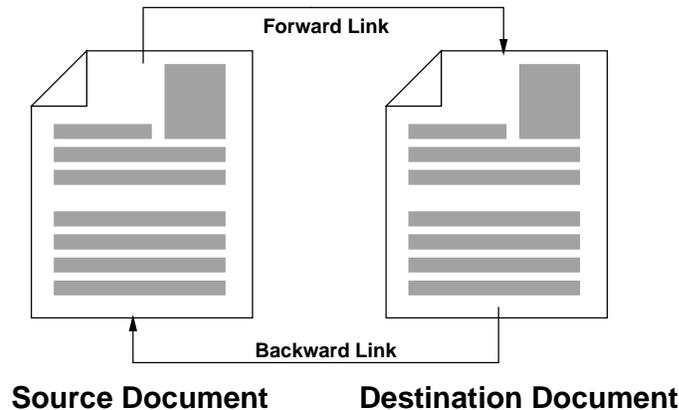


Figure 5.25: Concept of the Interproject Link Generator

A practical example will demonstrate the power of the *Interproject Link Generator*. A *User Requirements Document* in a special project has already been created. In a closed project in the NQA system there is a *Software User Manual* which should be referenced by the new *User Requirements Document*. What has to be done is the creation of a link which connects the *User Requirements Document* and the *Software User Manual*. There the *Interproject Link Generator* has to be used because the link partners are located in different projects.

The *User Requirements Document* will be chosen as *source object*, so the link generator will be started from there (see figure 5.26).

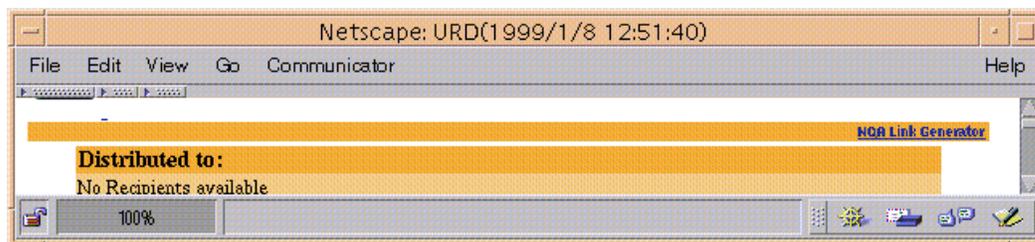


Figure 5.26: Interproject Link Generator Bar in the Header of a Document

After starting the tool a dialogue like shown in figure 5.27 is presented. The *Software User Manual* can now be selected as link destination. The changes in the *User Requirements Document's* header and the *Software User Manual's* footer are shown in figure 5.28 and 5.29.

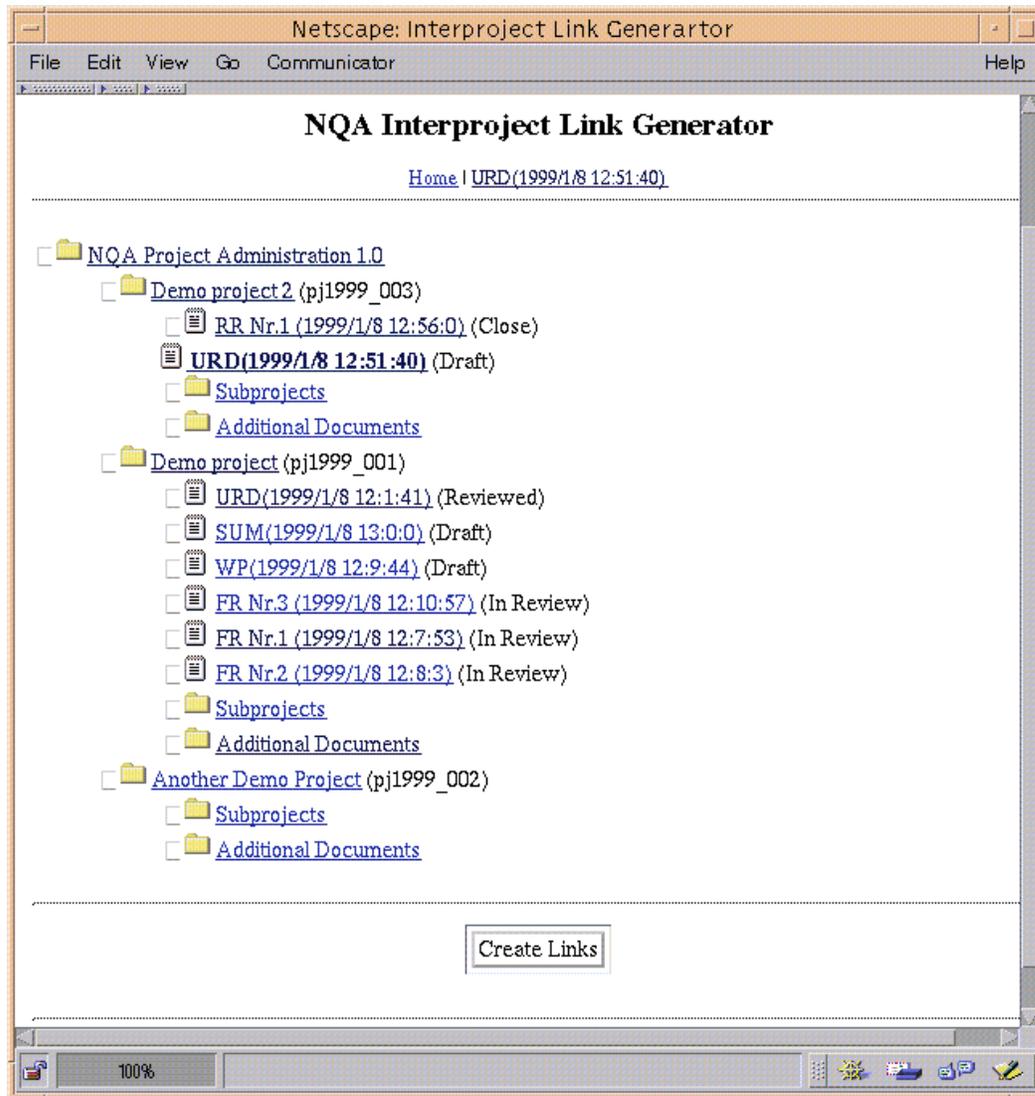


Figure 5.27: The Interproject Link Generator Dialogue

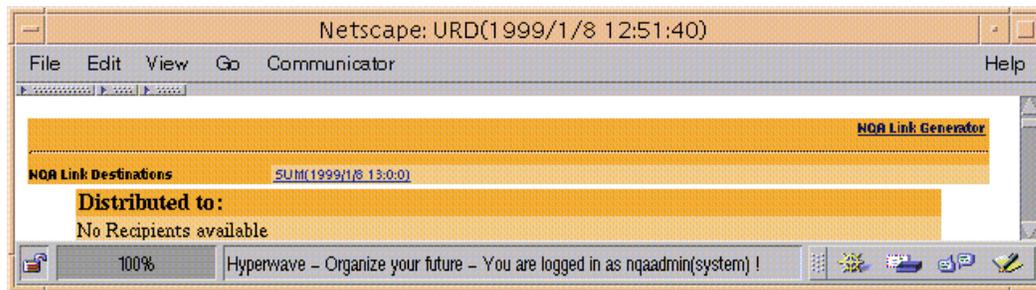


Figure 5.28: The Interproject Link Generator Header Bar with Link Destinations Listed

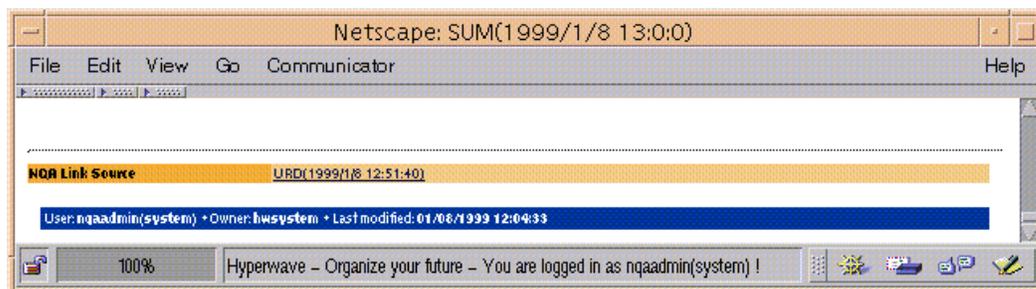


Figure 5.29: The Interproject Link Generator Footer Bar with Link Sources Listed

Chapter 6

Hyperwave as Platform for Web Based Applications

6.1 Hyperwave Features and Benefits

The *Hyperwave Technical White Paper*[15]¹ from may 1998 gives a complete list of the features of the Hyperwave Information Server version 4.1. The features discussed here are the ones important for programming the Hyperwave system from my personal point of view.

6.1.1 Problems in the “First Generation” World Wide Web

Navigating large hypertexts is difficult. As the number of documents and associated hyperlinks grow, users find it increasingly hard to get an overview, find the information they are looking for, locate or relocate new or updated information, learn how much information exists on a certain topic, make sure that they see every piece relevant to a certain topic, and recognize outdated information. This is known as the “lost in hyperspace” syndrome [15].

Links pointing to nowhere, that means, links which lost their destination are a usual phenomenon in the World Wide Web of today. This is simply a result of the fact, that a Web page (or the author of Web page) doesn’t know if it is the destination of link. When being removed, the link destination is lost too.

When inserting a new document A, it is required to edit at least one other document B, in order to make a link to the new document. This is not only

¹The online version can be found under
<http://www.hyperwave.com/whitepaper>

additional work, it also means that authors must have write access to document B, which would allow them to modify the contents of B or remove links to other documents [15].

These are just a few problems of the World Wide Web, or problems users having with it. Hyperwave, as a “*Next Generation Web Solution*” [25] offers solutions to the most of these “first generation” problems.

6.1.2 Hyperwave Structural Elements

One way to fight against the “lost in hyperspace” syndrome is to offer structural elements to organize data in a “natural” manner. In other words, the organization and grouping of data in containers is much more human-like than the use of hyperlinks in order to create relations.

A container contains a number of other elements, which could be documents or other containers. There are a number of predefined container classes: Collection, Sequence, MultiCluster, and AlternativeCluster [15].

The use of containers is a good replacement for *structural (navigational) links*. The author of a Web site in the Hyperwave system will just create *referencial links*, the rest (structural links) are defined through the chosen container structure and are maintained by the system. Tangling links (such without a destination) or no longer possible.

6.1.3 Hyperwave Metadata

Unlike many other Web-based systems that store documents as plain files in a file system, Hyperwave Information Server stores documents as objects in an object-oriented repository, together with other information objects (i.e. structure elements). These objects carry metadata, i.e. attribute names and values. These documents can also be of any type [15].

The metadata paradigm makes Hyperwave a full featured database. Just a few of the advantages of metadata are, that they

- describe the content of a document in more detail
- can be indexed and searched
- simplify the integration of Hyperwave in existing environments.

6.1.4 Hyperwave Programmability

The basis of the user interface is HTML and JavaScript. For newer browsers (Netscape Communicator 4.x, Microsoft Internet Explorer 4.x), Information Server makes use of Dynamic HTML (DHTML), Cascading Style Sheets (CSS), and JavaScript 1.2. The server automatically senses the browser version, and sends the corresponding instructions to the client. The interface for older browsers needs more images, and is slower.

Hyperwave Information Server dynamically assembles the HTML sent to the browser out of predefined, customizable building blocks. This process is controlled by an HTML extension called PLACE, which is a small macro language. Basically, PLACE macros are evaluated to HTML code.

In addition to the relatively simple PLACE language, Hyperwave Information Server also supports server-side JavaScript (SSJS). Together with a JavaScript interface to the Hyperwave API, this offers application programmers a very high degree of flexibility in design of their applications. The JavaScript code is pre-compiled and executed directly in the server, which yields high performance. The Hyperwave API gives access to all functions of the server, thus enabling to extend the server with almost arbitrary functionality [15].

6.1.5 Other Features

Other important features of the Hyperwave systems are:

Publishing Wizard: Microsoft's *Webpost API* was extended to work for Hyperwave servers.

ODMA: The *Open Document Management API (ODMA)* is a standard API for reading and storing documents from/to document management systems. Hyperwave Information Server also supports it.

Virtual Folders: The structure of a Hyperwave Information Server can be seamlessly integrated in the MS Windows Explorer.

Version Control

Annotation of server contents

Security through user and group management

6.2 Hyperwave Programming Paradigms

The *Hyperwave Programmer's Guide*² [11] is a valuable, and one of the few sources for Hyperwave programmers. This section is based on this guide. I have added some aspects from the practical point of view and some ideas collected in a Hyperwave PLACE Workshop [8].

6.2.1 PLACE

PLACE is a meta-HTML language, with which the appearance and function of WaveMaster's user interface can be configured. The name PLACE comes from the so-called *placeholders* used in the language. Placeholders transfer various types of information between the server and WaveMaster, including information about the current accessed collection or cluster, about the Hyperwave Information Server WaveMaster is connected to, the user currently logged in to the server, etc. A PLACE template is a file which contains HTML and PLACE statements and which is applied to objects which users access. Such a template could, for example, check if an error occurred and display the appropriate message, then if no error occurred proceed to display the appropriate buttons at the top of the page (for example identify and search buttons, and, if the user is identified, an "edit" button). Then it could check what type of object is being accessed at which point it would display a list of links if the object is a collection or display the text if it is a text object, etc. Lastly, it might display the user name of the current user and show links to the parent objects of the current object [11].

6.2.1.1 A PLACE Model

In order to use the PLACE language to modify the WaveMaster user interface, it might be useful to have a picture of what is going on behind the scenes. Of course, it is not necessary to know exactly how the WaveMaster communicates with the Hyperwave Server but a knowledge of the data flow can simplify programmer's work.

This sketch has to be understood as a model for the programmer, it is not a description of the processes and tasks working inside of WaveMaster.

Whenever a client requests a document from the Hyperwave server, this document is processed by the WaveMaster. Depending on the type of document a

²The newest version of the *Hyperwave Programmer's Guide* is located on the Hyperwave Web server:

<http://www.hyperwave.de/program>.

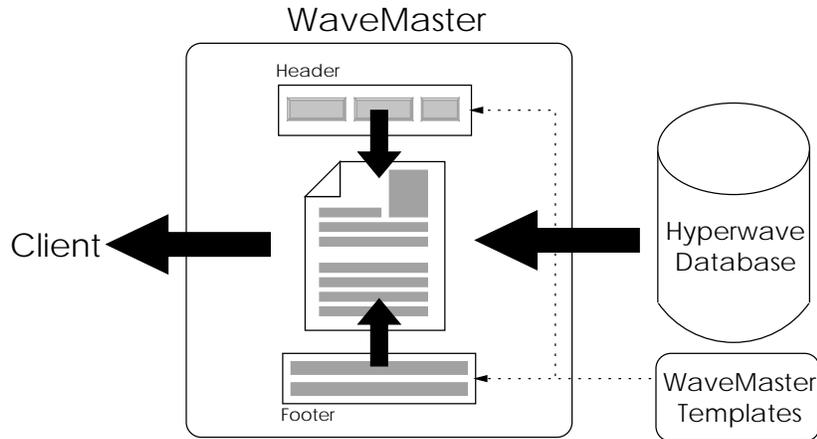


Figure 6.1: A PLACE Model

template for displaying the content of the document is chosen. In the case where the content is one which can be handled by a standard web browser, the requested document is delivered as it is. Often a menu bar or navigation facilities will be added by WaveMaster. But a Hyperwave server may serve more than just the MIME based documents, as usual in a regular Web server. If you look at Hyperwave Collections, Clusters, Sequences etc. a Web browser will hardly be able to handle a content like this. In this case it is up to WaveMaster to transform the Hyperwave internal representation of such a data into a format which can be displayed by a standard Web browser (mostly a conversion to HTML will meet the users requirements; dynamically configured Java Applets are just another possibility).

The kind of how WaveMaster handles and formats different content types can be configured through so called WaveMaster Templates. Intuitively such a template describes the look of a HTML page, where dynamic data is replaced by placeholders. Whenever WaveMaster delivers a document, it chooses an appropriate template (in fact there is only one template, but more about that later) and fills the placeholders in the template with values.

As mentioned in the introduction of this chapter, the name PLACE is derived from the placeholders used in the templates. If you examine your Hyperwave server's WaveMaster directory, you will find a scaring amount of HTML files. But when WaveMaster is in action, it will clue together all these files with an include mechanism. So when I talk about PLACE Templates, I mean the one and only PLACE Template in reality.

WaveMaster's entry point to all these template fragments is the `master.html` file. That is the file where `<HTML>` and `</HTML>` are located. All in between is an extremely complex conglomeration of macro calls and branch instructions.

WaveMaster will expect `master.html` on a defined place in the file system. The Hyperwave `PLACETemplate` attribute allows it to take the master file from the server itself. In this way you can hold different sets of templates and assign them to parts of the Hyperwave server instead of having one global `PLACE Template`. So several organizations could share one server, each having a different user interface design.

6.2.1.2 Basic Place Syntax

`PLACE` statements are put into normal HTML files and are surrounded by HTML commands or normal text. All `PLACE` constructs are enclosed by two percentage symbols (`%`) to distinguish them. A typical `PLACE` statement is for example `%%object.title%%`. When a document is retrieved, this `PLACE` statement is evaluated to the title of the current object. WaveMaster interprets the `PLACE` statement every time a document is retrieved [11].

Comments

Comments in `PLACE` are enclosed between `%%#` and `%`.

```
%%# this is a comment %
```

If-Statement

If-statements are used to branch off in the `PLACE` code according to whether particular conditions are fulfilled or not. Expressions are evaluated using the operators:

<code>==</code>		is equal to
<code>!=</code>		is not equal to
<code>></code>		is greater than
<code><</code>		is less than
<code>>=</code>		is greater than or equal to
<code><=</code>		is less than or equal to

The basic syntax for the if-statement is:

```
%%if expression%%
[HTML or more PLACE]
%%else%%
[HTML or more PLACE]
%%endif%%
```

A combination of expression is possible using the boolean operator AND (&&), OR (||) and NOT (!). The example shows the combination of two expressions using the AND (&&) operator:

```
%%if session.user.get_attrib(firstname)=='Bill' &&
    session.user.get_attrib(lastname)=='Gates' %%
    <B>Sorry, access denied!<\B>
%%else%%
    <B>You are welcome!</B>
%%endif%%
```

While-Statement

The while-statement is the only loop statement in PLACE. It is used to step through lists of dynamic length. A for statement would be far beyond the scope of PLACE; PLACE is not a programming language and so there is no possibility to hold count variables needed for for-loops. The while-statement is a good decision for dynamic lists, the only thing WaveMaster and the programmer has to know is, if there are still more elements in the list.

An expression can be formed using the same rules described for the if-statement:

```
%%while expression%%
[HTML or more PLACE]
%%endwhile%%
```

The listing of a collection's children is a good example for a while-loop:

```
%%while coll.next_entry%%
    <B>Title: %%coll.entry.title%%</B>
%%endwhile%%
```

Include and Hyperinclude Statement

As mentioned earlier in this chapter, the *master.html* is composed of several sub templates. At runtime these templates are glued together as defined by include statements.

In the standard Hyperwave distribution all includes are collected at one single point, that's the *hyperincludes.html* file in the WaveMaster directory. An include entry will look like the following:

```
%%include 'file name' %%
```

where *file name* is the full file name with path relative to the WaveMaster directory. If the master template file is located on the server and not on the file system, any files you want to include in the template must also be uploaded to the server and can be included with the `hyperinclude` statement:

```
%%hyperinclude 'name'%%
```

where *name* is the name or global object identifier of the text to be included.

Macro Definition

I have to point out once again that PLACE is not a programming language. That is the reason why there are no subroutines, functions, procedures or however you may call it in PLACE. Nevertheless PLACE macros are a good way to organize templates in a function like manner. Of course, you can not pass arguments to a macro but the organization of the template in modules raises readability and maintainability.

Macros have to start with

```
%%macro name%%
```

and end with

```
%%endmacro%%
```

with HTML or PLACE in between these delimiters. Note that the macro itself must appear in the template at a point before it is called. Also note that macros can call other macros which in turn can call other macros for up to as many levels as you want.

The definition of a macro is demonstrated through a short example: Imagine there is a piece of HTML which occurs several times throughout your template. Instead of repeating the same tags every time you need them, you could define a macro:

```
%%macro author%%
<IMG SRC='me.jpg' ALT='mmelcher'>
<A HREF='mailto:mmelcher@iicm.edu'>Martin Melcher</A>
%%endmacro%%
```

This macro is called by inserting `%%author%%` anywhere in the place code.

6.2.1.3 Applied PLACE Programming

Providing Multilinguality

One of Hyperwave's main advantages is the possibility to have one document in different languages. Depending on the user's preferences the document in the according language is displayed. Not only the documents on the server but also the user interface should change its language, when requested by the user.

The placeholder `%%session.language%%` holds the user's current language setting as a Hyperwave language string.

Valid Hyperwave language strings are:

Language String	Language
en	English
ge	German
fr	French
it	Italian
sp	Spanish
jp	Japanese

There are two ways for determining the current language, one is the use of the placeholder presented above:

```
%%if session.language == 'sp'%%
  Usario
%%else%%
  %%if session.language == 'ge'%%
    Benutzer
  %%else%%
    User
  %%endif%%
%%endif%%
```

The more elegant and compact form would be:

```
%%sp:Usario, ge:Benutzer, :User%%
```

PLACE Actions

Placeholders come in many types: there are placeholders that have to do with

annotations, clusters, collections, the search function, the server itself, etc., just to name a few. A further type of placeholder are *actions*. Actions are the only type of placeholder which is used to actually perform an action instead of to communicate a value to WaveMaster. An action is a process with which certain information is retrieved from the server or altered in some way. To achieve this WaveMaster has to know about the actions and their parameters to communicate with the Hyperwave Information Server [11].

Action calls are always embedded in anchors. Whenever the user clicks this anchor, a special variable *action* in the WaveMaster is set to a value defined by the action call (the anchor).

A part of the `master.html` template is dedicated to evaluate this *action* variable. Depending on the value of *action* another set of PLACE/HTML statements is delivered to the client.

Some actions in the Hyperwave system are predefined, that means that they don't have to be caught in the master template.

```
<A HREF=' '%action.call.home%' '>Go Home!</A>
```

would automatically produce a link leading to the current user's home collection.

The programmer has the possibility to define his own actions. In this case it is up to him to handle this action in the master template.

```
<A HREF=' '%action.call(user_defined.action)%%' '>a user defined action</A>
```

```
%%if action == 'user_defined.action' %%  
[do something]  
%%endif%%
```

PLACE in Combination with Server Side JavaScript

Java Script is a lightweight interpreted programming language with rudimentary object-oriented capabilities. The general-purpose core of the language has been embedded in Netscape Navigator and other Web browsers and embellished for Web programming with the addition of objects that represent the Web browser window and its contents. This “client side” version of JavaScript allows “executable content” to be included in the web page—it means that a Web page need no longer be static HTML, but can include dynamic programs that interact with the user, control the browser, and dynamically create HTML content [7].

You can look at PLACE as just another formatting language like HTML, in the Hyperwave manual it is called meta-HTML. Often PLACE is not powerful enough to solve a problem. Whenever you need the power of a real programming language and a high-level abstraction of a Hyperwave sever, client side JavaScript in combination with PLACE is a good choice.

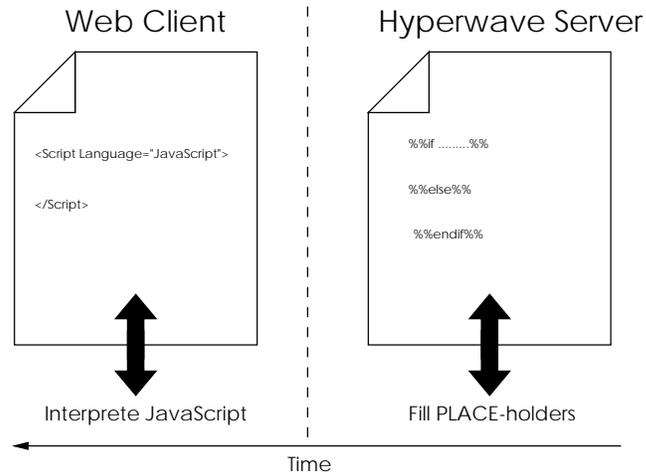


Figure 6.2: PLACE in Combination with JavaScript

PLACE and JavaScript are handled at different points and different times in the transaction chain. The WaveMaster which is responsible for parsing and filling PLACE templates does not care about JavaScript, it treats JavaScript like standard HTML. When it is time for the Web client to interpret the JavaScript code, the PLACE template is already transformed into a HTML file which can be displayed by the browser, there are no %% enclosed parts left.

This short example will demonstrate the co-operation between PLACE and JavaScript:

Given is the following situation: You have a collection with a set of documents in it. An unknown number of these documents has a custom Hyperwave attribute `countme=true` attached. When the user opens this collection you want to show the number of documents with `countme` set to `true`. In a combination of PLACE and JavaScript the solution looks like this:

```
<SCRIPT LANGUAGE=' 'JavaScript' '>
<!--
var count=0;
%%while coll.next_entry%%
  %%if coll.entry.get_attrib(countme)=='true' '%%
  count++;
```

```

    %%endif%%
%%endwhile%%
document.writeln('The number of counted documents is '+count);
//-->
</SCRIPT>

```

The HTML source delivered to your browser will look like the following:

```

<SCRIPT LANGUAGE='JavaScript'>
<!--
var count=0;
    count++;
    count++;
    count++;
    ....
    ....
    ....
document.writeln('The number of counted documents is '+count);
//-->
</SCRIPT>

```

Every time the while loop in PLACE recognizes a child of the current collection where the attribute `countme` is set to `true` simply a line with the content `count++;` is written into the HTML document. If you look at the JavaScript code produced in this way it may be clear that it was not created by a human being.

6.2.2 Hyperwave and CGI

6.2.2.1 What is the CGI

This short introduction to the CGI (Common Gateway Interface) is based on a tutorial by *Randal L. Schwartz* from the book *Learning Perl* [31].

Many of the more interesting Web pages include some sort of entry form. You supply input to this form and click on a button or picture. This fires up a program at the Web server that examines your input and generates new output. Sometimes this program (commonly known as a CGI program) is just an interface to an existing database, messaging your input into something the database understands and messaging the database's output into something a Web browser can understand (usually HTML).

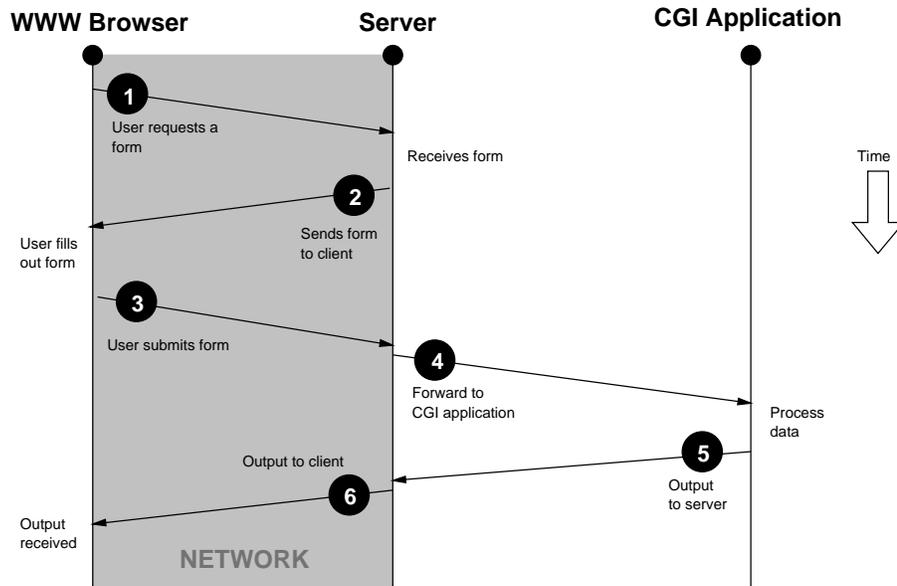


Figure 6.3: Form Interaction with CGI

Figure 6.3 shows the interaction between a HTML form and a CGI script. In step one, the user opens a URL with his Web browser. This URL specifies a Web server and a resource accessible through that server. This resource might be a HTML form which is then presented to the user. Each text-input field on the form has a name (given in the form’s HTML code) and an associated value, which is whatever the user types into the field. The form is bound to a CGI program via the HTML `<FORM>` tag. Submitting the form (in most cases this is done when clicking the *Submit* button in the form) means requesting the URL of the CGI program. The CGI is then responsible for passing the `name=value` pairs defined in the HTML form to the CGI program.

Depending on the `METHOD` attribute of the `<FORM>` tag the `name=value` pairs are passed in a different way to the CGI script. When using `METHOD=GET` a so called *query string* is formed which is tacked onto the end of the CGI program’s URL:

```
http://www.SOMEWHERE.org/cgi-bin/some_cgi_prog
?flavour=vanilla&size=double
```

`METHOD=POST` would pass all data over the standard input to the CGI script. Which one of the both methods is used depends of the amount of data to be transferred. Posting seems to be the appropriate method for a large amount of data.

When programming with the Perl module `CGI.pm` you don’t have to worry about this. The module recognizes automatically from where the input is to be

taken, but more on that later.

Finally, it should be mentioned, that CGI programs can work with any HTML document, not just forms. For example, you could write the HTML code

```
<A HREF=' 'http://www.SOMEWHERE.org/cgi-bin/fortune.cgi''>Receive your
fortune
</A>
```

and *fortune.cgi* could be a program that simply invokes the *fortune* program (on UNIX systems). In this case, there wouldn't be any argument supplied to the CGI program with the URL. Or the HTML document could give two links for the user to click on – one to receive fortune, and one to receive the current date. Both links could point to the same program, in one case with the argument **fortune** following the question mark in the URL, and in the other case with the argument **date**. The HTML links would look like this:

```
<A HREF=' 'http://www.SOMEWHERE.org/cgi-bin/fortune_or_date
?fortune' '>
<A HREF=' 'http://www.SOMEWHERE.org/cgi-bin/fortune_or_date
?date' '>
```

The CGI program (*fortune_or_date* in this case) would then see which of the two possible arguments it received and execute either the *fortune* or *date* program accordingly.

6.2.2.2 Perl and CGI.pm

The following sentences are taken from the preface of the book *Programming Perl* from Larry Wall, Tom Christiansen and Randal L. Schwartz [34]. This book is “the bible for Perl programmers” and it will be clear, that Perl is an excellent choice for programming CGI scripts. The amount of Perl resources on the World Wide Web³ [26] reflects the popularity of this language.

Initially designed as a glue language for the UNIX operating system (or any of its myriad variants), Perl also runs on numerous other systems, including MS-DOS, VMS, OS/2, Plan 9, Macintosh, and any variety of Windows you care to mention. It is one of the most portable programming languages today.[...]Web programmers are often delighted to discover that they can take their scripts from a Windows machine and run them unchanged on their UNIX server.

³A good entry point for Perl resources on the WWW is <http://www.perl.com>.

Perl is no longer just for text processing. It has grown into a sophisticated, general-purpose programming language with a rich software development environment complete with debuggers, profilers, cross-references, compilers, interpreters, libraries, syntax-directed , and all the rest of the trappings of a “real” programming language.[...]Perl is used by people who are desperate to analyze or convert lots of data quickly, whether you’re talking DNA sequences, Web pages, or pork belly futures.

Unlike a strictly interpreted language such as the shell, which compiles and executes a script one command at a time, Perl first compiles your whole program quickly into an intermediate format. Like any other compiler, it performs various optimizations, and gives you instant feedback on everything from syntax and semantic errors to library binding mishaps. Once Perl’s compiler frontend is happy with your program, it passes off the intermediate code to the interpreter to execute.

In most of older Perl CGI script you will find some lines of code with a similar look like these:

```
@pairs = split(/&/, $buffer);

foreach $pair (@pairs){
    ($name, $value) = split(/=/, $pair);

    $value =~ tr/+/ /;
    $value =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack('C',hex($1))/eg;
    $name =~ tr/+/ /;
    $name =~ s/%([a-fA-F0-9][a-fA-F0-9])/pack('C',hex($1))/eg;

    $FORM{$name} = $value;
}
```

Depending on the method (**GET** or **POST** see 6.2.2.1) the variable **buffer** is filled in a different way. What happens next, is the conversion from the *URL-encoded* script input into a human readable representation (in fact this is exactly the form-data the user entered before submitting data to the CGI script). The *URL-encoding* is necessary because when using the **GET** method, the data is appended to the script’s URL, and all characters in the data which would not be allowed in a URL are escaped (replaced by their hex value).

When using the Perl module *CGI.pm* this “manual conversion” of the script’s input data is no longer necessary. All is done there automatically. The programmer doesn’t have to care about where data is coming from (from STDIN or as argument of the command line).

The following lines of code demonstrate the use of CGI.pm; the programmer doesn't need to convert data himself:

```
...
use CGI qw(:standard);
...
$query = CGI->new();
$value = $query->param('name');
..
```

CGI.pm became a very powerful tool and it would be beyond the scope of this chapter to talk about all the features of the module. The best way to learn about all the possibilities of CGI.pm is its excellent manual page⁴ [3].

6.2.2.3 CGI Scripts and the Hyperwave Server

In the *Hyperwave Programmer's Guide*[11] a step-by-step instruction is given, how to deal with CGI scripts in the Hyperwave system.

In Hyperwave, CGI programs are not actually uploaded to the server, but rather they are put in a special directory meant for CGI programs. Hyperwave CGI objects must be created to reference the programs. These are objects of type "CGI" which contain only the name and location of the actual CGI program within their Path attribute. These objects have to be called by other (e.g. HTML forms) to start the CGI program.

In Hyperwave CGI programs reside outside the database in the special server directory `~hwsystem/dcserver/cgi`. This is the entry point for the server module `dcserver` which executes the CGI program. The first two path components of the Path attribute in the corresponding CGI object are interpreted as a relative path name beginning in the `~hwsystem/dcserver/cgi` directory.

After installing the CGI script in the Hyperwave server's filesystem, there are two ways to create the references in the server's database. One method is to use the Hyperwave Tool `hwinsdoc`. A description of `hwinsdoc` can be found in the appendix (see B.2).

The easier way for less experienced users or administrators is to create the CGI object with the WaveMaster interface using a Web browser:

⁴The CGI.pm manual page is available on many locations in the World Wide Web. The site of the CGI.pm author is http://stein.cshl.org/WWW/software/CGI/cgi_docs.html

1. Identify as a system user.
2. Click on the Edit button on top of your screen (in default layout).
3. Click on the “Insert document” icon in the toolbar and then click on the “CGI Object” tab.
4. In the form that appears you must insert the Path attribute, which consists of *directory/progname*, the collection you want to insert the CGI object into (enter its Name or GOid attribute here) and a title including a language.
5. Commit your entries by clicking on Insert CGI Object.

6.2.3 The Hyperwave API

The documents in the Hyperwave documentation dealing with the API are the *Hyperwave API Definition*⁵[14] and chapter two of the *Hyperwave Programmer's Guide*⁶[11]. While the *API Definition* is a language independent description of the API, the *Programmer's Guide* deals with the server side JavaScript implementation of the API.

The Hyperwave Information Server API is targeted at experienced users with programming skills who want to add their own set of functionalities to the HWIS. It provides access to the server's operation by providing a set of functions which cover the server's capabilities at a relatively high level. Additionally, it provides a set of specific HWIS data types along with their common access methods. (For example, the function "children", which is used to retrieve the descendants of a collection, returns a list of HWIS objects. The list as well as the HWIS objects and the access methods of both are defined by the API.)

Server side JavaScript in combination with PLACE is in most cases a substitution for the use of CGI script. While in Hyperwave version 2.6a and earlier CGI was often the only possibility to fulfill some tasks, the introduction of the Hyperwave API with version 4.0 changed the situation. Server side JavaScript is also a good alternative for the use of Perl in CGI scripting. When programming Hyperwave with Perl, one is forced to make calls to the *Hyperwave Tools* (in the appendix some of them are described; see Appendix B). This results in a low execution speed and a relatively high overhead caused by error checking. Server side JavaScript as CGI scripting language reduces this overhead through a better integration in the Hyperwave system.

⁵Online version: <http://www.hyperwave.de/apidef>

⁶See <http://www.hyperwave.de/program> for the online version.

For Hyperwave experienced programmers it is also very easy to learn the JavaScript concepts. The reason for this is, that the *object oriented* world of Hyperwave is mapped to *object based* properties of the JavaScript language.

6.2.3.1 The HWJS Command

The `hwjs` commandline tool is a program that interprets files that consists of pure JavaScript. With this utility it is possible to access a Hyperwave server with a script language [11].

This little demonstration script, taken from the *Hyperwave Programmer's Guide* shows the concepts behind `hwjs`:

```
// this is the file hello.js
writeln('hello world');
```

This script now can be invoked this way: `hwjs hello.js` and would produce the desired output on the terminal. Used as CGI script, the following version of `hello.js` would make sense:

```
#!/usr/local/bin/hwjs
writeln('hello world');
```

So the script is recognized in UNIX systems as to be executed with `hwjs` and no explicit call to the tool is necessary.

The following table just lists all object available with `hwjs`. For a full description of them and examples how to use them please consult the *Hyperwave Programmer's Guide*, chapter 2.2 [11].

Class/Object	Description
HW_API_Attribute	attribute
HW_API_Object	object
HW_API_ObjectArray	objectarray
HW_API_Reason	reason for error reporting
HW_API_Error	error
HW_API_Content	content representation
HW_API_Server	the communication class
KeyValue	key value pairs for http header
KeyValueField	field of keyvalue
HTMLJavaScriptParser	HTML parser
SendMail	mail facility
File	file access facility
OptionParser	commandline parsing tool
argv	the commandline object
environment	environment variables
write()	writes into the output document
writeln()	writes into the output document
writeError()	writes on stderr (debug only)
read()	reads from stdin
readln()	reads a line from stdin
getpass()	reads a hidden information (password) from the stty

6.2.3.2 JavaScript in the WaveMaster Templates

JavaScript is used in the PLACE templates to enhance the functionality of PLACE or to cover functionality which PLACE is not able to perform. The normal approach of developing an application with JavaScript in the templates is to process the request to see what should happen, then use some classes and objects to access the Hyperwave server and prepare the response to fulfill the request.

The following example from the *Hyperwave Programmer's Guide*, chapter 2.3 [11], demonstrates all possibilities how to embed JavaScript in the PLACE templates:

```
<!-- using the SERVER tag -->
<SERVER>
  //this is pure JavaScript
  function pref() {
    return 'HREF';
  }
}
```

```

function url() {
    return 'http://www.hyperwave.com';
}
write('hello and welcome');
</SERVER>

<!-- invocation in a tag -->
<A 'pref()='url()'>Go to Hyperwave</A>

<!-- a include -->
<SERVER SRC=''file://test.js''>

```

The `<SERVER>` tag works like the known `<SCRIPT>` tag. The difference is that the text contained within opening and closing tag is interpreted by the server before the document is delivered to the client. The example shows too how JavaScript can be embedded in HTML tag as attributes using backquotes.

It would be far beyond the scope of this section to describe all the Hyperwave objects and methods which can be used with JavaScript in WaveMaster templates. They are just listed here in tabular form. A detailed description is given in the *Hyperwave Programmer's Guide*, chapter 2.4 [11].

Class/Object	Description
HW_API_Attribute	attribute
HW_API_Object	object
HW_API_ObjectArray	objectarray
HW_API_Reason	reason for error reporting
HW_API_Error	error
HW_API_Content	content representation
HW_API_Server	the communication class
KeyValue	key value pairs for http header
KeyValueField	field of keyvalue
HTMLJavaScriptParser	HTML parser
SendMail	mail facility
File	file access facility
server	the current server object
request	the current request object
wavemaster	wavemaster information object
client	the client object
write()	writes into the output document
writeln()	writes into the output document
writeError()	writes on stderr (debug only)

6.3 A Hyperwave Programming Example

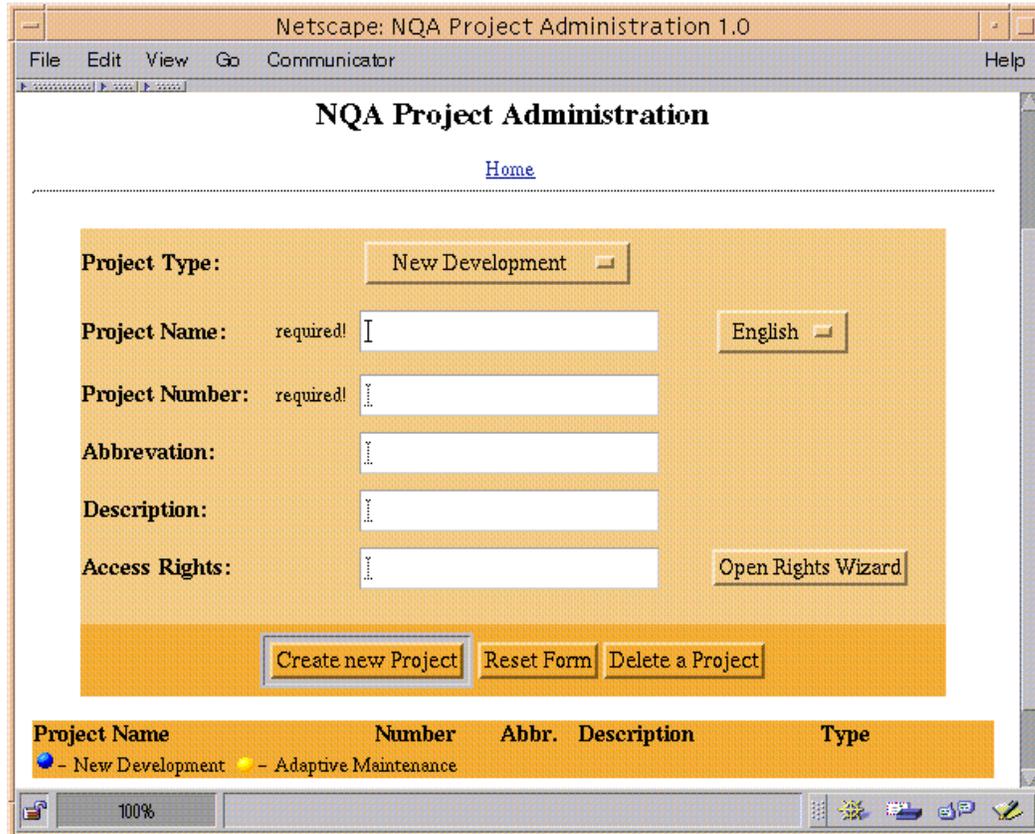


Figure 6.4: The Project Creation Dialogue

In this section I will present an example how Hyperwave programming can be used in a real world application. The code is taken from the NQA system. The goal is to create a simple interface for creating and deleting projects in the NQA system. All possible Hyperwave programming paradigms are used there in some way:

- PLACE in combination with client side JavaScript
- CGI programming using the Hyperwave commandline tools
- The server side JavaScript implementation of the Hyperwave API

I won't discuss every single PLACE-holder and every single line of code here. I just want to show how things can be put together to solve complicated tasks.

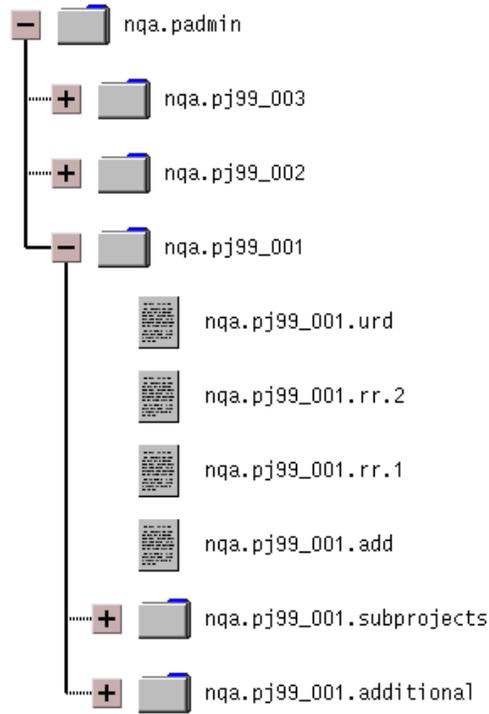


Figure 6.5: NQA Project Collection Structure

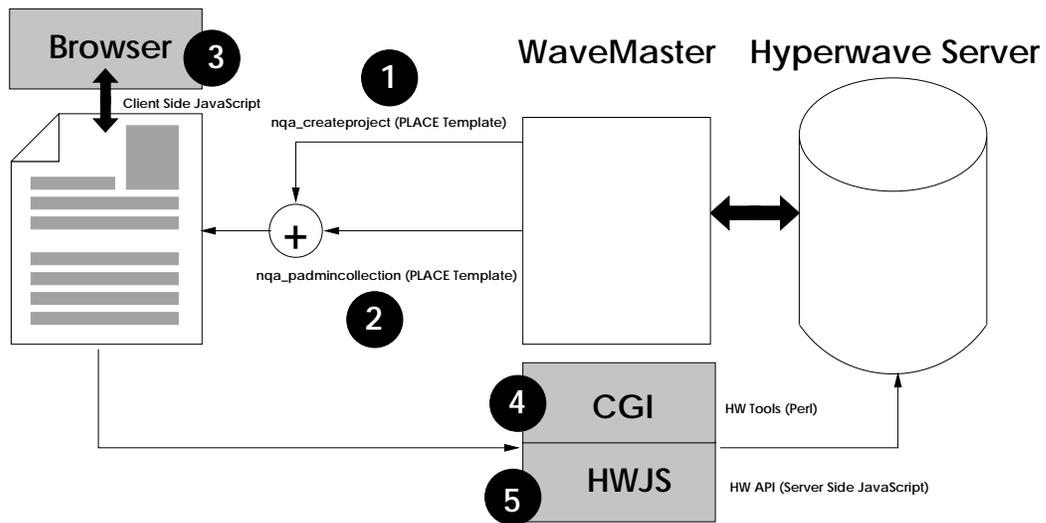


Figure 6.6: Example Overview

This example depends on a given Hyperwave collection structure. Figure 6.5 presents the structure of an NQA project and will help to understand the tasks performed in the example. The collection `nqa.padmin` is the base collection for all projects. It is created when the NQA system is installed. Every project collection is a child of `nqa.padmin`. The name attribute (in fact the URL) has the form `nqa.pnumber` where *pnumber* is the project number chosen by the systems's user when creating a new project.

Each project collection consists of exactly two subcollections

- `nqa.pnumber.additional` for eventually existing additional project data and
- `nqa.pnumber.subprojects` as parent collection for subprojects.

Each document or report belonging to the project is a direct successor of `nqa.padmin`. They are following the naming convention

$$\text{nqa.pnumber.doctype}[\text{.repnr}]$$

where *doctype* is mostly an abbreviation for the document or report (e.g. URD for User Requirements Document) and *repnr* an integer value to distinguish between several reports of the same type.

Figure 6.6 shows the interaction of the different concepts used during this example. The numbers in the figure will be referenced when discussing the different techniques in-depth in the following subsections.

6.3.1 When should the HTML form be displayed?

I won't describe the whole NQA system here, so some preconditions have to be met. A Hyperwave collection with the custom attribute `nqa.padmincollection` set to `true` already exists. When the URL of this collection is opened in some way with a Web browser, a form should be displayed where information for the project to create should be entered. Also a list of already created project should be shown. The resulting form should look like the one in figure 6.4.

The standard Hyperwave behaviour when opening a collection would be that the objects in the collection are displayed in some way, depending on the collection's type. If we recognize that the current object is a collection and the attribute `nqa.padmincollection` is set to `true` we have to take over control, otherwise the standard behaviour is required.

```

    %%# the object is the NQA Project Administration Collection %%
    %%#------%%
1.  %%if object.get_attrib(nqa_padmincollection)== "true"%%
2.    %%nqa_padmincollection%%
3.  %%endif%%

```

These lines have to be put at the beginning of the Hyperwave body macro. It has to be placed in a way that no other actions can be performed in the rest of the body template. Line 1 simply checks if the current object is our *Project Administration Collection*, if so, the macro `nqa_padmincollection` is opened.

The macro has to be defined in the file *hwincludes.html*:

```
%%include "v4.0/include/nqa_padmincollection.html"%%
```

Here the assumption was made that `nqa_padmincollection.html` is really defined in the file *nqa_padmincollection.html*.

The following is the complete listing of the `nqa_padmincollection` macro:

```

%%# NQA Project Administration Macro %%
%%# Martin Melcher 1998 %%

1.  %%macro nqa_padmincollection%%
2.  <TABLE WIDTH=600 BORDER=0><TD>
3.  <H2 ALIGN="center">%%ge:NQA Projektverwaltung,
                                :NQA Project Administration%%</H2>
    %%# goto's %%
4.  <CENTER>
5.  <FONT SIZE=-1><A HREF="index.htm">Home</A></FONT>
6.  %%if parents.count==1 && parents.entry.get_attrib(nqa_project)== "true"%%
7.    <FONT SIZE=-1> | <A HREF="%%parents.entry.get_attrib(Name)%%">
        %%ge:Projekt&uuml;bersicht, :Project Sheet%%</A></FONT>
8.  %%endif%%
9.  </CENTER>
10. <HR>

    %%#call form for creating a new project%%
11. %%if session.user.is_system ||
    (parents.count==1 && parents.entry.get_attrib(nqa_project)== "true")%%
12.  %%if object.is_editable%%
13.    %%nqa_createproject%%

```

```

14. %%endif%%
15. %%else%%
16. <CENTER><FONT FACE="arial" COLOR="#ff0000"><P>
17. %%if session.language=="ge"%%
18.     Zum Anlegen oder L&ouml;schen von Projekten
        identifizieren Sie sich bitte als Systemuser.
19. %%else%%
20.     In order to create or delete projects please
        identify as system user.
21. %%endif%%
22. <P></FONT></CENTER>
23. %%endif%%

24. <TABLE BORDER=0 WIDTH=600 %%bgcolor_light%% CELLSPACING=0>
25. <TR %%bgcolor_dark%%><TH ALIGN="left">
        %%ge:Projektname,   :Project Name%%</TH>
26. <TH ALIGN="left">%%ge:Nummer,   :Number%%</TH>
27. <TH ALIGN="left">%%ge:Abk.,   :Abbr.%%</TH>
28. <TH ALIGN="left" WIDTH="10%">%%ge:Beschreibung,   :Description%%</TD>
29. <TH ALIGN="center">%%ge:Typ,   :Type%%</TD>
30. %%while coll.next_entry%%
31.     %%if coll.entry.get_attrib(nqa_project)=="true"%%
32.         <TR>
33.             <TD VALIGN="top">
34.                 <IMG SRC="%%coll.entry.icon%%" BORDER=0 ALIGN="absbottom">
                    &nbsp;<A HREF="%%coll.entry.uri%%">%%coll.entry.title%%</A></TD>
35.                 <TD VALIGN="top">%%coll.entry.get_attrib(nqa_projectnr)%%</TD>
36.                 <TD VALIGN="top">%%if coll.entry.get_attrib(nqa_projectabbr)!="%%
                    %%coll.entry.get_attrib(nqa_projectabbr)%%
                    %%else%%&nbsp;&nbsp;&nbsp;%%endif%%</TD>
37.                 <TD VALIGN="top">%%if coll.entry.get_attrib(Description)!="%%
                    <FONT SIZE=-1>%%coll.entry.get_attrib(Description)%%
                    </FONT>%%else%%&nbsp;&nbsp;&nbsp;%%endif%%</TD>
38.                 <TD ALIGN="center" VALIGN="top">
                    %%if coll.entry.get_attrib(nqa_projecttype)=="n"%%
                    <IMG SRC="%%gate.object(v4.0/graphics/ball_blue.gif)%%" BORDER=0>
                    %%endif%%%%if coll.entry.get_attrib(nqa_projecttype)=="a"%%
                    <IMG SRC="%%gate.object(v4.0/graphics/ball_yellow.gif)%%" BORDER=0>%%
                    endif%%</TD>
39.             </TR>
40.     %%endif%%
41. %%endwhile%%

42. <TR %%bgcolor_dark%%><TD COLSPAN=5>
43. <IMG SRC="%%gate.object(v4.0/graphics/ball_blue.gif)%%"

```

```

    BORDER=0 ALIGN="absbottom"><FONT SIZE=-1>-
    %%ge:Neuentwicklung, :New Development%%</FONT>&nbsp;
44. <IMG SRC="%%gate.object(v4.0/graphics/ball_yellow.gif)%%"
    BORDER=0 ALIGN="absbottom"><FONT SIZE=-1>-
    %%ge:Adaptive Wartung, :Adaptive Maintenance%%</FONT>
45. </TD>
46. </TABLE>

47. <HR>

48. </TD></TABLE>

49. %%endmacro%%

```

This macro is divided into four main parts:

Line 4–10: A link back to the NQA index page is established. If the collection we are handling is the **Project Administration Table** of a subproject, a link to the index page of the current project is created.

Line 11–23: Only Hyperwave system users (users who belong to the group system) are allowed to create a top level project. If the current collection is a **Project Administration Collection** for a subproject, also non-system users with write permission are allowed to create projects. This is checked in line 11. If all the conditions mentioned before are fulfilled, a macro `nqa_createproject` is called.

Line 24–41: Here all existing projects which can be accessed by the currently logged in user are listed.

Line 42–46: On the bottom of the page, just under the listed projects, a description of the symbols used in the project list is drawn.

The essential part of this macro is line 11–23. I will describe later what happens in the macro `nqa_createproject`. In this macro the Hyperwave multilinguality feature is used in both possible forms:

The `%%if session.language=='ge' '%%...%%else%%...%%endif%%` and the much more elegant form `%%ge:..., :...%%`.

The lines 30–41 give a good example how to use the PLACE while loop to inspect the contents of a collection.

6.3.2 Getting User Input and Calling the CGI Script

In line 13 in the macro `nqa_padmincollection` the macro `nqa_createproject` is called. This macro displays a HTML form for creating a project. The user input is checked for correctness and is passed to a CGI script which is finally responsible for creating a new project collection on the server (see points 1 and 2 in figure 6.6).

```

    %%# creates a new project in the actual context %%
    %%# Martin Melcher 1998 %%

1.  %%macro nqa_createproject%%

2.  <SCRIPT LANGUAGE="Javascript">
    <!--
    // sets the Name attribute according to the Projectnr attribute
    // Name = nqa.Projectnr
3.  function setName(f) {
4.    for (var i=0; i<f.length; i++) {
5.      if (f.elements[i].name == "Name") var n = i;
6.      if (f.elements[i].name == "Projectnr") var pnr = f.elements[i].value;
7.    }
8.    f.elements[n].value = "nqa." + pnr;
9.  }

    // checks if the field is really blanc
10. function isBlanc(s) {
11.   for (var i=0; i<s.length; i++) {
12.     var c=s.charAt(i);
13.     if ((c!=' ') && (c!='\n') && (c!='\t')) return false;
14.   }
15.   return true;
16. }

    // check if " occurs
17. function checkQuotation(s) {
18.   if (s.indexOf('"') != -1) return false;
19.   return true;
20. }

    //check if non ASCII characters occur
21. function checkNonASCII(s) {
22.   if (escape(s).indexOf('%') != -1) return false;
23.   return true;
24. }

```

```
    // verifies that all non optional fields are filled and
    // the syntax is correct
25. function verify(f) {

    // Name,Projectnumber and Projectuser are required
26.   if ((f.Title.value==null) || (f.Title.value=="") ||
        isBlanc(f.Title.value)) {
27.     alert("%%ge:Sie mssen einen Projektnamen angeben,
            :You have to specify a project name!%%");
28.     return false;
29.   }
30.   if ((f.Projectnr.value==null) || (f.Projectnr.value=="") ||
        isBlanc(f.Projectnr.value)){
31.     alert("%%ge:Sie mssen eine Projektnummer angeben,
            :You have to specify a project number!%%");
32.     return false;
33.   }

    // no spaces etc. in Projectnumber
34.   if (! checkNonASCII(f.Projectnr.value)) {
35.     alert("%%ge:Bitte verwenden Sie nur Zeichen aus den Bereichen
            [A-Z][a-z][1-9][_ -] in der Projektnummer.,
            :Please use only characters wich are in [A-Z][a-z][1-9][_ -]
            for the Project Number.%%");
36.     return false;
37.   }

    // " must never occur
38.   for (var i=0; i < f.length; i++) {
39.     var e = f.elements[i];
40.     if ((e.value != null) && (! checkQuotation(f.elements[i].value))) {
41.       alert("%%ge:Bitte verwenden Sie keine
            Anführungszeichen (\") in den Attributen.,
            :Please do not use Quotations (\") in any of
            the attributes.%%");
42.       return false;
43.     }
44.   }

45.   return true;
46. }
    //-->
47. </SCRIPT>
```

```

48. <FORM ACTION="nqa_createproject.pl" METHOD="post"
    ONSUBMIT="setName(this); return verify(this);">

    %%# non editable fields %%
49. <INPUT TYPE="hidden" NAME="Parent" VALUE="%%object.get_attrib(Name)%%">
    %%#set by Javascript%%
50. <INPUT TYPE="hidden" NAME="Name" VALUE="">
51. <INPUT TYPE="hidden" NAME="User" VALUE="%%session.username%%">

    %%# editable fields %%
52. <TABLE WIDTH=540 BORDER=0 CELLSPACING=0 %%bgcolor_light%% ALIGN="center">

53. <TR><TD><B>%%ge:Projekttyp:, :Project Type:%%</B></TD>
54. <TD>&nbsp;</TD>
55. <TD><SELECT NAME="Projecttype">
56. <OPTION VALUE="n" SELECTED>%%ge:Neuentwicklung,
        :New Development%%</OPTION>
57. <OPTION VALUE="a">%%ge:Adaptive Wartung,
        :Adaptive Maintenance%%</OPTION>
58. </SELECT></TD>
59. <TD>&nbsp;</TD>

60. <TR><TD><B>%%ge:Projektname:, :Project Name:%%</B></TD>
61. <TD><FONT SIZE=-1>%%ge:erforderlich!, :required!%%</FONT></TD>
62. <TD><INPUT TYPE=INPUT NAME="Title" VALUE="" SIZE=25></TD>
63. <TD><SELECT NAME="Language">
64. <OPTION VALUE="en" %%if session.language!="ge"%%
    SELECTED%%endif%%>%%ge:Englisch,:English%%</OPTION>
65. <OPTION VALUE="ge" %%if session.language=="ge"%%
    SELECTED%%endif%%>%%ge:Deutsch,:German%%</OPTION>
66. </SELECT></TD>

67. <TR><TD><B>%%ge:Projektnummer:, :Project Number:%%</B></TD>
68. <TD><FONT SIZE=-1>%%ge:erforderlich!, :required!%%</FONT></TD>
69. <TD><INPUT TYPE=INPUT NAME="Projectnr" VALUE="" SIZE=25></TD>
70. <TD>&nbsp;</TD>

71. <TR><TD><B>%%ge:Abk&uuml;rzung:, :Abbreviation:%%</B></TD>
72. <TD>&nbsp;</TD>
73. <TD><INPUT TYPE=INPUT NAME="Projectabbr" VALUE="" SIZE=25></TD>
74. <TD>&nbsp;</TD>

75. <TR><TD><B>%%ge:Beschreibung:, :Description:%%</B></TD>
76. <TD>&nbsp;</TD>
77. <TD><INPUT TYPE="TEXT" NAME="Description" VALUE="" SIZE="25"></TD>

```


The Input field named “Name” is of the type “hidden” so this value is not entered by the user (see line 50). The JavaScript function `setName` is responsible for passing a value to this input field. `setName` is defined in the lines 3–9. What happens here is, that the value from the field named “Projectnr” is combined with a fixed string “nqa.” and passed to the value of the hidden input field “Name”. So all projects in the NQA system will have a Hyperwave name attribute of the form “nqa.number” where number is a user defined string.

The following table lists all the data finally passed to the CGI script `nqa_createproject.pl`.

Name	Input Type	Value	Remarks
Parent	hidden	Name attribute of the current collection	
Name	hidden	“nqa.number” where number has the value of the input field <i>Projectnr</i>	Set by JavaScript
User	hidden	Username of the currently logged in Hyperwave user	
Projecttype	select	The possible values depend on the organisation where the NQA system is used	
Title	input	The Hyperwave title attribute for the new project	required
Language	select	A language for the title attribute	
Projectnr	input	A unique number for the new project	required
Projectabbr	input	An abbreviation for the new project	This field is never used and is there just because of historical reasons
Description	text	A free form string describing the project	
Rights	text	The Hyperwave access rights for the new project	This field controls who will be finally able to access or modify the content of the project.

Notice the combination of PLACE and JavaScript in the first section of the macro. PLACE is used there to bring the multilingual features of Hyperwave into JavaScript.

One important strategy used here is, to check the syntactic correctness of the user input with JavaScript instead of handling possible errors in the CGI script. When the user makes an invalid input, a warning messages indicates the error and the input can be corrected without much overhead. CGI programming is always very critical as far as the performance is concerned. All CGI transactions should be as short as possible in order to reduce the servers load. If we assume that the data passed to the CGI script is correct, no error detection is necessary in the script itself what results in a gain of execution speed and reduces the amount of code to be written.

6.3.3 Doing the Work: The CGI Script

The CGI script finally is the place, where physically changes are made to the server database (see point 4 in figure 6.6). In this case, the *hwinscoll* Hyperwave commandline tool is used to create the project collection and a subcollection for subprojects. This example is also a good demonstration for the changes in the Hyperwave programming schemes from the Hyperwave Information Server version 2.6 to 4.0. The whole NQA system was designed and programmed with the possibilities and restrictions of the HWIS 2.6a in mind. The collection for additional documents as subcollection of a project collection was added after HWIS version 4.0 appeared. To make use of the comfortable new features of the new server API, this collection is no longer created using the *hwinscoll* command (see line 18).

```
1.  #! /usr/local/bin/perl
```

```
#####
# nqa_createproject.pl Martin Melcher 1998
#####
# This script is used in the NQA Hyperwave system to create a
# new project
#####

#####
# include required modules
#####
```

```
2.  use CGI qw(:standard);
```

```

3. use NQAHW;

#=====
# identification option
#=====
4. $ident = "$NQAHW::IDENT";

#=====
# get CGI parameters
#-----
# Parent: collection to insert project
# Name: hw Name attribue
# Title: hw Title attribute
# Language: hw Title Language
# User: hw owner of the project
# Description: hw Description attribute
# Projecttype: NQA Projecttype - n New Development
#                               - a Adaptive Maintenance
# Projectnr: NQA Project Number
# Projectabbr: NQA Project Abbreviation
# Rights: NQA Project Access Ritghts
#=====
5. $request = CGI->new();
6. $Parent = $request->param("Parent");
7. $Name = $request->param("Name");
8. $Title = $request->param("Title");
9. $Language = $request->param("Language");
10. $User = $request->param("User");
11. $Description = $request->param("Description");
12. $Projecttype = $request->param("Projecttype");
13. $Projectnr = $request->param("Projectnr");
14. $Projectabbr = $request->param("Projectabbr");
15. $Rights = $request->param("Rights");

#=====
# create the project collection
#-----
# hwinscoll using
#           Parent      = $Parent
#           Name        = $Name
#           Title       = $Language:$Title
#           Description  = $Description
#           Rights      = $Rights
# the following attributes are set a insert time

```

```

#           nqa_project      = true
#           nqa_projecttype = $Projecttype
#           nqa_projectnr   = $Projectnr
#           nqa_projectabbr = $Projectabbr
#=====
16. NQAHW::exception("hwinscoll_project", $Language, system "hwinscoll
    -ident \"\$ident\" -parent $Parent
    -name $Name -formatted
    -title \"Title=$Language:$Title\\nqa_project=true\\
            nqa_projectnr=$Projectnr\\
            nqa_projecttype=$Projecttype\\
            nqa_projectabbr=$Projectabbr\"
    -description \"\$Description\"
    -rights \"\$Rights\" > $$tmp");

#=====
# create the subproject collection
#-----
# hwinscoll using Owner      = $User
#           Parent          = $Name
#           Name            = $Name.subprojects
#           Title           = en:Subprojects
#                           ge:Subprojekte
#           Sequence        = 5
#           Rights          = $Rights
# the following attributes are set a insert time
#           nqa_padmincollection = true
#           nqa_parentproject = $Name $Title
#
#=====
17. NQAHW::exception("hwinscoll_subproject", $Language, system "hwinscoll
    -ident \"\$ident\" -parent $Name -name $Name.subprojects
    -formatted
    -title \"Title=en:Subprojects\\Title=ge:Subprojekte\\
            nqa_padmincollection=true\\
            nqa_parentproject=$Name $Title\\
            Sequence=5\"
    -rights \"\$Rights\" > $$tmp");

# Additional Documents Collection
18. system "hwjs nqa_createadditional.js $Name \"\$Rights\" $ident";

```

```

=====
# send a HTTP Redirection Header to the client that the page, the
# script was called from will be reloaded
=====
19. print "Location:".ENV{"HTTP_REFERER"}."\r\n";
20. print "Content-type: text/html\r\n\r\n";
    # browser doesn't understand redirection
21. print "<HTML><HEAD><TITLE>NQA Redirection</TITLE></HEAD><BODY>";
22. print "<A HREF=\"".ENV{"HTTP_REFERER"}."\">
        Return to previous accessed document.</A>";
23. print "</BODY></HTML>";

    # delete tmp file
24. unlink("$$1.tmp");
25. unlink("$$1.tmp");

    # delete log file if exists from deleted projects with the same name
26. unlink("$Projectnr.log");

```

The documentation in the code will help to understand what is going on there. The script might be separated in three parts:

Line 1–15: Here all the CGI specific data handling is done. This task is simplified using the CGI.pm Perl module. Line 4 is essential for the understanding of the code. the Perl variable `ident` is a string of the form “username password”. This string is used later for identification when starting *hwinscoll* or the JavaScript API interpreter *hwjs*.

Line 16–18: This is the dirty part of the script, where the control is passed to *hwinscoll* and *hwjs* (see point 5 in figure 6.6). It is very hard to detect errors in this phase of the script execution. That is the reason, why *hwinscoll* is started there in a kind of debugging environment. The Perl function `exception` evaluates the return value of the *hwinscoll* command and terminates the execution of the CGI script, sending an error message to the client, if necessary. What is done exactly in line 18 will be explained in the next subsection.

Line 19–26: When all the transactions were successful, the client has to be forced to update his image of the current collection (finally there should be a new project which has to be displayed immediately). The lines sent to the client over the CGI represent a HTTP redirection header. The client is forced to reload his actual URL. The Hyperwave server reacts in sending back the updated project administration collection.

6.3.4 Working with Server Side JavaScript

This script is started as input to the Hyperwave JavaScript parser (*hwjs*).

```
system "hwjs nqa_createadditional.js $Name \"\$Rights\" $ident";
```

The CGI script *nqa_createproject.pl* is responsible for creating the parameters **Name**, **Rights** and **ident**. The meaning of the parameters is explained in the code's documentation. The Perl variable **ident** is of the form "username password" so two distinct strings will be passed to the script.

```
// nqa_createadditional.js V1.0 Martin Melcher 1998
//-----

// get values from commandline
// expecting:
//          1. Name of Parent Collection
//          2. Rights Attribute
//          3. Username for Identification
//          4. Password for Identification

1. var parent = argv[1];
2. var rights = argv[2];
3. var user   = argv[3];
4. var passwd = argv[4];

// get a server object
5. var nqaserver = new HW_API_Server("localhost");
6. nqaserver.identify({username:user,password:passwd});

// inserts a collection object cobject as child of cparent
7. function insertCollection(cobject,cparent) {
8.   var parobj = new HW_API_Object();
9.   parobj.Parent = cparent;
10.  nqaserver.insert({"object":cobject,"parameters":parobj});
11. }

// create the Additional Documents Collection
12. var cadd = new HW_API_Object();
13. cadd.Type = "Document";
14. cadd.DocumentType = "collection";
15. cadd.insert(new HW_API_Attribute("Title",
    new Array("en:Additional Documents",
```

```
                                "ge:Zusaetzliche Dokumente"))));  
16. cadd.Rights = rights;  
17. cadd.nqa_additional = "true";  
18. cadd.Name = parent+".additional";  
  
19. insertCollection(cadd,parent);
```

Line 1–4: The input parameters are parsed form the `argv` array.

Line 5–6: An object of the type *HW_API_Server* is created. This object represents a physical connection to the server running on *localhost*.

Line 7–11: A JavaScript function for creating collections is defined. It might be an overhead to define a function there, but in future versions of the NQA system more than just one collection might be created.

Line 12–19: The properties of a *HW_API_Object* are filled with the necessary attributes for a new collection and the function defined in line 7–11 is called.

Chapter 7

Conclusion

The NQA system was designed as an enhancement to the existing Hyperwave functionality. The result of this design was a strong integration into Hyperwave. So it is not amazing that for every new Hyperwave release NQA would have to be modified to reflect the new features in the server. Through the relatively short lifetime of a Hyperwave version and the bulk of new features in every new release, maintaining and enhancing a Hyperwave application is a challenging task for the developer. I am sure, that the user too is overtaxed with a user interface which changes from release to release and a set of new tools while the overall performance stagnates.

NQA is configured to meet the customer's requirements. In fact the configuration is done on a very low level. In some cases a modification of the source code is necessary. This makes a configuration management very hard. In subsection 4.2.1 the principles behind *Development by Configuration* are explained and a modification of source code doesn't fit into this concept. To reach the goals of *Development by Configuration* the separation of function and data in the NQA system would have to be forced.

NQA HW was criticized to make too restricted use of the Hyperwave features. The reason for this is, that NQA Hyperwave is the successor of a tool called NQAHT (NQA Hypertext) which was the little brother of NQAHW as far as functionality is concerned. NQAHT was a pure CGI application for standard HTTP Web servers. So the requirements for NQAHW just touched the Hyperwave possibilities.

7.1 The Future of NQA Hyperwave

A serious problem for Hyperwave developers is the fact, that all the PLACE templates and program codes for their system are simple ASCII files. This allows

the hacking and “self-modification” of the system by the customer. In the NQA project first steps were made into a direction, where platform binaries are used as CGI scripts. This allows the protection of the system with a licensing scheme similar to the one used for the Hyperwave server (the NQA installation would then be bound to a single Hyperwave server identified by its IP-address).

At the moment, NQA could be characterized as “*web based, project oriented document management system with restricted workflow functions*”. ISCN plans to make improvements in direction workflow management. The term describing NQA 2.0 should be “*web based workflow management tool*” from ISCN’s point of view.

With the product strategy presented in subsection 4.2.2 one problem arises: The developer has to stay in direct contact with the customer to enable updates and maintenance of the system. As long as the number of customers is restricted this problem can be managed. Especially for the NQA system I would propose the development or adaptation of a commercially available *web based update and maintenance tool*.

This idea is shown in figure 7.1. On a central update and maintenance server of the distributing company, in this case ISCN, a section for each customer exists. In this section the always newest customer specific NQA configuration is hold. The installed NQA system uses middleware components to retrieve updates dynamically and guarantees that all modifications to the system are delivered to the customer.

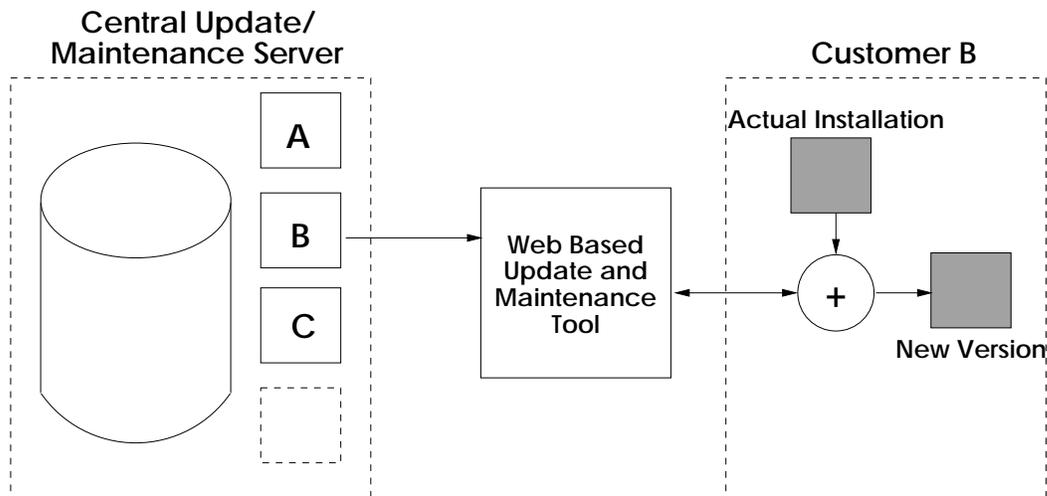


Figure 7.1: NQA Update/Maintenance Strategy

Appendix A

A Complete List of Placeholders

This list of placeholders is a part of the *Hyperwave Reference Guide*¹ [13][17].

Placeholders which have to do with actions

Action variables

action This is a variable which holds a value according to the action which was most recently called, e.g. its value is `action.identify` if the identify action `action.call.identify` was called.

action.parameter It is possible to submit a parameter when calling a predefined action. After such an action is called, `action.parameter` holds the value of this parameter.

Actions

All of these placeholders perform a certain function when they are accessed as a link.

action.call(ActionName) Calls a self-defined action `ActionName`.

action.recall(ActionName) Calls a self-defined action `ActionName` with sequence and query information of current object supplied.

action.call.changepassword Calls action to change your password.

action.call.delete.object This action is used to delete objects from collections.

action.call.edit.attributes This action is used to modify the attributes of an object on the server according to changes submitted in a form.

action.call.edit.text This action is used to modify a text on the server according to changes made by the user.

action.call.edit.upload This action is used to replace documents on the server.

action.call.extended.search Calls the extended search action.

action.call.help(topic) Calls action to get to help page about a specific topic.

action.call.home Calls action to go to the currently logged in user's home collection.

action.call.identify Calls the identification action.

action.call.insert.attributes This action is used when creating objects which consist only of attributes, such as collections and remote objects.

action.call.insert.link This action is used to insert links.

action.call.insert.text This action is used to insert a new text into the server.

action.call.insert.upload This action is used to insert new documents (files) into the server.

action.call.lock.object This action is used to lock objects.

action.call.movecopy.object This action is used to move or copy objects from one collection to another.

action.call.query An action which is used to submit information in the query instead of the parameter part of an URL.

action.call.search Calls the search action.

¹The *List of Placeholders* is available online. An actual version can be found under <http://www.hyperwave.de/ref/listofplace>

action.call.setlanguage Calls an action which sets the language for a session. This is used in conjunction with `action.setsession.form.language`.

action.call.setvar Calls action to set self-defined session variables.

action.call.unlock.object This action is used to unlock objects.

Constants

The placeholders below are constants which are the possible values of the variable `action`.

action.changepassword The value of action if the change password action was called.

action.delete.object The value of action if the delete action was called.

action.edit.attributes The value of action if the edit attributes action was called.

action.edit.text The value of action if the edit text action was called.

action.edit.upload The value of action if the edit upload action was called.

action.extended.search The value of action if the extended search action was called.

action.framecontent The value action is set to if a document is being retrieved from within a frame.

action.help(topic) The value of action if the help action was called.

action.identify The value of action if the identification action was called.

action.insert.attributes The value of action if the insert attributes action was called.

action.insert.link The value of action if the insert link action was called.

action.insert.text The value of action if the insert text action was called.

action.insert.upload The value of action if the insert upload action was called.

action.lock.object The value of action if the lock object action was called.

action.movecopy.object The value of action if the move/copy action was called.

action.noaction The value of action if no action was called.

action.search The value of action if the search action was called.

action.setlanguage The value of action if the setlanguage action was called.

action.unlock.object The value of action if the unlock object action was called.

Actions used in forms

action.setsession.form.language If you are using `action.call.setlanguage`, this is used as the NAME attribute when submitting the language prefix in a form. It is the same as `session.form.language`.

Placeholders for information about annotations to the current object

annotations.count Number of annotations to the current document

annotations.entry.call(ActionName) Call an action `ActionName` for the annotation the annotation pointer is pointing to.

annotations.entry.get_attrib(AttributeName) Gets the attribute of type `AttributeName` for the annotation the annotation pointer is pointing to.

annotations.entry.icon Icon to be shown for the object type of the annotation the annotation pointer is pointing to.

annotations.entry.title Title of the annotation the annotation pointer is pointing to.

annotations.entry.uri URI of the annotation the annotation pointer is pointing to.

annotations.next_entry This Boolean placeholder moves annotation pointer to the next entry. Returns true if there is a next entry, otherwise false.

Placeholders for information about the client connecting to WaveMaster

client.agent.architecture The architecture which the client connecting to WaveMaster is running on. Possible values are "Unix", "Windows", "Macintosh", "OS/2", etc.

client.agent.name The name of the client connecting to WaveMaster. Possible values are "Netscape", "MSIE", etc.

client.agent.version.major The major version number of the client connecting to WaveMaster.

client.agent.version.minor The minor version number of the client connecting to WaveMaster.

client.hostname Host name of the client connecting to WaveMaster (the user's host or a proxy)

client.ipadr IP-address of the client connecting to WaveMaster (the user's host or a proxy)

client.referrer The URL of the server where the most recently clicked-on link is.

Placeholders for information about the current cluster.

cluster.count Number of documents in the current cluster.

cluster.entry.call(ActionName) Calls an action ActionName for the cluster entry the cluster pointer is pointing to.

cluster.entry.content Content of the cluster entry the cluster pointer is pointing to, if it is a text.

cluster.entry.get_attr(AttributeName) Gets the attribute of type AttributeName for the cluster entry the cluster pointer is pointing to.

cluster.entry.icon Icon to be shown for the object type of the cluster entry the cluster pointer is pointing to.

cluster.entry.imagemap Imagemap for the cluster entry the cluster pointer is pointing to, if it is an image with imagemap.

cluster.entry.imagemap.usemap Usemap text for the cluster entry the cluster pointer is pointing to, if it is an image with imagemap.

cluster.entry.size The exact size of the cluster entry in bytes.

cluster.entry.size.approximate The approximate size of the cluster entry in bytes, kbytes or Mbytes depending on the size of the object.

cluster.entry.title Title of the cluster entry the cluster pointer is pointing to.

cluster.entry.uri URI of the cluster entry the cluster pointer is pointing to.

cluster.next_entry This Boolean placeholder moves the cluster pointer to the next entry. Returns true if there is a next entry, otherwise false.

Placeholders for information about the current collection.

coll.count Number of documents in the current collection.

coll.entry.call(ActionName) Calls an action ActionName for the collection entry the collection pointer is pointing to.

coll.entry.get_attr(AttributeName) Gets the attribute of type AttributeName for the collection entry the collection pointer is pointing to.

coll.entry.icon Icon to be shown for the object type of the collection entry the collection pointer is pointing to.

coll.entry.query Query of the collection entry the collection pointer is pointing to. (That is the characters after '?' in the URI)

coll.entry.reldpath Path with which the collection entry the collection pointer is pointing to can be accessed. It is the collection's name or GOid, if there is no name.

coll.entry.size The exact size of the collection entry in bytes.

coll.entry.size.approximate The approximate size of the collection entry in bytes, kbytes or Mbytes depending on the size of the object.

coll.entry.title Title of the collection entry the collection pointer is pointing to.

coll.entry.uri URI of the collection entry the collection pointer is pointing to.

coll.next_entry This Boolean placeholder moves the collection pointer to the next entry. Returns true if there is a next entry, otherwise false.

Placeholders for information about the collection head of the current collection

collhead.annotations.count The number of annotations to the collection head.

collhead.annotations.entry.call(ActionName) Calls an action ActionName for the annotation that the next entry pointer is pointing to.

collhead.annotations.entry.get_attr(AttributeName) Gets the attribute of type AttributeName for the annotation that the next entry pointer is pointing to.

collhead.annotations.entry.icon The icon of the annotation that the next entry pointer is pointing to.

collhead.annotations.entry.title The title of the annotation that the next entry pointer is pointing to.

collhead.annotations.entry.uri The URI of the annotation that the next entry pointer is pointing to.

collhead.annotations.next_entry This Boolean placeholder is true if there is a next entry in the list of annotations to the collection head.

collhead.call(ActionName) Calls an action ActionName for the collection head.

collhead.content Content of the collection head, if it is a text document

collhead.get_attr(AttributeName) Gets the value of the attribute of type AttributeName for the collection head.

collhead.icon Icon to be shown for the object type of collection head.

collhead.imagemap Imagemap for the collection head, if it is an image with imagemap.

collhead.imagemap.usemap Usemap text for the collection head, if it is an image with imagemap.

collhead.is_editable This Boolean placeholder is true if the collection head is editable.

collhead.title Title of the collection head.

collhead.uri URI of the collection head.

Placeholders related to the delete function

delete.object.form This placeholder is used as the NAME attribute in forms when submitting the GOids of the objects to be deleted.

delete.object.form.parent This placeholder is used as the NAME attribute in forms when submitting the GOid of the collection or cluster that objects are to be moved from when using the delete action (action.call.delete.object).

delete.object.persists A Boolean placeholder which is true if an object is still on the server after a delete operation.

Placeholders that have to do with editing

edit.attributes.form.key This placeholder is used as the NAME attribute in forms to submit an attribute type when adding a new attribute while editing attributes.

edit.attributes.form.newpassword This placeholder is used as the NAME attribute in forms to submit a password when editing a user record.

edit.attributes.form.newpasswordverify This placeholder is used as the NAME attribute in forms to submit a password verification when editing a user record.

edit.attributes.form.object This is used to submit one or more users to be removed from or added to a group.

edit.attributes.form.removekey This is used as the NAME attribute in forms to submit the attribute type to be removed when e.g. removing multiple users from multiple groups.

edit.attributes.form.removevalue This is used as the NAME attribute in forms to submit the attribute value to be removed when e.g. removing multiple users from multiple groups.

edit.attributes.form.value This placeholder is used as the NAME attribute in forms to submit an attribute value when adding a new attribute.

edit.text.form.content This placeholder is used as the TEXTAREA NAME attribute in forms to submit text which has been edited.

edit.upload.form This is used as the NAME attribute in forms when uploading a file as value, e.g. in the replace document function.

Placeholders related to errors

error This placeholder holds an error name if an error occurred and is empty otherwise.

error.is.special This Boolean placeholder is true if the error is actually a list of errors.

error.is.warning A Boolean placeholder which is true if an error is a warning.

error.special.entry.is.error This Boolean placeholder is true if the error the error pointer is pointing to is an error.

error.special.entry.is.warning This Boolean placeholder is true if the error the error pointer is pointing to is a warning.

error.special.entry.maptoresource Maps the entry in the error list to its meaning.

error.special.next.entry This Boolean placeholder is used when going through a list of errors. It is true if there is a next entry and false if there is not.

Placeholders related to the extended search function

extended.search.count The number of items in the search result list.

extended.search.customindex.count The number of custom indexed variables that have been configured.

extended.search.customindex.entry.attribute The attribute name of the custom indexed attribute the pointer is pointing to.

extended.search.customindex.entry.type The type of the custom indexed attribute the pointer is pointing to, e.g. NameKey, Time, etc.

extended.search.customindex.next.entry The custom index pointer. Returns true if there is a next custom indexed attribute, false otherwise.

extended.search.entry.call(ActionName) Calls the action ActionName

extended.search.entry.get_attrib(AttributeName) Gets the attribute of type AttributeName for the entry the search results pointer is pointing to.

extended.search.entry.icon The icon for the entry the search results pointer is pointing to.

- extended.search.entry.parents.count** The number of parent collections that the entry the search results pointer is pointing to has.
- extended.search.entry.parents.next_entry** This Boolean placeholder is used when displaying the list of parents of an item in the search results. It is true if there is a next entry and it moves the search results-parents pointer to the next entry.
- extended.search.entry.parents.entry.uri** The URI of the collection the search results-parents pointer is pointing to.
- extended.search.entry.parents.entry.title** The title of the collection the search results-parents pointer is pointing to.
- extended.search.entry.title** The title of the search result the search results pointer is pointing to.
- extended.search.entry.uri** The URI of the search result the search results pointer is pointing to.
- extended.search.form.fulltextquery** This placeholder is used as the NAME attribute when submitting the fulltext query to the server in the search form.
- extended.search.form.keyquery** This placeholder is used as the NAME attribute when submitting the key query to the server in the search form.
- extended.search.form.languages** This placeholder is used as the NAME attribute in forms when submitting the language the search is to be done in. The VALUE submitted must be a language prefix, e.g. "en" for English or "ge" for German.
- extended.search.form.map(AttributeName)** This is used to "map" a search term to an indexed attribute AttributeName, which means that the term will be searched for in this attribute. E.g. `<INPUT TYPE="checkbox" NAME="%%extended.search.form.map(title)%%" VALUE="Title=s1">` maps the search term s1 to title search.
- extended.search.form.maxobjects** This placeholder is used as the NAME attribute in forms when submitting the maximum number of objects to be shown in the search results.
- extended.search.form.objectquery** This placeholder is used as the NAME attribute when submitting the object query to the server in the search form.
- extended.search.form.scope** This placeholder is used as the NAME attribute in forms when submitting the scope of the search.
- extended.search.form.scope.server** This placeholder is used as the VALUE attribute in forms when the preference to search the entire server is submitted.
- extended.search.form.sortorder** This placeholder is used as the NAME attribute in forms when submitting the desired sort order of the search results.
- extended.search.form.var(variable)** This placeholder is used as the VALUE attribute in forms when submitting e.g. a search term or author name when making a search.
- extended.search.next_entry** This Boolean placeholder is used when displaying the list of search results. It is true if there is a next entry and it moves the search results pointer to the next entry.
- extended.search.vars.count** The number of variables that were submitted for a search using `extended.search.form.var(variable)`.
- extended.search.vars.next_entry** This Boolean placeholder is true if there is a next entry in the list of search variables.
- extended.search.vars.entry.name** The name of the variable the variable pointer is pointing to.
- extended.search.vars.entry.value** The value of the variable the variable pointer is pointing to.

Placeholders related to the WaveMaster process

- gate.date** Date on WaveMaster host (GMT).
- gate.hostname** Hostname of WaveMaster host (with multihoming the name of the host the user connected to).
- gate.hostname_with_port** Hostname of WaveMaster with port (if port is not 80).
- gate.object(ObjectName)** Object with ObjectName from WaveMaster's HTML directory. This directory is specified in the configuration file (.db.contr.rc) with the variable HTML_DIR.
- gate.port** Port WaveMaster is listening at.
- gate.protocol** The protocol the gateway is using.
- gate.time** Time on WaveMaster host (GMT).
- gate.version** Version of WaveMaster.

Placeholders related to the group list

- groups.count** The number of groups on the server.
- groups.entry.call(ActionName)** Calls the action ActionName for the group the group pointer is pointing to.
- groups.entry.get_attr(AttributeName)** Gets the attribute of type AttributeName for the group record the group pointer is pointing to.

groups.entry.icon The icon for group entries.
groups.entry.uri The URI of the group the group pointer is pointing to.
groups.next.entry Boolean placeholder which is true if there is a next entry in the list of groups.

Placeholders related to the Hyperwave Server

hyper.date Date on the Hyperwave server host (local time).
hyper.dbactions Number of database actions since the Hyperwave server was started.
hyper.dbserver.version The version of the database server module of the Hyperwave server being accessed.
hyper.dcsserver.cachehits.sinceup The number of cache hits in dcsserver since it has been up.
hyper.dcsserver.cachehits.60m The number of cache hits in dcsserver in the last 60 minutes.
hyper.dcsserver.cachehits.15m The number of cache hits in dcsserver in the last 15 minutes.
hyper.dcsserver.cachehits.1m The number of cache hits in dcsserver in the last minute.
hyper.dcsserver.cacheinquiries.sinceup The number of cache inquiries made to the document cache server since it has been up.
hyper.dcsserver.cacheinquiries.60m The number of cache inquiries made to the document cache server in the last hour.
hyper.dcsserver.cacheinquiries.15m The number of cache inquiries made to the document cache server in the last 15 minutes.
hyper.dcsserver.cacheinquiries.1m The number of cache inquiries made to the document cache server in the last minute.
hyper.dcsserver.filedescriptors The number of currently open file descriptors.
hyper.dcsserver.imagedata.sinceup The amount of image data served by the dcsserver since it has been up.
hyper.dcsserver.imagedata.60m The amount of image data served by the dcsserver in the last 60 minutes.
hyper.dcsserver.imagedata.15m The amount of image data served by the dcsserver in the last 15 minutes.
hyper.dcsserver.imagedata.1m The amount of image data served by the dcsserver in the last minute.
hyper.dcsserver.megsretrieved Number of megabytes retrieved from the Document Cache server.
hyper.dcsserver.otherdata.sinceup The amount of non-text and non-image data served by the dcsserver since it has been up.
hyper.dcsserver.otherdata.60m The amount of non-text and non-image data served by the dcsserver in the last hour.
hyper.dcsserver.otherdata.15m The amount of non-text and non-image data served by the dcsserver in the last 15 minutes.
hyper.dcsserver.otherdata.1m The amount of non-text and non-image data served by the dcsserver in the last minute.
hyper.dcsserver.remoteports The number of ports which are open to receive a remote document from another Hyperwave server.
hyper.dcsserver.totaldata.1m, 15m, 60m The total amount of data served by dcsserver in the last 1, 15, and 60 minutes respectively.
hyper.dcsserver.totaldata.sinceup The total amount of data served by dcsserver since the server has been running.
hyper.dcsserver.users The number of currently logged in users.
hyper.dcsserver.textdata.sinceup The amount of text data served by the dcsserver since it has been up.
hyper.dcsserver.textdata.60m The amount of text data served by the dcsserver in the last 60 minutes.
hyper.dcsserver.textdata.15m The amount of text data served by the dcsserver in the last 15 minutes.
hyper.dcsserver.textdata.1m The amount of text data served by the dcsserver in the last minute.
hyper.dcsserver.upsince.date, .time Starting date and time of the Document Cache server process.
hyper.dcsserver.upsince.days, .hours, .minutes, .seconds Number of days, hours, minutes and seconds the Document Cache server has been running.
hyper.dcsserver.userjobs The number of user jobs.
hyper.dcsserver.version The version of the document cache server module of the Hyperwave server being accessed.
hyper.ftserver.queries.sinceup The number of fulltext queries made since the fulltext server has been up.
hyper.ftserver.queries.60m The number of fulltext queries made in the last 60 minutes.
hyper.ftserver.queries.15m The number of fulltext queries made in the last 15 minutes.
hyper.ftserver.queries.1m The number of fulltext queries made in the last minute.
hyper.ftserver.searchengine.name The search engine which is currently switched on for the server. Possible values are "verity" and "native".
hyper.ftserver.upsince.date The date the fulltext server was last started.
hyper.ftserver.upsince.days The amount of time in days the ftserver has been up.
hyper.ftserver.upsince.hours The amount of time in hours the ftserver has been up.
hyper.ftserver.upsince.minutes The amount of time in minutes the ftserver has been up.
hyper.ftserver.upsince.seconds The amount of time in seconds the ftserver has been up.
hyper.ftserver.upsince.time The time the fulltext server was last started.

hyper.ftserver.version The version of the fulltext server module of the Hyperwave server being accessed.
hyper.hgserver.version The version of the hgserver module of the Hyperwave server being accessed.
hyper.lastreorg.date The date of the last reorganization of the Hyperwave server.
hyper.lastreorg.days The number of days since the last reorganization of the Hyperwave server.
hyper.lastreorg.hours The number of hours since the last reorganization of the Hyperwave server.
hyper.lastreorg.minutes The number of minutes since the last reorganization of the Hyperwave server.
hyper.lastreorg.seconds The number of seconds since the last reorganization of the Hyperwave server.
hyper.lastreorg.time The time of the last reorganization of the Hyperwave server.
hyper.modifications.sinceup, .60m, .15m, .1m Number of database modifications on the Hyperwave server since the Hyperwave server was started, in the last 60 minutes, the last 15 minutes and the last minute.
hyper.object(ObjectName) Object with ObjectName from the Hyperwave server.
hyper.objects.sinceup Number of objects retrieved from the Hyperwave server since the Hyperwave server was started.
hyper.objects.60m Number of objects retrieved from the Hyperwave server in the last 60 minutes.
hyper.objects.15m Number of objects retrieved from the Hyperwave server in the last 15 minutes.
hyper.objects.1m Number of objects retrieved from the Hyperwave server in the last minute.
hyper.queries.sinceup, .60m, .15m, .1m Number of database queries on the Hyperwave server since the Hyperwave server was started, in the last 60 minutes, the last 15 minutes and in the last minute.
hyper.time Time on the Hyperwave server host (local time).
hyper.upsince.date Starting date of the Hyperwave server.
hyper.upsince.time Starting time of the Hyperwave server.
hyper.upsince.days Number of days the Hyperwave Server process has been running.
hyper.upsince.hours Number of hours the Hyperwave server process has been running.
hyper.upsince.minutes Number of minutes the Hyperwave server process has been running.
hyper.upsince.seconds Number of seconds the Hyperwave server process has been running.
hyper.usersessions Total number of user sessions since the Hyperwave server was started.
hyper.servername The name of the Hyperwave server being accessed (the server string, not the host name).

Placeholders for showing icons for Hyperwave object types in the collection hierarchy.

icon.alternativecluster.uri, icon.anchor.alternativecluster.uri URI of the icon for alternative clusters/anchors of alternative clusters.
icon.audio.uri, icon.anchor.audio.uri URI of the icon for audio objects/anchors of audio objects.
icon.cgi.uri, icon.anchor.cgi.uri URI of the icon for CGI objects/anchors of CGI objects.
icon.cluster.uri, icon.anchor.cluster.uri URI of the icon for clusters/anchors of clusters.
icon.collection.uri, icon.anchor.collection.uri URI of the icon for collections/anchors of collections.
icon.ftp.uri, icon.anchor.ftp.uri URI of the icon for ftp objects/anchors of ftp objects.
icon.generic.uri, icon.anchor.generic.uri URI of the icon for generic objects/anchors of generic objects.
icon.gopher.uri, icon.anchor.gopher.uri URI of the icon for gopher objects/anchors of gopher objects.
icon.hgi.doc.uri, icon.anchor.hgi.doc.uri URI of the icon for HGI documents objects/anchors of HGI documents.
icon.hgi.coll.uri, icon.anchor.hgi.coll.uri URI of the icon for HGI collections objects/anchors of HGI collections.
icon.image.uri, icon.anchor.image.uri URI of the icon for image objects/anchors of image objects.
icon.movie.uri, icon.anchor.movie.uri URI of the icon for movie objects/anchors of movie objects.
icon.multicluster.uri, icon.anchor.multicluster.uri URI of the icon for multiclusters/anchors of multiclusters.
icon.postscript.uri, icon.anchor.postscript.uri URI of the icon for postscript/anchors of postscript documents.
icon.program.uri, icon.anchor.program.uri URI of the icon for program objects/anchors of program objects.
icon.query.uri, icon.anchor.query.uri URI of the icon for query objects/anchors of query objects.
icon.scene.uri, icon.anchor.scene.uri URI of the icon for scene objects/anchors of scene objects.
icon.sequence.uri, icon.anchor.sequence.uri URI of the icon for sequences/anchors of sequences.
icon.server The server icon.
icon.server.uri The URI of the server icon.
icon.serverpool The server pool icon.
icon.serverpool.uri The URI of the server pool icon.
icon.telnet.uri, .anchor.telnet.uri URI of the icon for telnet objects/anchors of telnet objects.
icon.text.uri, .anchor.text.uri URI of the icon for text objects/anchors of text objects.
icon.unknown.uri, .anchor.unknown.uri URI of the icon for unknown objects/anchors of unknown objects.
icon.wais.uri, .anchor.wais.uri URI of the icon for wais objects/anchors of wais objects.

icon.www.uri, **.anchor.www.uri** URI of the icon for www objects/anchors of www objects.

Placeholders which are used as the NAME attribute when inserting attribute information in forms

- insert.attributes.form.key** This placeholder is used as the NAME attribute in forms to submit an attribute type when adding a new attribute while inserting a new object. It can have one or two parameters.
- insert.attributes.form.newpassword** This placeholder is used as the NAME attribute in forms to submit a password when creating a user record.
- insert.attributes.form.newpasswordverify** This placeholder is used as the NAME attribute in forms to submit the verification of a new password when creating a user record.
- insert.attributes.form.value** This placeholder is used as the NAME attribute in forms to submit an attribute value when adding a new attribute while inserting a new object. It can have one or two parameters.
- insert.attributes.form.url** This placeholder is used as the NAME attribute in forms to submit a URL when inserting a remote object.
- insert.form.collection** This placeholder is used as the NAME attribute when submitting the name or GOid of the collection a new object is to be inserted into.
- insert.form.language** This placeholder is used as the NAME attribute when submitting the language prefix for the title of an object which is being inserted.
- insert.link.destination.object** This placeholder returns the GOid of the destination object of an anchor.
- insert.link.form.destination.attrib(AttributeName)** This placeholder is used to submit an arbitrary attribute type for the link destination.
- insert.link.form.destination.object** This placeholder is used to submit the name or GOid of the object which will contain the link target when making a link.
- insert.link.form.destination.text** This placeholder is used to submit the phrase used as the destination anchor when making a link.
- insert.link.form.source.all** This placeholder is used as the NAME attribute when submitting the preference that all occurrences of the source phrase should be made into source anchors.
- insert.link.form.source.all.yes** This placeholder is used as the VALUE attribute when submitting the preference that all occurrences of the source phrase should be made into source anchors.
- insert.link.form.source.attrib(AttributeName)** This placeholder is used to submit an arbitrary attribute type for the link source.
- insert.link.form.source.object** This placeholder is used to submit the name or GOid of the source object when making a link.
- insert.link.form.source.text** This placeholder is used to submit the phrase used as the link anchor when making a link.
- insert.link.is_multipledestination** This Boolean placeholder is true if the destination anchor phrase occurs more than once in the text when making a link.
- insert.link.is_multiplesource** This Boolean placeholder is true if the source anchor phrase occurs more than once in the text when making a link.
- insert.link.multipledestination.content** This placeholder displays the text that should contain the destination anchor such that there are radio buttons which can be used to pick out exactly one destination for the anchor.
- insert.link.multiplesource.content** This placeholder displays the text that should contain the source anchor such that there are checkboxes which can be used to pick out one or more sources for the anchor.
- insert.link.source.object** This placeholder returns the GOid of the source object of an anchor.
- insert.newobject.call(ActionName)** Calls the action ActionName for the most recently inserted object.
- insert.newobject.get_attr(AttributeName)** Gets the attribute AttributeName for the most recently inserted object.
- insert.newobject.icon** The icon for the most recently inserted object.
- insert.newobject.title** This placeholder returns the title of the most recently inserted object.
- insert.newobject.uri** This placeholder returns the URI of the most recently inserted object.
- insert.text.form.content** This placeholder is used as the NAME attribute when submitting text in a text area.
- insert.text.is_multipledestination** This Boolean placeholder is used when making an annotation and is true when there are multiple occurrences of the destination phrase.
- insert.text.multipledestination.content** This placeholder is used when making an annotation and displays the text with radio buttons next to each occurrence of the destination anchor phrase, if there are multiple occurrences.
- insert.text.form.destination.object** This placeholder is used as the NAME attribute when submitting the GOid of the destination object when making an annotation.
- insert.text.form.destination.text** This placeholder is used as the NAME attribute when submitting the phrase to be used as the destination anchor when making an annotation.

insert.upload.form This placeholder is used as the NAME attribute when submitting a file to the server.
insert.upload.form.base This placeholder is used as the NAME attribute when submitting the base when inserting a new text document.

Placeholders for information about the last visited collection

lastcoll.call(ActionName) Calls an action ActionName for the last visited collection.
lastcoll.get_attrib(AttributeName) Gets the attribute with the attribute type AttributeName for the last collection visited.
lastcoll.is_coll_with_fullcollhead A Boolean placeholder which is true if the last visited collection is a collection with a full collection head. This can be used in "if" statements, e.g.

```
%%if lastcoll.is_coll_with_fullcollhead%%
[further PLACE or HTML statements]
%%endif%%
```

lastcoll.is_editable A Boolean placeholder which is true if the last visited collection is a collection which is editable by the current user.
lastcoll.relpath Path with which the last visited collection can be accessed. This placeholder returns the name or GOid of the collection.
lastcoll.title This gets the Title attribute of the last collection visited.
lastcoll.uri The URI of the last collection visited.

Placeholders for the move/copy function

movecopy.object.form This is used to submit the GOids of the objects to be moved or copied.
movecopy.object.form.destination This placeholder is used as the NAME attribute in forms when submitting the collection the objects are to be moved/copied to.
movecopy.object.form.mode This placeholder is used as the NAME attribute in forms when submitting the mode (move or copy) to be used.
movecopy.object.form.mode.copy The VALUE attribute submitted in a form when you want to use "copy" mode.
movecopy.object.form.mode.move The VALUE attribute submitted in a form when you want to use "move" mode.
movecopy.object.form.parent This placeholder is used as the NAME attribute in forms when submitting the collection the objects are to be moved/copied from.

Placeholders for information about the current object

object.action.uri URI of the current action including parameter ;internal&action=... if there is one.
object.attributes.entry This placeholder is used to display the key-value pair for an attribute (e.g. Author=hwsystem).
object.attributes.entry.is_editable This Boolean placeholder is true if the attribute that the attribute pointer is pointing to is of an editable type.
object.attributes.entry.key Attribute type of the attribute the attribute pointer is pointing to.
object.attributes.entry.value Value of the attribute the attribute pointer is pointing to.
object.attributes.entry.value.escaped A placeholder used to allow umlauts and other special characters to be expressed as HTML entities when submitting edited attributes.
object.attributes.next_entry This Boolean placeholder moves the attributes pointer to the next attribute entry of the current object. Returns true if there is a next entry, otherwise false.
object.call(ActionName) Calls an action ActionName for the current object.
object.content Content of the current object, if it is a text document.
object.description The Description attribute of the current object.
object.edit.content This placeholder returns the content of the current object if it is a text object. It is used to display the content so the user can edit it.
object.get_attrib(AttributeName) This placeholder gets the value of the attribute AttributeName for the current object.
object.html.bodyattrs Body attributes of the current object. (for HTML BODY tag)
object.html.doctype DOCTYPE information of the current object. (for HTML DOCTYPE tag)
object.html.head This placeholder is used to return Java script to the heading of a document when displaying the document (the heading is removed when the document is uploaded to Hyperwave and if there was Java script in the heading it is stored separately).

object.html.links LINK information of the current object. (for HTML LINK tag)

object.html.meta META information of the current object. (for HTML META tag)

object.icon Icon to be shown for the object type of the current object.

object.imagemap Imagemap for the current object, if it is an image with imagemap.

object.imagemap.usemap Usemap text for the current object, if it is an image with imagemap.

object.inherit.rights This placeholder returns as value the rights of the collection a new object is being inserted into and is used to make it possible for a new object to inherit rights.

object.is_coll_with_collhead This Boolean placeholder is true if the collection has a full collection head or a collection head, otherwise false.

object.is_coll_with_fullcollhead This Boolean placeholder is true if the collection has a full collection head, otherwise false.

object.is_editable This Boolean placeholder is true if the current object can be edited by the current user. This function is quite expensive because it locks the current object and unlocks it again. Thus it should only be used if really needed.

object.is_locked This Boolean placeholder is true if the current object is locked.

object.is_in_sequence This Boolean placeholder is true if the current object is in a sequence, otherwise false.

object.is_realcollection This Boolean placeholder is true if the collection is a real collection, not a cluster and is false otherwise.

object.lock.get_attrib(AttributeName) Gets the value of the attribute of type AttributeName for the lock object. The possible attribute types are Author (the person who locked the object) and TimeCreated (the time when the object was locked).

object.query Query of the current object, i.e. the characters after '?' in the URI.

object.redirect.uri URI for redirection if current object is a remote WWW-, FTP- gopher- or telnet object.

object.relpath Path with which the current object can be accessed. This is the current object's name or its GOid, if there is no name.

object.size The exact size of the current object in bytes.

object.size.approximate The approximate size of the current object in bytes, kbytes or Mbytes, depending on the size of the object.

object.title Gets the title of the current object.

object.uri Gets the URI of the current object.

object.uri_plain The URI of an object without the query information.

Placeholders for information about the parents of the current object

parents.count Number of parents the current document has.

parents.entry.call(ActionName) Calls an action ActionName for the parent the parent pointer is pointing to.

parents.entry.get_attrib(AttributeName) Gets the attribute with AttributeName of the parent the parent pointer is pointing to.

parents.entry.icon Icon to be shown for the object type of the parent the parent pointer is pointing to.

parents.entry.sequence.count Number of children of the parent the parent pointer is pointing to.

parents.entry.sequence.first—prev—next—last.call(ActionName) Calls an action ActionName for the first, previous, next or last sibling in the sequence the parent pointer is pointing to.

parents.entry.sequence.first—prev—next—last.title The title of the first, previous, next or last sibling in the sequence the parent pointer is pointing to.

parents.entry.sequence.first—prev—next—last.uri The URI of the first, previous, next or last sibling in the sequence the parent pointer is pointing to.

parents.entry.sequence.next This Boolean placeholder is true if there is a next object in the parent sequence the parent pointer is pointing to.

parents.entry.sequence.number Order of the object in the parent sequence the parent pointer is pointing to.

parents.entry.sequence.prev This Boolean placeholder is true if there is a previous object in the parent sequence the parent pointer is pointing to.

parents.entry.title Title of the parent the parent pointer is pointing to.

parents.entry.uri URI of the parent the parent pointer is pointing to.

parents.next_entry This Boolean placeholder moves the parent pointer to the next entry in the list of parents. Returns true if there is a next entry, otherwise false.

parents.sequence.count The number of parents of an object which are sequences.

Placeholders for the change password form

password.form.newpassword Constant for the NAME of the input field for the new password.

password.form.newpasswordverify Constant for the NAME of the input field for the new password for verification.

password.form.oldpassword Constant for the NAME of the input field for the old password.

Placeholders for information about references to an object

references.in.count, references.out.count The number of references to/from the current object.

references.in.entry.call(ActionName), references.out.entry.call(ActionName) Calls an action ActionName for the current object in the reference in/out list.

references.in.entry.get_attr(AttributeName), references.out.entry.get_attr(AttributeName) Gets the attribute AttributeName for the current object in the reference in/out list.

references.in.entry.icon, references.out.entry.icon The icon for the current object in the reference in/out list.

references.in.entry.title, references.out.entry.title Gets the title of the current object in the reference in/out list.

references.in.entry.uri, references.out.entry.uri Gets the URI of the current object in the reference in/out list.

references.in.next_entry, references.out.next_entry These Boolean placeholders move the in reference/out reference list pointer to the next entry and returns true if there is a next entry.

Placeholders for the search result list

search.author This gives the value of the author which was submitted using search.form.author.

search.created_before.date This gives the value of the date which was submitted using search.form.created_before.date.

search.created_or_modified_after.date This gives the value of the date which was submitted using search.form.created_or_modified_after.date.

search.fulltext.entry.parents.entry.title This gives the title of the parent of the fulltext search result item the pointer is pointing to.

search.fulltext.next_entry This Boolean placeholder is true if there is a next entry in the list of fulltext search results.

search.is_fulltext_search This Boolean placeholder is true if the search was a fulltext search, otherwise false.

search.is_title_search This Boolean placeholder is true if the search was a title and keyword search, otherwise false.

search.is_user_group_search This Boolean placeholder is true if the search was in users and groups, otherwise false.

search.searchexp This gives the value of the search term which was submitted using search.form.searchexp.

search.title.count, .fulltext.count Number of matched objects in title/fulltext search.

search.title.entry.call(ActionName), .fulltext.entry.call(ActionName) Calls an action ActionName for the entry the search pointer is pointing to.

search.title.entry.get_attr(AttributeName), .fulltext.entry.get_attr(AttributeName) Attribute with AttributeName of the entry the search pointer is pointing to.

search.title.entry.icon, .fulltext.entry.icon Icon to be shown for the object type of the entry the search pointer is pointing to.

search.title.entry.parents.count, .fulltext.entry.parents.count The number of parents the entry the search pointer is pointing to has.

search.title.entry.parents.entry.title This gives the title of the parent of the title search result item the pointer is pointing to.

search.title.entry.parents.entry.uri, .fulltext.entry.parents.entry.uri This gives the URI of the parent of the title/fulltext search result item the pointer is pointing to.

search.title.entry.parents.next_entry, .fulltext.entry.parents.next_entry This Boolean placeholder moves the parent pointer to the next entry. Returns true if there is a next entry, otherwise false.

search.title.entry.title, .fulltext.entry.title Title of the entry the search pointer is pointing to.

search.title.entry.uri, .fulltext.entry.uri URI of the entry the search pointer is pointing to.

search.title.next_entry, .fulltext.next_entry These Boolean placeholders move the search pointer to the next entry in the list of title search results and the list of fulltext search results respectively. They return true if there is a next entry, otherwise false.

Placeholders for the search form

search.form.author Constant for the name of the input field for the author that should be searched for.

search.form.created_before.date Constant for the name of the input field for the date found objects should be created before.

search.form.created_or_modified_after.date Constant for the name of the input field for the date found objects should be created or modified after.

search.form.languages Constant for the name of the input field for the search language. (a text field)

search.form.maxobjs Constant for the name of the input field for the maximal number of found objects.

search.form.scope Constant for the name of the input field for the search scope. (radio buttons)

search.form.scope.lastcoll, .server Constants for the values of the input fields for the search scope.

search.form.scope.lastcoll.title Title of the last visited collection.

search.form.searchexp Constant for the NAME attribute of the text input field for the search expression.

search.form.searchin.title, .keyword, .fulltext Constants for the NAME attributes of the checkboxes used for choosing what to search in (titles, keywords, fulltext).

search.form.searchin.name, .uname, .ugroup, .group Constants for the NAME attributes used to submit what to search in during user/group search.

search.form.yes Constant for the VALUE attribute of check box input fields.

Placeholders for sequences

sequence.count Number of objects in the current sequence.

sequence.first.title Title of the first object in the sequence.

sequence.first.uri URI of the first object in the sequence.

sequence.last.title Title of the last object in the sequence.

sequence.last.uri URI of the last object in the sequence.

sequence.next A Boolean variable which is true if there is a next object in the sequence, false if the last object of the sequence is being shown.

sequence.next.title Title of the next object in the sequence.

sequence.next.uri URI of the next object in the sequence.

sequence.number Position in the current sequence.

sequence.prev A Boolean variable which is true if there is a previous object in the sequence, false if the first object in the sequence is being shown.

sequence.prev.title Title of the previous object in the sequence.

sequence.prev.uri URI of the previous object in the sequence.

sequence.title Title of the current sequence.

sequence.uri URI of the current sequence.

Placeholders for information about the server pool

serverpool.count The number of servers in the server pool.

serverpool.entry.call(ActionName) Calls an action ActionName for the current server pool entry.

serverpool.entry.get_attrib(AttributeName) Gets the attribute AttributeName for the current server pool entry.

serverpool.entry.icon Gets the icon of the current server pool entry.

serverpool.entry.title Gets the title of the current server pool entry.

serverpool.entry.uri Gets the URI of the current server pool entry.

serverpool.next_entry Moves the server pool entry pointer. Returns true if there is a next entry.

Placeholders for information about the current session

session.form.language Constant for the NAME attribute of the input field for the preferred language.

session.getvar(VariableName) Gets the variable VariableName that was previously set from an input field of a form or which is set by the system. Example: `%%if session.getvar(EDIT_MODE) == 'yes'%%` can be used to branch off if the user is in WaveMaster's edit mode.

session.language The currently selected language (as Hyperwave language string, e.g. ge, en)

session.prefmimetypes The current user's preferred MIME types. These are in the form `MimeType[";Quality=number"]*(,MimeType[";Quality=number"])`
e.g. `text, image, application/postscript or image;Quality=50.`

session.username The user name of the current user on the Hyperwave server (if the user is member of the system group (system) is attached to the user's name)

session.user.name Same as `session.username`.

session.username.nosystem The username of the current user on the Hyperwave server (without (system) attached)

session.user.name.nosystem Same as session.username.nosystem.

session.user.get_attrib(AttributeName) Gets the attribute AttributeName for the current user.

session.user.is_system This Boolean variable is true if the current user is a member of the group "system".

Placeholders for the list of currently logged in users.

userlist.count Number of users in the user list.

userlist.entry.client Client via which the user pointed to by the user list pointer is accessing the Hyperwave server.

userlist.entry.hostname Name of the host or proxy from where the user pointed to by the user list pointer is connected to the Hyperwave server.

userlist.entry.id Id of the user pointed to by the user list pointer.

userlist.entry.idle.time Idle time of the user pointed to by the user list pointer.

userlist.entry.is_self This Boolean variable returns true if the user the user list pointer is pointing to is the current user.

userlist.entry.login.date Date of login of the user pointed to by the user list pointer.

userlist.entry.login.time Time of login of the user pointed to by the user list pointer.

userlist.entry.username Name of the user pointed to by the user list pointer.

userlist.next_entry This Boolean placeholder moves the user list pointer to the next entry. Returns true if there is a next entry, otherwise false.

Placeholders having to do with the list of all users on the server.

users.count The number of user records on the server.

users.entry.call(ActionName) Calls the action ActionName for the user the user list pointer is pointing to.

users.entry.get_attrib(AttributeName) Gets the attribute of type AttributeName from the user record the user list pointer is pointing to.

users.entry.icon The icon for user entries.

users.entry.uri The URI of the entry in the user list that the pointer is pointing to.

users.next_entry This Boolean placeholder moves the user list pointer to the next entry. Returns true if there is a next entry, otherwise false.

Other placeholders

lastobj.call(ActionName) Calls an action ActionName for the last accessed object.

Appendix B

Hyperwave Tools Used in the NQA System

B.1 General Information

In this chapter only these *Hyperwave Tools* are listed which were used in the NQA system's CGI scripts. The manual pages for all *Hyperwave Tools* can be found in the *Hyperwave Reference Guide*¹ [13][18]. The author of the manual pages is *Bruno Reisinger* <breising@hyperwave.com>.

B.1.1 Default Values and Environment Variables

Default values are overridden by environment variables which are overridden by commandline options.

¹All manual pages for the *Hyperwave Tools* can be accessed online under http://www.hyperwave.de/hw_tools

Option	Default Value	Environment Variable	Short Description
<i>-hwhost</i>	localhost	HWHOST	name of the Hyperwave server
<i>-hwport</i>	418	HWPORT	port number of the Hyperwave server
<i>-language</i>	en	HWLANGUAGE	language of the title
<i>-sortorder</i>		HWSORTORDER	sort order of objects
<i>-cdate</i>		HWCDATE	creation time
<i>-odate</i>		HWODATE	opening time
<i>-edate</i>		HWEDATE	expiration time
<i>-rights</i>		HWRIGHTS	access rights
<i>-user</i>		HWUSER	identify as user
<i>-parent</i>		HWPARENT	name or id of parent collection

B.1.2 Return Codes

Return Code	Description
>0	Hyperwave server error
0	No error occurred
-1	Wrong argument in commandline
-2	Wrong language
-3	User not identified
-4	No objects found
-5	More than one object found and option <i>-mult</i> not used
-6	Requested attribute not found in current object
-7	Collection not found
-8	Parent not found
-9	Document to replace not found
-10	File not found
-11	Error while parsing
-12	Attribute file not found
-13	Error in attribute file
-14	Object to replace is not a document
-15	Upload failed
-16	File open error
-17	More than one parent exists (specify a parent or use <i>-mult</i>)

B.2 hwinsdoc

hwinsdoc may be used to insert or replace any type of Hyperwave document

Option	Description
Getting help and version info	
<i>-help</i>	prints the usage information (a short description of each value) and the default values.
<i>-version</i>	prints the version string of the Hyperwave tool
Connection to Hyperwave Server	
<i>-hwhost</i> hwhost (HWHOST)	specifies the Hyperwave server host
<i>-hwport</i> hwport (HWPORT)	specifies the port to connect to
Identification	
<i>-identify</i> ['user[password]']	manual identification
<i>-user</i> user (HWUSER)	shifts the identification
Specifying the parent collection (required!)	
<i>-parent</i> parent (HWPARENT)	name or id of parent collection
Specifying the document type to insert (required!)	
<i>-type</i> type	sets the type of the document
<i>-mime</i> mimetype	sets the MIMEType of the document; the MIMEType is checked against the value of option <i>-type</i> . If option <i>-type</i> is omitted, hwinsdoc tries to generate the DocumentType from the MIMEType.
Specifying the document title (Required option unless you are inserting HTML or HTF document!)	
<i>-title</i> title	title of the document
<i>-language</i> lang (HWLANGUAGE)	specifies the language of the title; has no effect for a formatted title
<i>-formatted</i>	title is already formatted
Specifying the document name	
<i>-name</i> name	name of the document; it is possible to set several names at once. -name n1 -name 'n2 n3 n4' -name n5 sets the name attribute n1, n2, n3, n4, n5

Option	Description
Adding additional attributes	
<i>-cdate</i> cdate (HWCDATE)	sets <i>TimeCreated</i>
<i>-edate</i> edate (HWEDATE)	sets <i>TimeExpired</i>
<i>-odate</i> odate (HWODATE)	sets <i>TimeOpen</i>
<i>-rights</i> rights (HWRIGHTS)	sets the access rights of the document
Reading additional attributes from a file or STDIN	
<i>-attribute</i> file	you can use this to set attributes which you cannot set directly (e.g. Keyword). Each line must have the form: attribute=value hwinsdoc stops reading on empty lines or end of file.
Options which depend on the type of document beeing inserted You have to use one of these options; it must be in accordance with the type (mimetype) set! If only one option is available then it is required. Generic documents require the -path and remote documents the option -protocol .	
Inserting images, movies, PostScript, program, scene, sound or text (plain, HTML or HTF) documents	
<i>-path</i> path	specifies the name of the file which contains the data of the new document
Inserting a CGI Document	
<i>-relurl</i> relurl	specifies a reference to a CGI program reachable via the CGI directory of the server (hwsystem/dserver/cgi)
Inserting a Generic document	
<i>-path</i> path	specifies the name of the file which contains the data of the new document
<i>-genmime</i> mimetype	sets the mime type of the generic document
<i>-arguments</i> arguments	arguments of a generic document
Inserting a Remote Document	
<i>-protocol</i> protocol	sets the protocol and thus the type of remote document
<i>-host</i> host	sets the remote host (e.g. WWW host)
<i>-port</i> port	sets the remote port (e.g. WWW port)
<i>-path</i> path	sets the path; the meaning depends on the protocol type (e.g. path of the WWW document)

Option	Description
Test Mode	
<i>-test</i>	prints only the attributes of the document and does not insert the document
Replacing an existing Hyperwave Document	
<i>-replace</i> replace	name or id of the document to replace
<i>-path</i> path	name or the file which holds the new data for the document

Examples

```
hwinsdoc -parent testcoll -mime text/html -path mytext.html
```

inserts the HTML test `mytest.html` in collection `testcoll`.

```
hwinsdoc -par hotlist -title 'Introducing HTML 3.2'
-rights 'R:a' -type R -prot http -host www.w3.org
-path '/pub/WWW/MarkUp/Wilbur/'
```

inserts the URL of the Introduction to HTML 3.2 into the collection with name `hotlist`. The remote object should be readable for its author.

```
hwinsdoc -par pictures -title 'Car' -type I -pat car.gif
```

inserts the image `car.gif` in the collection `pictures`.

```
hwinsdoc -replace mytext -path mytextnew.html
```

replaces the document with name `mytext`.

B.3 hwinscoll

hwinscoll may be used to insert collections and clusters in Hyperwave.

Option	Description
Getting help and version info	
<i>-help</i>	prints the usage information (a short description of each value) and the default values.
<i>-version</i>	prints the version string of the Hyperwave tool
Connection to Hyperwave Server	
<i>-hwhost</i> hwhost (HWHOST)	specifies the Hyperwave server host
<i>-hwport</i> hwport (HWPORT)	specifies the port to connect to
Identification	
<i>-identify</i> ['user[password]']	manual identification
<i>-user</i> user (HWUSER)	shifts the identification
Specifying the parent collection (required!)	
<i>-parent</i> parent (HWPARENT)	name or id of parent collection
Specifying the title of the collection (Required option unless you are inserting HTML or HTF document!)	
<i>-title</i> title	title of the collection
<i>-language</i> lang (HWLANGUAGE)	specifies the language of the title; has no effect for a formatted title
<i>-formatted</i>	title is already formatted
Specifying the name of the collection (Required option, unless collection is a cluster (option <i>-cluster</i>))	
<i>-name</i> name	name of the document; it is possible to set several names at once. -name n1 -name 'n2 n3 n4' -name n5 sets the name attribute n1, n2, n3, n4, n5
Adding additional attributes	
<i>-cdate</i> cdate (HWCDATE)	sets <i>TimeCreated</i>
<i>-edate</i> edate (HWEDATE)	sets <i>TimeExpired</i>
<i>-odate</i> odate (HWODATE)	sets <i>TimeOpen</i>
<i>-rights</i> rights (HWRIGHTS)	sets the access rights of the document
<i>-description</i> description	adds a short description
<i>-sortorder</i> so (HWSORTORDER)	defines the order in which the collection's children should be displayed
Collection is a Cluster	
<i>-cluster</i>	collection is a cluster; option <i>-name</i> is not necessary

Examples

```
hwinscoll -parent doc -name doc/nov96 -title 'November 1996'
```

inserts a subcollection in collection doc with name doc/nov96 and title en:November 1996.

```
hwinscoll -parent '~fmaier' -form -title  
'Title=en:Welcome\Title=ge:Willkommen' -edate 97/03/01 -cluster
```

insert a multilingual cluster with an expiration date.

B.4 hwdownload

hwdownload downloads a collection structure and content from a Hyperwave server and maps it to a corresponding directory structure.

Option	Description
Getting help and version info	
<i>-help</i>	prints the usage information (a short description of each value) and the default values.
<i>-version</i>	prints the version string of the Hyperwave tool
Connection to Hyperwave Server	
<i>-hwhost</i> hwhost (HWHOST)	specifies the Hyperwave server host
<i>-hwport</i> hwport (HWPORT)	specifies the port to connect to
Identification	
<i>-identify</i> ['user[password]']	manual identification
<i>-user</i> user (HWUSER)	shifts the identification
Specifying the parent collection and Hyperwave attributes	
<i>-parent</i> parent (HWPARENT)	name or id of the parent collection (default is '/')
<i>-object</i> object	name or id of the object
Further options	
<i>-targetdir</i> targetdir	the path to download the data into (default is the current directory)
<i>-stripprefix</i> stripprefix	prefix you want to strip from the names of downloaded objects
<i>-base</i> base	the base to create in exportet HTML files
<i>-outofscopetarget</i> target	used to handle links which originate in documents being exportet but whose destinations are in documents on the Hyperwave server which are not being exportet; the value of target is a URL; if not set, these links will be removed
<i>-nonrecursive</i>	lets you specify not to handle collections recursively
<i>-nohmi</i>	suppresses creation of .hmi files
<i>-nolinks</i>	tells hwdownload not to create any file system links, thus losing some collection relational information

Examples

```
hwdownload -hwho someserver -ident myname -object someobject
-targetdir wheretodir -outofscopetarget http://someserver
```

This command downloads a collection tree to the file system. **someserver** is the internet address of the Hyperwave server containing the collection (or other object) **someobject**, which is to be downloaded recursively. If objects with restricted access rights are downloaded, then identification (**myname**) is necessary. **-targetdir** is used to download the information to a directory other than the current one. In order to keep links to documents which are on the server but outside the scope of the currently downloaded collection tree, the base URL **http://someserver** is specified in order to make these links point to the gateway of the server at that URL.

```
hwdownload -object internal/info/products -stripprefix internal
-outofscopetarget www.smith.com
```

downloads the recursive content of the collection **internal/info/products** from localhost. The prefix **internal** is removed from **Name** attributes of downloaded objects. The links in documents pointed to the server **www.smith.com**, if the documents they reference are not among the downloaded objects.

B.5 hwmodify

hwmodify is used to change the attributes of a set of selected objects.

Option	Description
Getting help and version info	
<i>-help</i>	prints the usage information (a short description of each value) and the default values.
<i>-version</i>	prints the version string of the Hyperwave tool
Connection to Hyperwave Server	
<i>-hwhost</i> hwhost (HWHOST)	specifies the Hyperwave server host
<i>-hwport</i> hwport (HWPORT)	specifies the port to connect to
Identification	
<i>-identify</i> ['user[password]']	manual identification
<i>-user</i> user (HWUSER)	shifts the identification
Preselecting an initial set of objects Objects can be preselected in one of three different ways. Note: You must use one of these options!	
Exact specification	
<i>-object</i> object	name or id of the object
Children of a collection	
<i>-collection</i> collection	name or id of a collection; selected objects are the children of the specified collection
Hyperwave KeyQuery	
<i>-keyquery</i> keyquery	specifies the KeyQuery term
<i>-language</i> lang (HWLANGUAGE)	replaces the default language for the query on titles
<i>-formatted</i>	keyquery is already formatted
<i>-collection</i> collection	name or id of a collection; all selected objects are descendants of the specified collection
Selection of descendants and anchors of preselcted objects In the next step you can use these option.	
<i>-recursive</i>	selects additionally all descendants, that is, recursive children of selected collections
<i>-anchor</i>	selects additionally all the anchors of the selected documents

Option	Description
Filtering preselected objects using a Hyperwave ObjectQuery The last step is to filter the selected objects using a Hyperwave ObjectQuery (optional).	
<i>-query</i> oquery	specifies the ObjectQuery term
The Modification Command (required!)	
<i>-command</i> command	<p>the modification command is applied to each object; the structure of the command:</p> <p>rem attr: remove the attribute attr rem attr=val: remove the attribute attr with value val add attr=val: add the attribute attr with value val command=<i>simple_command</i> command=<i>simple_command/command</i></p> <p>Thus it is possible to modify several attributes at once. Note: You are not able to change base attributes.</p>
Further optional parameters	
<i>-list</i>	displays only a list of selected objects; does not apply the modification command
<i>-multiple</i>	modify multiple objects

Examples

```
hwmodify -o technical/doc -comm 'rem Keyword'
```

removes all `Keyword` fields from object with name `technical/doc`.

```
hwmodify -coll technical -res -mult -comm 'rem Rights\add Rights=R:g iicm'
```

changes the `Rights` field of all objects under collection `technical` to `R:g iicm`.

```
hwmodify -form -key 'Name=xyz' -comm 'add Name=abc'
```

adds an additional unique name to object with name `xyz`.

B.6 hwci

hwci is used to check in one or more Hyperwave documents.

Option	Description
Getting help and version info	
<i>-help</i>	prints the usage information (a short description of each value) and the default values.
<i>-version</i>	prints the version string of the Hyperwave tool
Connection to Hyperwave Server	
<i>-hwhost</i> hwhost (HWHOST)	specifies the Hyperwave server host
<i>-hwport</i> hwport (HWPORT)	specifies the port to connect to
Identification	
<i>-identify</i> ['user[password]']	manual identification
<i>-user</i> user (HWUSER)	shifts the identification
Preselecting an initial set of objects You must use one of the options below to initially select the objects to be checked in.	
Exact specification	
<i>-object</i> object	name or id of the object
Children of a collection	
<i>-collection</i> collection	name or id of a collection; selected objects are the children of the specified collection
Hyperwave KeyQuery	
<i>-keyquery</i> keyquery	specifies the KeyQuery term
<i>-language</i> lang (HWLANGUAGE)	replaces the default language for the query on titles
<i>-formatted</i>	keyquery is already formatted
Hyperwave ObjectQuery It is possible to filter the selected objects using a Hyperwave ObjectQuery. This step is optional.	
<i>-query</i> oquery	specifies the ObjectQuery term

Option	Description
Further optional parameters	
<i>-recursive</i>	tells hwci to check in collections recursively
<i>-number</i> number	gives all objects being checked in the specified number
<i>-comment</i> comment	gives all objects being checked in the specified comment
<i>-forcedcheckin</i>	hwci normally checks in only documents that are already version controlled; this option tells the tool to also check in objects which were not previously version controlled
<i>-ignoreerrors</i>	tells hwci not to stop after finding an error.

Example

```
hwci -identify -collection papers -keyquery -formatted 'Author=jjones'
-recursive -number 2.0 -forcedcheckin
```

Checks in recursive children of collection **papers** on localhost if the author attribute of the document is **jjones**. Documents which were not previously version controlled are also checked in because of the **-forcedcheckin** option. All checked in documents are given the version number 2.0 unless the document already has a higher version number, in which case hwci outputs an error.

B.7 hwco

hwco is used to check out one or more Hyperwave documents.

Option	Description
Getting help and version info	
<i>-help</i>	prints the usage information (a short description of each value) and the default values.
<i>-version</i>	prints the version string of the Hyperwave tool
Connection to Hyperwave Server	
<i>-hwhost</i> hwhost (HWHOST)	specifies the Hyperwave server host
<i>-hwport</i> hwport (HWPORT)	specifies the port to connect to
Identification	
<i>-identify</i> ['user[password]']	manual identification
<i>-user</i> user (HWUSER)	shifts the identification
Preselecting an initial set of objects You must use one of the options below to initially select the objects to be checked out.	
Exact specification	
<i>-object</i> object	name or id of the object
Children of a collection	
<i>-collection</i> collection	name or id of a collection; selected objects are the children of the specified collection
Hyperwave KeyQuery	
<i>-keyquery</i> keyquery	specifies the KeyQuery term
<i>-language</i> lang (HWLANGUAGE)	replaces the default language for the query on titles
<i>-formatted</i>	keyquery is already formatted
Hyperwave ObjectQuery It is possible to filter the selected objects using a Hyperwave ObjectQuery. This step is optional.	
<i>-query</i> oquery	specifies the ObjectQuery term

Option	Description
Further optional parameters	
<i>-recursive</i>	tells hwco to check out collections recursively
<i>-forcedcheckout</i>	hwco normally checks out only documents that are already version controlled; this option tells the tool to also check in and then check out objects which were not previously version controlled
<i>-ignoreerrors</i>	tells hwco not to stop after finding an error.

Example

```
hwco -identify -collection home -recursive -forcedcheckout
```

Checks in recursive descendants of collection **home** on localhost. Non version controlled documents are checked in and checked out again.

B.8 hwdochistory

hwdochistory gives you a listing of the version history of a document.

Option	Description
Getting help and version info	
<i>-help</i>	prints the usage information (a short description of each value) and the default values.
<i>-version</i>	prints the version string of the Hyperwave tool
Connection to Hyperwave Server	
<i>-hwhost</i> hwhost (HWHOST)	specifies the Hyperwave server host
<i>-hwport</i> hwport (HWPORT)	specifies the port to connect to
Identification	
<i>-identify</i> ['user[password]']	manual identification
<i>-user</i> user (HWUSER)	shifts the identification
Selecting the document	
<i>-object</i> object	name or id of the object

Example

```
hwdochistory -identify -object report97
```

This command outputs the document version history for the object with the name `report97` on localhost.

Appendix C

A NQAHW Roadmap for Programmers

This is an excerpt of the *Roadmap for Programmers* which is used at ISCN Ltd. to configure the NQA system for customers. The reason why not the full roadmap is published there is, that it is an integral part of ISCN's NQA strategy to sell NQA configurations. So the publication of a do-it-yourself instruction is not in ISCN's interest.

The parts of the roadmap published should just demonstrate that the modular composition of NQA makes a reconfiguration relatively simple.

C.1 Scope of the Roadmap for Programmers

This roadmap is designed for system or web administrators who are planning to modify the NQAHW system. These modifications will become necessary if the roles or scenarios in your company aren't the same as used in the delivered NQAHW system.

In most cases the replacement of document templates or report forms will be enough to satisfy your companies requirements. For these tasks no programming knowledge is necessary, only HTML fundamentals should be present in order to make changes to existing templates or create new ones.

If scenarios in your company are totally different from these presented in the delivered NQAHW version, a configuration of Perl Scripts and PLACE Templates becomes necessary. So you have to be familiar with these two topics. There are step by step instructions included in this roadmap, although the understanding of the PLACE semantic and basic Perl programming skills will help you manage your configuration.

C.2 Replacing Document Templates in NQAHW

C.2.1 How Document Templates are handled in NQAHW

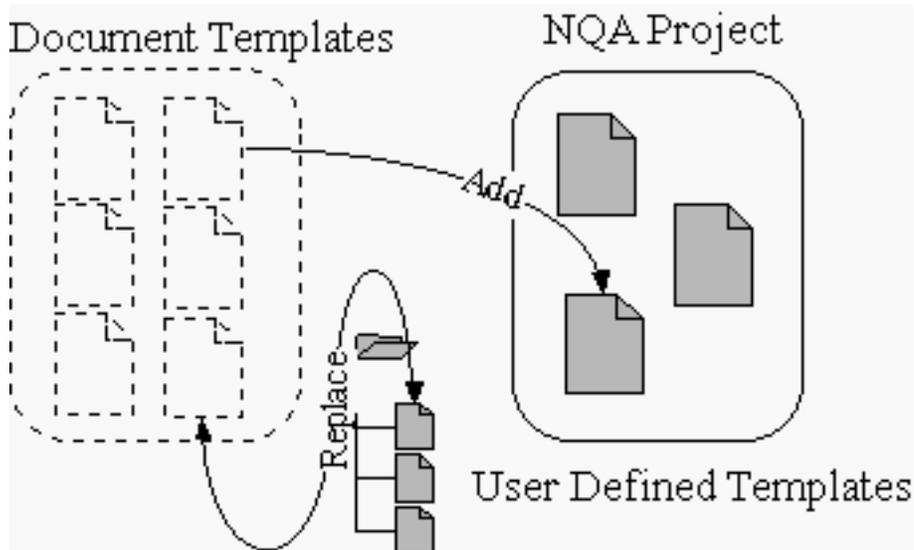


Figure C.1: How Document Templates are handled in NQAHW

Whenever you add a new document template to an existing NQA project, this template is taken from a repository. In the actual release, this repository is part of the NQA Manual. The NQA document management functions were designed in a way, that they identify a document template in the repository by its name. It doesn't inflect the document manager what kind of template is behind this name. So you can modify the existing HTML template or even replace it by whatever document you like.

C.2.2 The Replacement Procedure

First prepare a document template which meets the requirement of your company. Keep in mind, that this template should be editable by all users who might access the NQA system. So if you decide to put a Microsoft Word document template in the system, UNIX or MAC users won't be able to edit this template. Connect to your NQA server and browse to the document template you want to be replaced. Just follow the 'Document Templates' link on the NQAHW start page. If the template is displayed, login as system user and switch into the edit mode. Use the Hyperwave functionality to replace the displayed document on the server.

The replacement operation will only be successful if both of the documents, the document on the server and your new document are of the same MIME type (text/html, application/msword etc.).

Proceeding like this guarantees, that the name attribute of the new template stays in exactly that formatting, the document manager expects it to be.

C.3 Replacing Form Templates in NQAHW

C.3.1 How Form Templates are handled in NQAHW

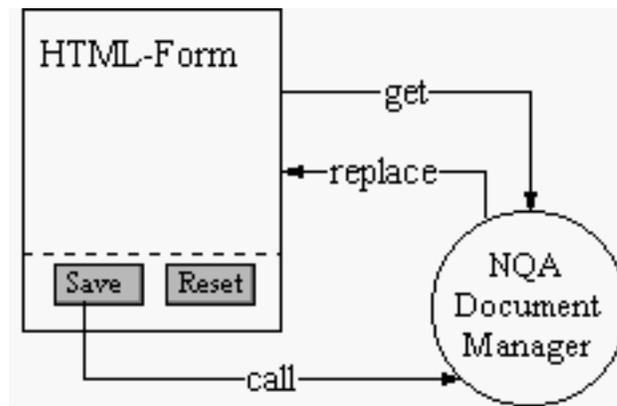


Figure C.2: How Form Templates are handled in NQAHW

Whenever you add a new form template to an existing NQA project, this template is taken from a repository. In the actual release, this repository is part of the NQA Manual. The NQA document management functions were designed in a way, that they identify a form templates in the repository by its name. It doesn't inflect the document manager what kind of template is behind this name. So you can modify the existing HTML template or even replace it by whatever document you like. The forms which come with NQAHW are HTML documents which can be handled by the document manager. If you create your own HTML form, you have to take care, that in the form the same action is called and that the button section in the form follows a defined ordering.

If you press the 'Save' button on a form, the document manager will be activated. It takes the form and changes it in a way, that the form content, you entered, will be set as the forms values. After this the document manager will replace the editable version on the server with the new created version.

If you want to use forms which are not based on HTML form techniques you have to modify some code in the document manager, all tasks are described below.

C.3.2 The Replacement Procedure

Replacing a form with a custom HTML form

The HTML form you create has to follow this structure in order to make the document manager work correctly:

```
<HTML>
<HEAD>
  <TITLE>place your title here</TITLE>
</HEAD>

<BODY>
  <FORM ACTION="/nqa_insertform.pl" METHOD="post">

  place your own tags here

  <INPUT TYPE="submit" NAME="Submit" VALUE="Save">
  <INPUT TYPE="reset" NAME="Reset" VALUE="Reset">
  </FORM>
</BODY>
</HTML>
```

- Connect to your NQA server and browse to the form template you want to be replaced. Just follow the 'Document Templates' link on the NQAHW start page.
- If the template is displayed, login as system user and switch into the edit mode.
- Use the Hyperwave functionality to replace the displayed document on the server. The replacement operation will only be successful if both of the documents, the document on the server and your new document are of the same MIME type (text/html, application/msword etc.).
- Proceeding like this guarantees, that the name attribute of the new template stays in exactly that formatting, the document manager expects it to be.

Using non-HTML forms

- First prepare a form template which meets the requirement of your company. Keep in mind, that this template should be editable by all users who might access the NQA system. So if you decide to put a Microsoft Word document template in the system, UNIX or MAC users won't be able to edit this template.
- Connect to your NQA server and browse to the document template you want to replace. Just follow the 'Document Templates' link on the NQAHW start page.
- If the template is displayed, login as system user and switch into the edit mode.
- Use the Hyperwave functionality to replace the displayed document on the server.
- Proceeding like this guarantees, that the name attribute of the new template stays in exactly in that formatting, the document manager expects it to be.
- The last step is to tell the document manager that it should not treat the form like a standard HTML form. In the directory your Hyperwave server is installed, open the file `dcserver/cgi/cgi-bin/nqa_copytemplate.pl`. Changes have to be made in the following section of the file.

```
%nqa_types = (
  nqa_wp => ["true","Draft","true","WP($tc)","false"],
  nqa_urd => ["true","Draft","true","URD($tc)","false"],
  nqa_add => ["true","Draft","true","ADD($tc)","false"],
  nqa_sum => ["true","Draft","true","SUM($tc)","false"],
  nqa_tp => ["true","Draft","true","TP Nr.$n ($tc)","false"],
  nqa_rr => ["true","Close","true","RR Nr.$n ($tc)","true"],
  nqa_atp => ["true","Draft","true","ATP Nr.$n ($tc)","false"],
  nqa_wpr => ["true","Draft","true","WPR($tc)","false"],
  nqa_spr => ["true","Close","true","SPR Nr.$n ($tc)","true"],
  nqa_scr => ["true","Close","true","SCR Nr.$n ($tc)","false"],
  nqa_smr => ["true","Close","true","SMR Nr.$n ($tc)","false"],
  nqa_cr => ["true","Draft","true","CR Nr.$n ($tc)","false"],
  nqa_fr => ["true","In Review","true","FR Nr.$n ($tc)","true"],
  nqa_ncr => ["true","Draft","true","NCR Nr.$n ($tc)","false"],
  nqa_ca => ["true","Draft","true","CA Nr.$n ($tc)","false"],
);
```

Look for the line where your form template is listed and change the last boolean value in this line to **false**. This indicates, that the document should not be treated as HTML form.

Bibliography

- [1] *AT&T Online Telework Guide*:
<http://www.att.com/telework>
- [2] Bredin Alice, *The Virtual Office Survival Handbook*, John Wiley & Sons, Inc., New York, 1996
- [3] *CGI.pm Manual Pages*:
<http://stein.cshl.org/WWW/software/CGI/cgi.docs.html>
- [4] Daily Kevin, *Quality management for software*, NCC Blackwell, Oxford, 1992
- [5] Deep John, Holfelder Peter, *CGI mit Perl*, SYBEX, Düsseldorf, 1997
- [6] Eisenberger Richard, *JavaScript*, Markt und Technik, Haar bei München, 1997
- [7] Flanagan David, *JavaScript, The Definitive Guide, Second Edition*, O'Reilly & Associates, Inc., Cambridge, 1997
- [8] Gütl C., Moser M., *Place Workshop*, Graz 1998
- [9] Hornby A.S., Parnwell E.C., *The Oxford English-Reader's Dictionary*, Langenscheidt KG, Berlin und München, 1979
- [10] *Hyperwave WWW Homepage*:
<http://www.hyperwave.com>
- [11] *Hyperwave Programmer's Guide*, Hyperwave Information Management GmbH, Munich, May 1998
<http://www.hyperwave.de/program>
- [12] *Hyperwave Administrator's Guide*, Hyperwave Information Management GmbH, Munich, May 1998
- [13] *Hyperwave Reference Guide*, Hyperwave Information Management GmbH, Munich, May 1998

- [14] *Hyperwave API Definition Guide*, Hyperwave Information Management GmbH, Munich, May 1998
<http://www.hyperwave.de/apidef>
- [15] *Hyperwave Technical White Paper*, Hyperwave Information Management GmbH, Munich, May 1998
<http://www.hyperwave.com/whitepaper>
- [16] *Hyperwave User's Guide*, Hyperwave Information Management GmbH, Munich, May 1998
<http://www.hyperwave.de/user>
- [17] *Hyperwave List of Placeholders*:
<http://www.hyperwave.de/ref/listofplace>
- [18] *Hyperwave HW Tools Manual Pages*:
http://www.hyperwave.de/hw_tools
- [19] *ISCN WWW Homepage*:
<http://www.iscn.ie>
- [20] Jenner Michael G., *Software Quality Management and ISO 9001*, John Wiley & Sons, Inc., New York, 1995
- [21] Kirchmair Gerolf, *Telearbeit, Realität und Zukunft*, ÖGB Verlag, Wien 1996
- [22] Kolm Paul, Kral-Bast Claudia, Reifinger Ingrid, Tallfuss Werner, *Telearbeit von A-Z*, ÖGB Verlag, Wien 1996
- [23] Langhoff June, *Telecom Made Easy*, Aegis Publishing Group, 1995
- [24] Lucas Manfred, *Telearbeit: Strategien für die Zukunft Ihres Unternehmens*, ECON, Düsseldorf, 1997
- [25] Maurer Hermann, *Hyperwave - The Next Generation Web Solution*, Addison Wesley Longman, 1996
- [26] *Perl WWW Portal*:
<http://www.perl.com>
- [27] Messnarz Richard, Stubenrauch Robert, Melcher Martin, Bernhard Rainer, *NQA - Experience with Integrated Teamwork and Network based Quality Assurance*, Dublin, Ireland, 1998
- [28] Messnarz Richard, *NQA Hyperwave Version 1.0, User Requirements Document*, Dublin, Ireland, 1998

- [29] Nilles Jack M., *Making telecommuting happen*, Van Nostrand Reinhold, New York, 1994
- [30] Schneier Bruce, *Applied Cryptography*, John Wiley & Sons, Inc., New York, 1996
- [31] Schwartz Randal L., Christiansen Tom, *Learning Perl, Second Edition*, O'Reilly & Associates, Inc., Cambridge, 1997
- [32] *Stichwort Telearbeit, Neue Arbeitformen bestimmen die Zukunft*, Tagungsband, Bundesministerium für Arbeit, Gesundheit und Soziales, Wien, 1997
- [33] *Televillage 'Bruck an der Leitung'*:
<http://obelix.soe.oeaw.ac.at/telework>
- [34] Wall Lary, Christiansen Tom, Schwartz Randal L., *Programming Perl, Second Edition*, O'Reilly & Associates, Inc., Cambridge, 1996
- [35] *Xanadu WWW Homepage*:
<http://www.xanadu.com.au>