Hyper-G
Specification of Requirements

Frank Kappe, Hermann Maurer

IICM, Institutes for Information Processing Graz,
Graz University of Technology

Ivan Tomek

Jodrey School of Computer Science,
Acadia University, Wolfville, Nova Scotia

Abstract

Hyper-G is the name of an ambitious Hypermedia project[1] currently being developed as a joint effort by a number of institutes of the IIG (Institutes for Information-Processing Graz) of the Technical University of Graz[2] and the Austrian Computer Society.

Study of modern hypermedia systems, information systems, and user interfaces lead to a number of ideas, features, and examples of applications of Hyper-G. They were condensed, put into a logical relationship, and now form a framework of requirements that is contained in this report.

The requirements may also be seen as a description of Hyper-G's features. Care has been taken to isolate requirements from implementation details, design decisions, examples and applications of Hyper-G. For such issues, the reader is referred to e.g. [8, 17, 19] and [20].

[1]Funding of this project by the Austrian Ministry of Science, grant 613.531/5-26/90, is gratefully acknowledged.

[2]Yes, the 'G' in Hyper-G stands for Graz.

Contents

1 Introduction

The forthcoming sections contain a hierarchy of requirements for Hyper-G. They are structured in five groups: *General Requirements, User-related Requirements, Author-related Requirements, System Requirements* and *Implementation Requirements*. For ease of reference, a unique label is assigned to each requirement in such a way that, if requirement *a* is refined by requirement *b*, the label of *a* is a prefix of the label of *b*.

2 General Requirements

Req. G.1:	The system should allow access to all kinds of information one can think of (*Multimedia Axiom*).

Let us refine this to a set of more specific requirements:

Req. G.1.1:	Information is organized in *documents* of a specific *document type*. Document types include (but are not limited to):
Req. G.1.1.1:	Textual Information
Req. G.1.1.2:	Technical Drawings
Req. G.1.1.3:	Raster Images
Req. G.1.1.4:	Digitized Sound
Req. G.1.1.5:	Digital Movies
Req. G.1.1.6:	Animation
Req. G.1.1.7:	Maps
Req. G.1.1.8:	Electronic Mail, Bulletin Boards, Computer Conferencing
Req. G.1.1.9:	Dialogs (e.g. Menus, Forms)

A system able to handle above document types is generally referred to as a *multimedia system*. To put it simple, above requirements state that Hyper-G is such a multimedia system. Above list is by no means complete; e.g., document types like synthesized voice and music may be added in the future.

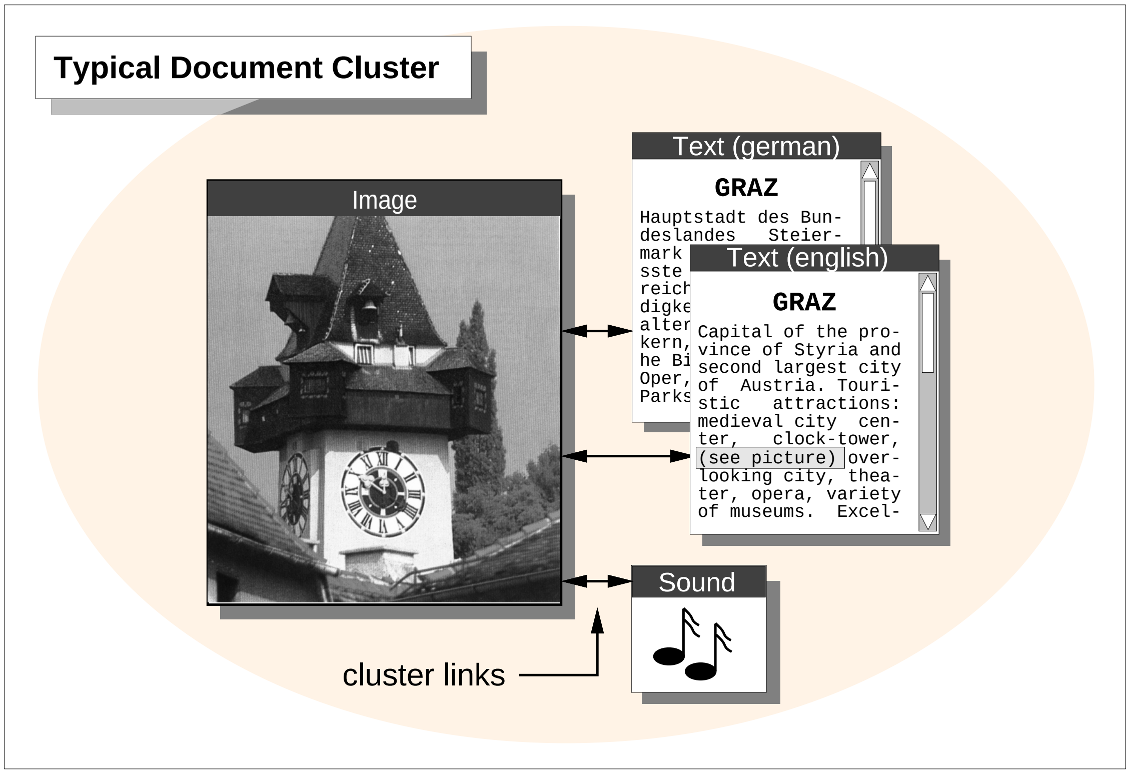
Basic Requirement **Req. G.1** introduces the notion of *documents*. These documents can be grouped into so-called *Document Clusters*.

Req. G.1.2:	*Document Clusters* are groups of *documents* that are semantically equivalent or related.

Examples of "semantically equivalent or related" documents are:

- A text document in German and the English translation.
- A text document and a sound document containing the voice of someone reading the text document.
- A picture and the description of the picture.

A special type of link (semantic link, see **Req. G.2.4**) is used to link the individual documents of a document cluster. While a discussion of advantages and disadvantages of document clusters may be found in [8], figure 1 illustrates an example.



GRAZ

Hauptstadt des Bundeslandes Steiermark
sste
reich
digke
alter
kern,
he Bi
Oper,
Parks

GRAZ

Capital of the province of Styria and second largest city of Austria. Touristic attractions: medieval city center, clock-tower, (see picture) overlooking city, theater, opera, variety of museums. Excel-

Figure 1: A typical Document Cluster

Req. G.1.3:	Documents, as well as links, collections, tours etc. are subclasses of the class *Hyper-G object.* All Hyper-G objects (and its subclasses) store attributes in addition to the information. These attributes include:
Req. G.1.3.1:	Author and User Group.
Req. G.1.3.2:	Creation/Modification/Access/Expiration time.
Req. G.1.3.3:	Language.
Req. G.1.3.4:	Rights for object's functions, for *author, group* and *world.*
Req. G.1.3.5:	Price for object's functions, for *group* and *world.*

Req. G.2: Documents (**Req. G.1.1**) may be cross-referenced and accessed by *links* (*Hypertext Axiom*).

This means that Hyper-G adopts the well-known hypertext paradigm. Requirements **Req. G.1** together with **Req. G.2** state that Hyper-G is what is known as a *hypermedia system*.

Req. G.2.1: The user is offered a visual representation of the link, called *anchor*. The user may 'activate' the link by activating the corresponding anchor.

Of course, the actual meaning of 'activate' depends on the user interface. Typically, the user will point to the anchor using some pointing device and then activate it by pressing a button. However, other means of activation are possible with unorthodox user interfaces [8, 10, 14, 20].

Req. G.2.1.1: Anchors belong to links rather than documents. Creation, modification or deletion of anchors is not regarded as document modification.

This implies that even users that have no right to modify a document may create links (with anchors) from/to that document, if they are granted the right to create links (*annotation*).

Req. G.2.1.2: The visual appearance of an anchor depends on the destination document type, and on whether the destination document has already been visited during the active session.

This significantly reduces the number of unwanted re-visits of the same document in a session. Also, this feature is used to lead the user to 'rare information', such as video clips, animation, sound, and images, and is most important in the early phase of Hyper-G. At a later stage, e.g., when high-quality images are not so rare any more, it may become partially obsolete.

Sometimes, the user would like to have the ability to trace links backward to the source when being at the target ("Show me the documents with links to the current document"). This 'associative search' [8] may be considered the opposite of conventional (forward) link following.

Hyper-G will support both forward and backward (associative) link following. Note that there are no new links or link types involved; rather the user is given the choice of how to use them.

Req. G.2.2: Links are directed, i.e. there are *source* and *destination* anchors. However, Hyper-G supports forward and backward link following.

Req. G.2.2.1: When forward following a link (i.e. source to destination), on activating the source anchor, the system will activate the destination document in such a way that the destination anchor is visible (if possible).

An example would be a paragraph in a destination text document that is scrolled so that at least the beginning of the paragraph is visible.

Req. G.2.2.2: When backward following a link (i.e. destination to source), on activating the destination anchor, the system will activate the source document in such a way that the source anchor is visible (if possible).

An example of this would be a part of a map that is scrolled such that the source anchor linking to the destination document (a more detailed map) is centered in the display window.

Req. G.2.3: Every document has a *default anchor* associated with it. The default anchor covers the whole document and can be used for both the source and the destination of links.

The default anchor is used to link whole documents, e.g. as destination or when using semantic links (see **Req. G.2.4**).

Links can be classified according to a number of classification schemes. The following requirements specify the types of links distinguished by Hyper-G.

Req. G.2.4: Hyper-G supports *semantic links* and *referential links*. Semantic links are used to tightly couple the individual documents of document clusters.

Semantic links are always user defined, as the semantic relation between two documents can not reasonably be detected automatically. However, referential links may be further subdivided:

Req. G.2.4.1:	Hyper-G supports *user defined* links as well as *automatically generated* links.

Req. G.2.4.1.1:	Automatically generated links may be pre-generated (*static*) or generated on demand (*dynamic*).

Dynamic links are generated using hidden database key queries (**Req. U.3.4.1**). The user may specify a list of *active databases* that are searched (*link filtering*).

Note that according to above definition user-defined links as well as semantic links are always static. Figure 2 gives an overview of the basic link types supported by Hyper-G. The set of link types is extensible, which may be necessary for certain applications such as *Hyper-Animation* [11].

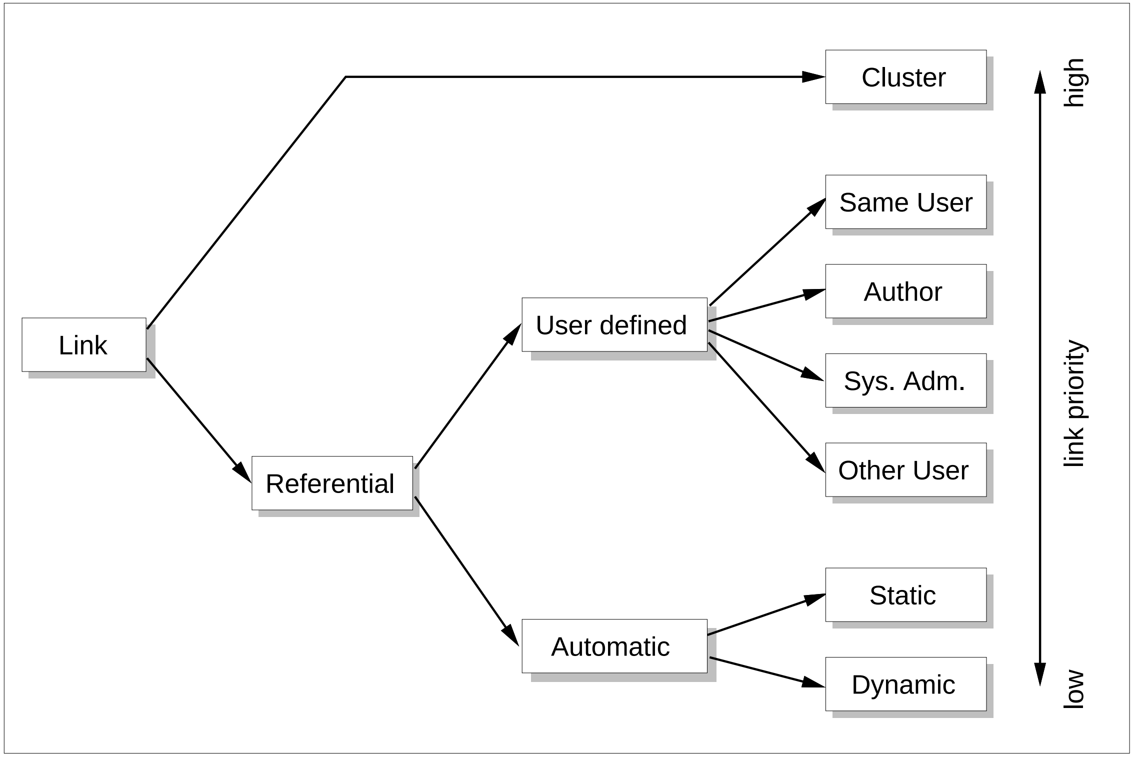


Figure 2: Basic Link Types and their Relative Priority

Req. G.2.5: A *Link Priority* is associated with any Hyper-G link. The link priority is used to reduce the number of links presented to the user.

Of course, the actual meaning of "priority" will vary with the user interface. In particular, a user interface may choose to always follow semantic links (highest priority) without user interference, i.e. to display a whole document cluster at a time. Also, a user interface may choose not to generate dynamic links (lowest priority) when there are any higher-priority links available. Figure 2 shows the relative priority assigned to the basic link types of Hyper-G.

Req. G.2.6: *Active Anchors* have an associated piece of code that is executed when the anchor is activated. Hyper-G supports a number of pre-defined active anchors for dialog documents and special effects.

Active anchors are most useful with dialog documents (e.g. answer judging) and for special effects (e.g. a special kind of destination anchor might 'fade out' the source document and 'fade in' the destination document.).

Req. G.3: Hyper-G provides access to very large amounts of data.

As stated in [4], Hyper-G is targeted to mega-quantities of documents and terra-quantities of bytes. This leads to interesting problems not only for system design, but – more important – for the users and the authors of such a large system [9]. An immediate consequence is

Req. G.3.1: Documents and links are maintained automatically.

Link maintenance involves such problems as deleting all links from or to a document that is to be deleted, in order to avoid dangling references and to ensure database integrity. Also, the system should assist the system administration in protecting the database against obsolete or outdated documents and/or links.

Req. G.3.1.1: All Hyper-G objects have an *Expiration Date* associated with them. Hyper-G objects age. When the expiration date is reached, the object may be deleted.

This is to protect the database from outdated or obsolete information. A typical expiration time of a user-supplied document might be six months. Two months before expiration, the user (i.e. the author of the document) is notified (by means of e-mail) and may extend this period. If no response is received, the object may be deleted (and with it all associated links).

Of course, some objects may be given a very long life time by the system administration. Also, additional factors such as usage count and time of last access may be used to decide on whether an object is to be deleted.

Req. G.3.2: The mayority of the links is generated automatically.

Especially for text documents, a number of links can be generated automatically either when inserting the document into the database, or at runtime.

Req. G.3.2.1: Whenever a new document is inserted, (static) links are generated automatically.

This is especially useful for annotations: A new annotation of a user is automatically linked to other documents, possibly annotations of other users. An off-line communication between users and authors starts.

A user related consequence (**Req. U.3**) of **Req. G.3** is given in section 3 (page 10).

An author related consequence (**Req. A.2**) of **Req. G.3** is found in section 4 (page 18).

A system related consequence (**Req. S.3**) of **Req. G.3** is found in section 5 (page 18).

Req. G.4: The system should be easily extensible.

This is especially true for the set of supported document types and user interfaces (**Req. U.1.1**) and has some effect on the implementation model [8].

3 User-related Requirements

Req. U.1:	The system should be usable by untrained users as well as expert users.

In general, Hyper-G is likely to see a number of scenarios [8, 10]. One is a university information system based on Hyper-G where users are expected to communicate with computers on a regular basis, another one is a so-called *viewseum* [14, 20] available to the general public. Hyper-G should adapt to the needs of all user groups.

While expert users may be expected to be familiar with the user interfaces of today's computers, the casual user must not. Therefore, Hyper-G can be operated using a number of different user interfaces.

Req. U.1.1:	Users control the system with user interface devices suited to their background knowledge and the application.

Interface Device	User Type	Typical Application
Keyboard	Home User	Remote terminal access
Keyboard & Mouse	Expert User	University Information System
Touch-Screen	Visitor	Viewseum
Home-Trainer	Visitor	HOTACT [16]
3D input/output	Visitor	Advanced 3D-Applications

Table 1: Typical User Interfaces, Users and Applications

Table 1 shows some typical combinations of user interface device, user type, and application.

Req. U.2:	The system should offer a choice of user languages.

This not only requires the user interface to work with a number of languages, but also means support of multi-lingual documents (e.g. translations of the same document provided by the author, or simple computer translations).

Req. U.2.1:	The user may specify an ordered list of language preferences.

Req. U.2.2:	A document cluster may include semantically equivalent documents that differ only in the *Language* attribute. A user interface may use this information and the user's preferences to select which document to show.

Req. U.2.3:	The user interface of the runtime system is available in a number of languages.

Req. U.2.4:	The language strategy of the user interface can be temporarily overridden.

Req. U.3:	The user is supported by high-level navigation tools.

Above requirement may also be seen as a consequence of **Req. G.3** (large number of documents). High-level navigation tools include use of user interface metaphors, database queries, collections, tours and scripts. These concepts are discussed in some detail in [8].

Req. U.3.1:	A variety of user interface metaphors is offered, including:
Req. U.3.1.1:	*Simple Hypermedia* [6]
Req. U.3.1.2:	*Desk-Top* (Xerox' *Star* [7], Apple Macintosh)
Req. U.3.1.3:	*Stack* (HyperCard [21])
Req. U.3.1.4:	*Book* [1]
Req. U.3.1.5:	*Holiday Travel* [5]
Req. U.3.1.6:	*Library* [4]

The list of supported user interface metaphors is extensible, as the user interface is built upon an underlying 'Hyper-G Kernel' that supports common functions.

Req. U.3.1.7:	Most User Interface Metaphors support simple navigation functions, such as *undo, redo, help, drop/goto book-mark, show copyright, show links, show annotations, show collection, show tours...*

Other means to aid the user's navigation are collections:

Req. U.3.2:	Information is structured in *Collections*. A collection is a set of documents or other collections ('subcollections').

This recursive definition yields a hierarchical structure of collections:

Req. U.3.2.1:	The collection hierarchy is an acyclic directed graph. There are a number of 'root' collections.

To put it simple, 'acyclic directed graph' means that the terms 'ascending' and 'descending' the structure are well defined like in a tree, however, a collection may be a subcollection of a number of collections. For example, the collection 'Biochemistry' might be a subcollection of collection 'Chemistry' as well as of 'Biology' (see figure 3).

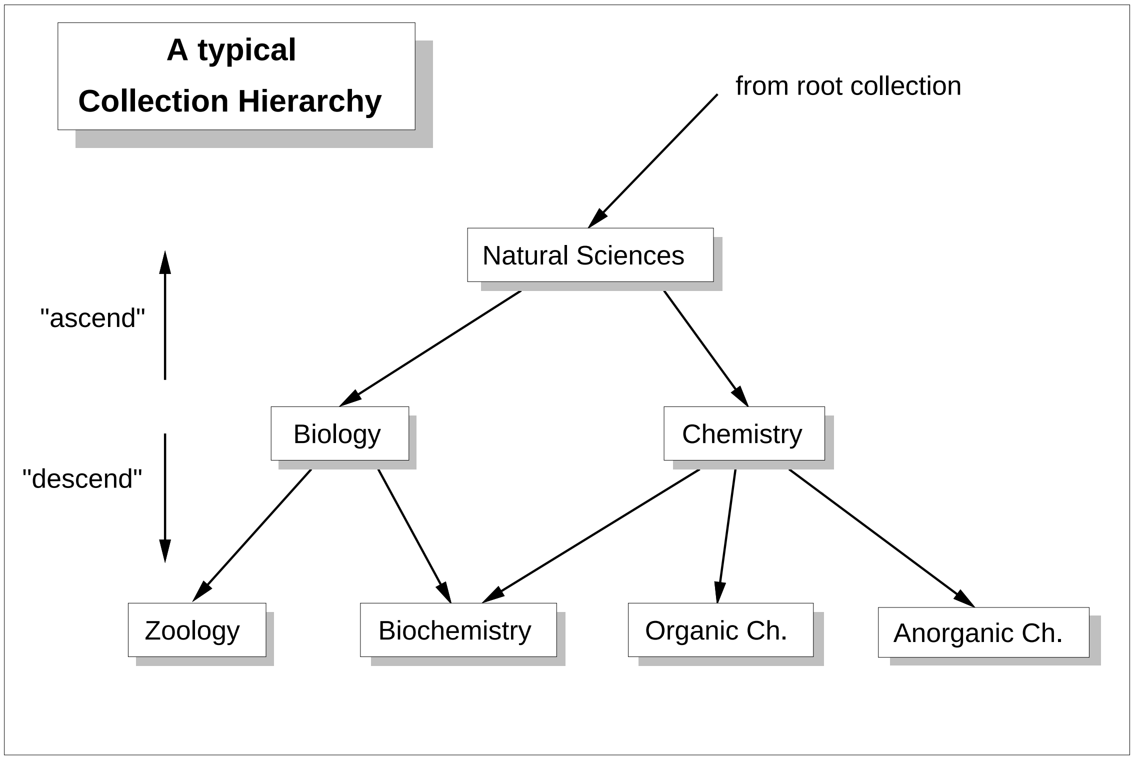


Figure 3: A typical Collection Hierarchy

Req. U.3.2.2:	Any document in the system is a member of at least one collection. The user may find the collection(s) a document belongs to, get a graphical representation of the collection hierarchy at that point, 'ascend' or 'descend' the hierarchy, and find other documents belonging to the collection.

Note that because of the logical definition of the collection hierarchy, and because of the relatively small number of collections, graphical browsing of the collection hierarchy seems feasible.

In addition, the user may go on "guided tours":

Req. U.3.3: *Guided Tours* allow to traverse the information network in a way specified by the author (of the tour). However, tours do not have to be a linear list of document clusters to be visited, but may utilize the full flexibility of the hypermedia system.

Req. U.3.3.1: Document clusters may be visited by a number of tours. When the user follows tour a and visits a document cluster that is also visited by tour b, the user may switch tours at this point.

That is, the user can get a list of the tours visiting that document cluster, and may choose a tour to follow. However, at some later point, the user should have the opportunity to continue the original tour. It turns out that requirement **Req. U.3.3.1** leads to the concept of labeled links, i.e. links that are labeled with the tour name [8].

Within tours, advanced navigation facilities are possible. Especially in the case of educational tours the user should be supported in making sure that nothing has been overlooked:

Req. U.3.3.2: The user is offered information on what parts of the tour have already be seen and what parts have not.

In addition to the database queries that are made behind the scenes when generating dynamic links, Hyper-G offers user controlled database queries:

Req. U.3.4: Hyper-G supports *Database Queries* to allow fast access to relevant information.

Two types of queries may be distinguished:

Req. U.3.4.1: *Key Queries* allow for prefix search in the database(s).

This type of query is most useful at the beginning of a session, when the user wants to gain an overview of what is in the database(s) related to a given topic (keyword). Key queries may be combined by 'and', 'or' and 'not' operators.

Req. U.3.4.1.1: The user may specify a list of *active databases*. Only these databases are searched.

This not only increases speed, but also reduces the number of unwanted links (*link filtering*). Note that dynamic links are done by key queries, too, so that the list of active databases also applies to the generation of dynamic links.

Req. U.3.4.1.2: The system allows for *inexact match* key queries.

This may be done in two ways:

- Either the user may specify a kind of regular expression to be searched for, or
- The system itself tolerates a certain amount of spelling errors (e.g. "naboleon" should find "Napoleon").

Hyper-G will offer both approaches. In addition, the use of synonym databases and hierarchical synonym classification schemes will be investigated.

Req. U.3.4.2: *Form Queries* are more flexible, but collection-specific queries. The user is allowed to fill out a form with a number of attributes that may be combined to perform complex queries. Forms may be filled out iteratively, showing after each incremental step the quantity of information found and hints on how to continue.

The user interface for database queries (especially form queries) is most easily made by dialog documents with active anchors.

Req. U.4: The system is configurable on a per-user basis. It remembers a user's preferences between sessions.

This means that users may configure the system to adapt to their special needs (e.g., language preferences (see **Req. U.2**).). Moreover, this configuration is retained by the system until the next session. Of course, this implies that the system is able to distinguish between users by some login procedure and keeps configuration files. It also suggests use of user groups, access rights and so on.

Req. U.4.1:	The system maintains a user profile, describing:
Req. U.4.1.1:	Security Data (User name, user group(s), account number, password, access rights
Req. U.4.1.2:	The user's system entry point (login menu, starting collection)
Req. U.4.1.3:	The user's language preferences
Req. U.4.1.4:	Preferred user interface metaphor
Req. U.4.1.5:	Preferred user interface metaphor's options
Req. U.4.1.6:	User logging (full/system/off)
Req. U.4.1.7:	Default active databases for queries
Req. U.4.1.8:	Whether user may change profile

A few explanations are necessary:

The *security data* is used to identify the user. However, the system may also be used anonymously (**Req. U.6**), in which case the default user profile of the user's terminal is applied. Of course, anonymous users cannot alter the configuration[3].

In principle, the *user's system entry point* might be any Hyper-G object. However, to define a root-level collection (remember, the collection hierarchy may have more than one root) or a dialog document that performs a database query would be reasonable.

The *user's language preference* is an ordered list of languages (**Req. U.2.1**). The selection 'Hungarian' – 'German' – 'English', for example, means that if a Hungarian version of a document is available, show it, else show the German version, else the English one. If none of the user's list is available, the system may display an appropriate error message, or display whatever it has. Also, the user interface language is to be set to the language of the first choice, if possible.

The *preferred user interface metaphor's options* are further options specific to the *preferred user interface metaphor*. For example, the user may specify to prefer short versions instead of long versions of text or voice documents, if semantically equivalent versions are available in a document cluster; or to prefer hearing a voice document rather than reading a semantically equivalent text document. In addition, preferred document managers [8] that should be used to display certain document types can be specified. An interface metaphor might even allow to configure the exact layout (position, size) of the various windows.

User logging (**Req. U.5.1**) can be turned to 'full' (full user state logged), 'system' (only the items necessary for **Req. U.5.1.2** logged) or 'off'.

[3]However, semi-identified and anonymously identified users potentially can (**Req. U.6.1** and **Req. U.6.2**).

The list of *active databases* is actually a list of active collections, to which queries are to be passed.

Req. U.5: At any time, the user may return to a previous user state.

This applies to all the previous states of an active session, which yields sort of an *undo* function. As a special case, when entering a new session, the user may return to where the previous session was left. **Req. U.5** requires the system to remember user states, keep user logs etc.:

Req. U.5.1: For every user, the system maintains a *user log*. The log stores a sequence of *user states*.

The log enables the system to return to a previous state (**Req. U.5**). The user state records all information necessary to do this. In addition, some graphical representation of the user's path through the information network may be displayed (*history*).

Req. U.5.1.1: Identified, semi-identified and anonymously identified users may also navigate backwards to the user states of previous sessions.

This is done by having *persistent* user logs, i.e. by saving the user log across sessions. Of course, this feature is not available for anonymous users. User identification modes are discussed under **Req. U.6**.

Req. U.5.1.2: User logs may be used by authors and/or system administrators to maintain the database.

Sometimes, users get stuck because of misleading or missing links, etc. The user logs may be used to find such weaknesses of the database. Not all the information of the user state is required to be logged ('User logging' set to 'system' would be sufficient (**Req. U.4.1.6**).). For example, only unsuccessful database queries need to be logged for this purpose. In particular, anonymous users may be traced also.

Req. U.6: The system can be used anonymously.

An information system that is to be used by a large number of users on a regular basis raises a data security problem. If users are known by name, the system might be used to find out who looks for certain information, monitoring users working time, etc. To avoid such problems from the beginning on, users may login anonymously, as in the Austrian videotex system [15]. The modes *semi-identified* and *anonymously identified* still allow to perform most operations of the system:

Req. U.6.1: The system may be used in *Semi-Identified Mode*. In this mode, the real identity of the user is known to the system, but not to other users.

Req. U.6.2: The system may be used in *Anonymously Identified Mode*. In this mode, the real identity of the user is not known to the system. However, the user's pseudo-identity persists across sessions.

Advantages and disadvantages of the four user identification modes are discussed in more detail in [8].

Req. U.7: Certain system features and information items are chargeable.

For example, printing a screen dump or saving it to disk may cost some (small) amount of money. In addition, information providers may charge something for the access to certain pieces of information, remote databases, etc. In conjunction with **Req. U.6** this implies that a rather tricky charging mechanism has to be implemented.

Req. U.7.1: Any Hyper-G object may be priced. The price is specified by the author of the object (**Req. G.1.3.5**). The system automatically tells the user in advance the price of the object to be accessed, if its greater than zero.

In order to charge both identified and anonymous users, Hyper-G supports two charging schemes:

Req. U.7.2: Hyper-G maintains user accounts for identified, semi-identified and anonymously identified users. Whenever a priced object is shown, the object's costs are deducted from the user's account.

In theory, objects may also have a negative price, in which case the amount is added to the user's account. Identified and semi-identified users may be allowed to overdraw their account, and will receive an invoice at regular intervals.

Req. U.7.3:	At special terminals, anonymous users may be charged also.

This is done by letting the users (anonymously) buy tokens of a certain value, and then deduct from that value until the remaining value on the token reaches zero. Tokens will typically be credit card-like magnetic cards or IC-cards. Of course, anonymous and anonymously identified users are not allowed to overdraw their account.

4 Author-related Requirements

Req. A.1:	Any user is a potential author.

The database may be manipulated by potentially every user, either by manipulating documents or links (*annotation*). Of course, access rights control the user's ability to change existing data items or create new ones.

Req. A.2:	The author is supported by high-level information editing tools.

These information editing tools divide into document preparation tools and link editing tools. They are not part of the Hyper-G runtime system and discussed in [8].

5 System Requirements

This section describes requirements related to system administration.

Req. S.1:	A number of *terminals* is attached to the system. On any terminal a Hyper-G *session* may be started. The capabilities of the terminals may differ.

That is, the terminals are not all the same. There may be terminals with special user interfaces, communication features, hard/softcopy capabilities etc. As a consequence, Hyper-G must not rely on the presence of specific terminal facilities.

Req. S.1.1:	For each terminal, there is a terminal profile, describing:
Req. S.1.1.1:	Terminal facilities, such as:
Req. S.1.1.1.1:	Input Devices
Req. S.1.1.1.2:	Voice I/O capability
Req. S.1.1.1.3:	Digital Sound I/O capability
Req. S.1.1.1.4:	Digital Video I/O capability
Req. S.1.1.1.5:	Hardcopy capability
Req. S.1.1.1.6:	Softcopy (diskette) capability
Req. S.1.1.1.7:	Communication capabilities (E-Mail, conferencing)
Req. S.1.1.1.8:	Charging of anonymous users
Req. S.1.1.2:	Security information, such as:
Req. S.1.1.2.1:	The terminal's access rights (operations allowed on this terminal)
Req. S.1.1.2.2:	The rights of anonymous users at this terminal
Req. S.1.1.3:	The prices of specific terminal facilities

Again, a few explanations are in order:

Useful combinations of *Input Devices* are listed in table 1. Some terminals may be equipped with hardcopy or softcopy devices ('softcopy' means that the user/visitor can take a diskette back home with selected documents on it).

Hard- and Softcopy facilities are usually priced (**Req. S.1.1.3**). The charging of anonymous users may be done using tokens, coins, or terminals equipped with magnetic or IC-Card readers or similar devices (**Req. U.7.3**).

Req. S.2:	Access rights control whether a given user on a given terminal may perform a certain function on an object.

We can further refine this concept:

Req. S.2.1:	The right of a certain user on a certain terminal to perform a specific function of certain Hyper-G object is deduced from the object's protection bits (for *author, group* and *world*, **Req. G.1.3.4**), the user's rights **Req. U.4.1.1** and the terminal's access rights **Req. S.1.1.2.1**. Anonymous users get the terminal's default right (**Req. S.1.1.2.2**). Users of user group *system* may override rights.

To be specific, the user's right to perform a certain operation is calculated in the following way: For any operation (e.g. read, write, print, copy, link to, link from) there is an associated bit, say bit number i. If it's set to '1', this means "allowed", a '0' means "not allowed". These *protection bits* are stored together with the following entities:

- Any Hyper-G object holds protection bits for:
  - Author (*object.author.bits*)
  - Group (*object.group.bits*)
  - World (*object.world.bits*)
- Protection bits are associated with every user (*user.bits*).
- Access rights are also associated with terminals (*terminal.bits*).

In addition, a user may be a member of a number of user groups ($user \in G_i$). Members of group *system* may override any protection bits with exception of *terminal.bits*, i.e. system administration can only be done from a subset of the terminals as an additional security measure. Also, if for example a terminal does not support printing, it does not make too much sense to insist on your right to print something. With these prerequisites in mind, the protection algorithm looks like shown in figure 4.

This protection scheme is somewhat similar to the UNIX approach, but with the following modifications:

- There are terminal protection bits. This serves for two reasons:
  - The terminal's protection bits are taken into account for all users, including the system administration. This means that the system administrators may login at any terminal, however, the specific rights needed to perform system administration (e.g. the right to delete objects) are only granted at specific terminals.
  - The protection bits also specify certain functions to do with the document that are not available at all terminals, e.g. hardcopying. A terminal that does not support that specific function would set the according protection bit to "not allowed", overriding any different settings of user and/or document protection bits.

```
if $user \in system$
   then $bits := terminal.bits$
   else if $user = object.author$
            then $bits := object.author.bits$ and $user.bits$ and $terminal.bits$
            else if $user \in object.group$
                     then $bits := object.group.bits$ and $user.bits$ and $terminal.bits$
                     else $bits := object.world.bits$ and $user.bits$ and $terminal.bits$

if ($bits$ and $2^i$) $\neq 0$
   then /* function #i allowed */
   else /* function #i not allowed */
```

Figure 4: Hyper-G Protection Scheme

- The users themselves have protection bits associated with them. This may be used to specifically deny certain rights from a user. For example, for some reason a user may be (temporarily) denied the right to annotate something, without the need to change the rights of the objects or modify user groups.
- The system may be used anonymously, in which case users get their rights from the terminal profile (**Req. S.1.1.2.2**).

As stated in **Req. G.1.3.5**, a Hyper-G object has a price associated with a particular function (e.g. read, hardcopy, softcopy, link to, link from) for group and world (of course, if the corresponding function is prohibited by the associated protection bit, the 'price tag' is meaningless; also, the price for the author is always zero). Calculation of the price to be paid by a specific user is done similar to the protection scheme described above. If the user is a member of the specified user group, the 'group' price tag is used, else the 'world' price is to be paid.

A typical application would be to specify that linking-to an object (e.g. using it in a tour) costs amount w for anybody except users of group *mybestfriends*, which pay amount g.

It should be noted that variable-length lists have been avoided in the design of Hyper-G object's attributes, because of their implementation overhead associated with database design. As a consequence, in order to distinguish between a number of user groups, more database objects pointing to the same piece of information (e.g. document) have to be established. In our above example, in order to let users of user group *mysecondbestfriends* pay amount $g1$, a second Hyper-G object with 'group' set to *mysecondbestfriends* has to be created.

Also, the terminal has prices associated with certain functions, which are added to the price determined by the object (e.g., the price for a hardcopy is calculated by the price the author specified plus the price the terminal's price for a copy). However, the amount collected for using the Hyper-G object is transferred to the author, while the amount collected by the terminal is for the system (administration).

So, as a conclusion:

Req. S.2.2:	The price to be paid for a certain function on a certain object at a certain terminal is determined by the object's price tags (for *group* or *world*, **Req. G.1.3.5**) and the terminal's price for that particular function.

Req. S.3:	The system should avoid uncontrolled growth of unimportant or outdated information.

The system administration should be supported in finding such information and deleting it at regular intervals.

6 Implementation Requirements

Req. I.1:	Access to information should be reasonably fast.

This requirement is somehow contradictory to **Req. G.3** (large amount of data). However, it is well-known that a response time larger than about 250 milliseconds confuses users[2]. A lower response-time variance seems to be more important than absolute response times [3]. Also, users are willing to accept, e.g., to wait a few seconds for a database query, if the system supplies them with immediate feedback of acceptance of the query. We have seen systems (e.g. videotex [12, 13]) whose acceptance by the public has been dampened by less than optimal response time.

Above requirement has significant impact on the design of the databases (speed vs. space tradeoffs) and the computers used. From the requirements of previous sections it is quite obvious that Hyper-G will use a set of databases to store its documents, links, anchors etc. Because Hyper-G's data items are structured in an object-oriented way (e.g. they are all derived from a class 'Hyper-G object') it would be best to use object-oriented databases to do the job. Unfortunately, only experimental object-oriented database systems suitable for our purposes exist at this time. Also, a standard relational database system would turn out to be too slow to handle the huge number of objects Hyper-G developed for. In addition, there are legal problems associated with the use of commercial database systems. Hence we decided to build our own (simple, but speedy) database management system customized to the special needs of Hyper-G.

In fact, Hyper-G will use two kinds of databases (some arguments for this split are discussed in [8]):

Req. I.1.1:	The *Keys & Attributes Database (KAD)* holds the keys and attributes of all Hyper-G objects. In order to gain speed, the database is split into specialized databases holding:
Req. I.1.1.1:	documents
Req. I.1.1.2:	links
Req. I.1.1.3:	anchors
Req. I.1.1.4:	collections
Req. I.1.1.5:	tours

The attributes stored with each object are listed under **Req. G.1.3**. The keys are used to search for the object using the simple key queries described under **Req. U.3.1.4** as well as to link the links to the anchors and the anchors to the documents. The *KAD* will be implemented using simple text files with inverted indices. It does not contain documents, only pointers to the documents are stored.

The documents themselves are stored as part of a collection database. While sometimes this collection database will just contain pointers to the actual documents (e.g. to raster images), text documents (e.g. the entries of an encyclopedia) might be grouped together in one file, as this allows a collection-wide full text search that is needed for associative dynamic links and form queries. Also, the graphical appearance of anchors is stored together with the documents, while their attributes are stored in the KAD.

Req. I.1.2:	The documents and anchors are stored in collection databases, while their keys and attribute are held in the KAD. The actual layout of the databases will vary with the type of collection. The collection supports collection-wide full-text search (if applicable) and/or form queries.

Note that the notion that the collection itself performs this actions and the exact layout of the database is not to be known outside conforms to the object-oriented design principles of encapsulation and data hiding.

Req. I.1.3:	The databases themselves ensure database integrity.

For example, if a document is to be deleted, also the anchors of the document as well as the links to/from that anchors are deleted. This facilitates database maintenance as required by **Req. G.3.1** and **Req. S.3**.

Sometimes it will not be feasible to keep all the data in Hyper-G's local databases. Therefore, Hyper-G will support access to remote databases (e.g. official airline guide, phone books, any kind of data that keeps changing), but this access will be hidden from the user. In particular, that kind of data will be embedded into the user interface metaphor and will be accessible by links.

Req. I.2:	Hyper-G supports access to remote databases. Access to such databases shall be hidden from the user.

That means that users is confronted with a consistent user interface metaphor throughout the system, so that they would not be able to tell whether an information item is retrieved from a local or remote database (except, perhaps, because of a delay when retrieving remote information).

References

[1] Benest I. D. : "A Hypertext System with Controlled Hype". In *Proc. of the Hypertext II Conference, York*, 1989.

[2] Card S. K., Morgan T. P., and Newell A. : *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates, Hillsdale, N.J., London, 1983.

[3] Carson G. S. : "Graphics, Networking, and Distributed Computing". In *Proc. Workshop on Graphics and Communications (ESPRIT Project 2463 - ARGOSI), Breuberg, Germany*, Springer, October 1990.

[4] Davies G., Maurer H., and Preece J. : *Presentation Metaphors for a very large Hypermedia System.* IIG Report 282, IIG, Graz University of Technology, Austria, April 1990. To appear in: Journal of Micro Computer Applications 2 (1991).

[5] Hammond N. and Allison L. : "The Travel Metaphor as Design Principle and Training Aid for Navigating Around Complex Systems". In Diaper D. and Winder R. (editors), *People and Computers III*, Cambridge university Press, 1987.

[6] Hütter M. : *Design und Implementierung eines Hypermedia Basis Systems.* Master's thesis, Technical University Graz, Austria, June 1991. In German.

[7] Johnson J., Roberts T. L., Verplank W., Smith D. C., Irby C., Beard M., and Mackey K. : "The Xerox Star: A Retrospective". *IEEE Computer*, 22(9):11–26, September 1989.

[8] Kappe F. : *Aspects of a Modern Multi-Media Information System.* PhD thesis, Technical University Graz, Austria, 1991. Will also be published as IIG Technical Report.

[9] Kappe F. : "Spezielle Eigenschaften großer Hypermedia-Systeme". In *Proc. of Hypertext/Hypermedia '91, Graz, Austria*, Springer, May 1991. In German. To appear.

[10] Kappe F. : "Unorthodoxe Anwendungen von Hypermedia-Systemen". In Wallmannsberger J. (editor), *Hypertext - State of the Art*, R. Oldenbourg, Vienna, Munich, 1991. In German. To appear.

[11] Kappe F. and Maurer H. : "Animation in Hyper-G - An Outline". In Haase V. and Zinterhof P. (editors), *Proc. Future Trends in Information Technology '90, Salzburg, Austria*, pages 235–248, Austrian Computer Society, R. Oldenbourg, Vienna, Munich, September 1990.

[12] Maurer H. : *Bildschirmtext muß ein Erfolg werden.* IIG Report B42, IIG, Graz University of Technology, Austria, February 1984. In German.

[13] Maurer H. : *Bildschirmtextähnliche Systeme.* IIG Report B11, IIG, Graz University of Technology, Austria, 1981. In German.

[14] Maurer H. : *The Viewseum - An Introduction.* IIG Report 279, IIG, Graz University of Technology, Austria, April 1990.

[15] Maurer H., Roszenich N., and Sebestyen I. : "Videotex without Big Brother". *Electronic Publishing Review*, 4:201–214, 1984.

[16] Maurer H. and Soral G. : *HOTACT: HOmeTrainer And Computer Technology.* IIG Report 283, IIG, Graz University of Technology, Austria, August 1990.

[17] Maurer H. and Tomek I. : "Broadening the Scope of Hypermedia Principles". To appear in: Hypermedia 3 (1991).

[18] Maurer H. and Tomek I. : *Hypermedia Bibliography.* IIG Report 286, IIG, Graz University of Technology, Austria, November 1990. An updated version appears in: Journal of Micro Computer Applications 2 (1991).

[19] Maurer H. and Tomek I. : "Some Aspects of Hypermedia Systems and their Treatment in Hyper-G". *Wirtschaftsinformatik*, 32(2):187–196, April 1990.

[20] Maurer H. and Williams M. : "Hypermedia Systems and other Computer Support in Museums". To appear in: Journal of Micro Computer Applications 2 (1991).

[21] Williams G. : "HyperCard". *Byte*, 12(14):109–117, December 1987.