Published in the Proceedings of IMIA WG6, Geneva, May 1994

# A Terminology Server for Medical Language and Medical Information Systems

AL Rector[1] WD Solomon[1] WA Nowlan[2] TW Rush[2]

[1]Medical Informatics Group, Department of Computer Science, University of Manchester, Manchester, M13 9PL, UK

[2]Medical Products Group, Hewlett-Packard Ltd, Bristol, BS12 6QZ, UK

## Abstract

GALEN is developing a Terminology Server to support the development and integration of clinical systems through a range of key *terminological services,* built around a language-independent, re-usable, shared system of concepts - the CORE model. The focus is on supporting applications for medical records, clinical user interfaces and clinical information systems, but also includes systems for natural language understanding, clinical decision support, management of coding and classification schemes, and bibliographic retrieval. The Terminology Server integrates three modules: the Concept Module which implements the GRAIL formalism and manages the internal representation of concept entities, the Multilingual Module which manages the mapping of concept entities to natural language, and the Code Conversion Module which manages the mapping of concept entities to and from existing coding and classification schemes. The Terminology Server also provides external referencing to concept entities, coercion between data types, and makes its services available through a uniform applications programming interface. Taken together these services represent a new approach to the development of clinical systems and the sharing of medical knowledge.

**Keywords:** Terminology server, Natural language processing, Coding and classification schemes Electronic medical records, Knowledge representation

## 1. Introduction: The Idea of a 'Terminology Server'

Clinical practice centres on the care of patients by doctors, nurses, and other clinicians . Medical information should centre on the record of that care. There is a world-wide move towards 'patient-centred' information systems in which clinical information gathered by health care professionals during the process of patient care is both used to further that care and re-used to serve other functions within the health care systems.

If clinical information is to be re-used and shared, the basic concepts used to describe that care must be shared. Different specialised systems may organise those basic concepts differently for their own purposes, but the fundamental concepts must be common to all applications. In terms of classic data-modelling, we can imagine many different data models, but the meaning of the entities in those models— the meaning of 'the information that goes in the boxes on the modelling diagram' — must be shared. Such shared systems of concepts are increasingly known as 'ontologies' in the database and artificial intelligence communities.

The GALEN[1] project is funded by the European Commission as part of the AIM programme. GALEN's goal is to develop a 'Terminology Server' to manage language-independent shared systems of concepts for clinical applications. The Terminology Server will be a new type of integrating service for heterogeneous information systems. GALEN aims to demonstrate the feasibility and usefulness of such a Terminology Server:

- To provide infrastructure support for the development and integration of clinical systems.
- To provide a flexible, extensible basis for achieving 'coherence without uniformity' amongst the many different clinical information services required.
- To serve as an accessible repository of language-independent medical conceptual knowledge, and to map this repository to potentially many different natural languages.

[1] General Architecture for Languages Enclopædias and Nomenclatures in Medicine. The members of the GALEN consortium are: University of Manchester (UK, Coordinator), Hewlett-Packard Ltd (UK), Hôpital Cantonal Universitaire de Genève (Switzerland), Consiglio Nazionale delle Ricerche (Italy), University of Liverpool (UK), Katholieke Universiteit Neijmegen (Netherlands), University of Linköpking (Sweden), The Association of Finnish Local Authorities (Finland), The Finnish Technical Research Centre (Finland), GSF-Medis Institut, (Germany), Conser Systemi Avanzati (Italy)

- To convert between existing representations and coding schemes.
- To provide dynamically generated local nomenclatures or ‘coding schemes’ which are more comprehensive and thoroughly organised than can be held as a static structure or managed manually.

If computer systems are to play a significant role in clinical care, then formal ontologies which can be manipulated by computer systems are essential. Manual ‘coding systems’ or ‘controlled vocabularies’ interpreted by human users (largely on the basis of the natural language rubrics attached to the symbolic codes) are no longer sufficient. The difficulties of using even such massive efforts as the Unified Medical Language System [1], SNOMED-III [2] and the Read Codes [3] are all too apparent. Such systems are becoming too large to manage, but remain too small to contain the detail required to meet clinical requirements. Their organisation remains too limited to support acceptable clinical interfaces, and too rigid to support the variety and rapid evolution of clinical care.

To capture more detail and achieve greater organisation the meaning of the concepts must be captured not just in the rubrics but in the symbolic structure itself so that it can be manipulated computationally. Mechanisms are needed to encapsulate the resulting intrinsically variable descriptions into the fixed formats used by relational databases. These requirements have been extensively discussed elsewhere and we shall not review them further here [4-8].

Medicine is not alone in perceiving the need for shared terminology. Sharing and re-use of ‘ontologies’ is now a major growth area in many areas of information and knowledge based systems development [9-13]. However, medicine may be unique in its scale, its large and diverse body of professional users and sublanguages and in its common international effort to share knowledge based on extensive shared understanding of the domain. If there is not already a shared model of clinical medicine and disease, there is a vigorous international effort to create one, an effort largely motivated by clinical goals. GALEN is one response to the special needs of supporting these *clinical* efforts to share knowledge and practice. Others include [14-16].

Because of medicine’s distinctive situation, GALEN takes a distinctive approach to knowledge sharing. We shall return at the end of this paper to the relationship between our concept of a medical Terminology Server and other knowledge sharing efforts. In the next section we discuss GALEN’s approach to meeting needs of the clinical community; Section three provides a functional description of the GALEN Terminology Server. Sections four and five discuss the architecture of the Terminology Server and the special features of the GALEN modelling formalism — the GRAIL Kernel — which derive from the special clinical requirements for reuse and information sharing. The final section provides an overall discussion including questions of evaluation and maintenance.

## 2. GALEN

### 2.1 Fundamental proposition

The fundamental proposition of the GALEN project is that there is a terminological — or more properly, a conceptual — component of clinical language which can be usefully separated from other aspects of medical natural language processing, information modelling, knowledge based systems, and user interface design. GALEN contends that this conceptual component can be made largely independent of surface natural language characteristics. We suggest that this model is sufficiently strongly shared across clinical and linguistic groups to permit the development of an ‘interlingua’ [17] based on a single coherent COncept REference (CORE) model of medical concepts. We believe that such a CORE Model of medical concepts is the appropriate reference point for developing coherent collections of clinical applications which work together successfully and build on each others achievements.

Because access to the CORE Model and related information is a common and pervasive requirement for many applications, GALEN aims to encapsulate access to the CORE Model and related functions in a server — the ‘Terminology Server’. In a network environment the Terminology Server will both mediate amongst existing systems and act as a repository for terminology to facilitate developing new systems. We do not claim that such a ‘Terminology Server’ will solve all problems of mediation amongst existing systems or of building new systems. Indeed, one of the primary aims of GALEN is to modularise the overall task of building clinical systems. The goal of the Terminology Server is to relieve individual applications of technically difficult operations involving terminology, or conceptual knowledge of the domain. Our image is of groups of applications, developers and sites co-operating to develop and maintain one or more CORE Models which they all share and which support their joint efforts.

## 2.2 GRAIL

The modelling formalism in which the CORE Model is built is known as the GRAIL (**G**ALEN **R**epresentation **A**nd **I**ntegration **L**anguage) Kernel [18]. GRAIL is a compositional formalism — rather than having to enumerate all and only those clinical concepts that are available, the GRAIL modeller specifies elementary concept entities, and relations that may be used to combine them into 'complex' concept entities. This process can be recursive, thus providing for indefinitely complex concept entities.

GRAIL is *generative* and GRAIL models are *sparse*. A GRAIL model contains only the minimum information necessary to sanction the generation of all sensible concept entities. An indefinitely large number of concept entities can be inferred from the sanctions in the model and generated as needed without having to store them explicitly.

GRAIL classifies composite concepts automatically on the basis both of their definition and of indefeasible statements which are conceptually necessary to a concept. Hence there is no need for maintaining multiple classifications manually or even for specifying them in advance. Concepts such as “congenital heart disease” can be classified automatically under both congenital diseases and heart diseases without manual intervention.

GRAIL also provides a facility for attaching ‘extrinsic’ information to concept entities. Extrinsic information is information which does not affect an entity’s classification. For example, the statement:

```
Aspirin extrinsically mayBeBoughtIn 100mgTablets
```

is a representation of additional ‘real-world’ knowledge beyond what is necessarily true about Aspirin conceptually. The well structured taxonomies in GRAIL models are often useful and compact ways to organise other extrinsic knowledge.

## 2.3 The Terminology Server

The Terminology Server provides an encapsulation of, and a networked applications programming interface to, the CORE Model, the facilities provided by the GRAIL formalism, and linguistic and coding functionality. It provides means of referring to concept entities, asking questions of them, and transforming them into other representations, such as natural language. Individual modules within the Terminology Server handle different aspects of the overall task. The Terminology Server provides a uniform interface to the services provided by each of these modules, as well as combining multiple services into those useful for external applications. There are five major tasks that the Terminology Server as a whole performs:

- managing external references to concept entities ('Reference management') and coercion between data types;
- implementing the GRAIL formalism, and managing the internal representation of concept entities (implemented by the Concept Module) ;
- managing the data and functionality required to map concept entities to natural language (and, potentially, the inverse), (handled by the Multilingual Module);
- managing the data and functionality required to map concept entities to and from existing coding and classification schemes (handled by the Code Conversion Module);
- providing the functionality and management to handle extrinsic information (the Extrinsic Information Module).

Section 3 provides a functional description of the Terminology Server; Section 4 provides an overview of its architecture.

## 2.4 Expected Applications

The test of the Terminology Server will be whether it supports applications successfully. To be successful it must be shown to support applications both individually and, more importantly, within an environment of heterogeneous interworking clinical information systems. Our goal is not an abstract ‘pure’ representation of the essence of medical thought ; rather, the goal is a practical tool for developers of clinical information systems. Experience suggests that, within limits, ‘cleaner’, more formal representations lead to systems which are more flexible and extensible. However the ultimate criteria is use in practical applications; compromises are therefore inevitable.

Applications should benefit from the Terminology Server in at least four ways:

- Operations involving terminology can be delegated outside of individual applications;
- Development should be easier because it is based on existing ontologies and, increasingly, re-uses other work which uses those ontologies.
- Communication with other applications using the shared ontology should be possible.
- Many of the tasks of updating the system as new developments appear should be easier, because much of this work will be done by those maintaining the CORE Model. (There is, of course, the converse obligation to perform regression testing when there are major changes to the CORE Model or other aspects of the Terminology Server.)

GALEN itself includes experimental applications using the Terminology Server for electronic medical records, decision support systems, classification management and bibliographic retrieval. Other evaluations are being undertaken through collaboration with other projects. The results of these experiments will be reported on separately as they mature. More broadly, we expect to see at least six families of applications make use of the GALEN Terminology Server and CORE Model:

- Medical records, clinical user interfaces and clinical information systems
- Natural language understanding systems.
- Clinical decision support systems
- Management of, and conversion amongst, coding and classification schemes
- Bibliographic retrieval indexing
- Retrieval of clinical information, intelligent querying, research, and epidemiological analysis.

## 3. A Functional Description of the Terminology Server

The Terminology Server provides services for applications. In this section we first describe the patterns of functions and types of data handled by the terminology server. We then describe the kinds of question that may be asked.

### 3.1 Modes of Use and Types of Data

#### 3.1.1. Modes of Use

The GALEN Terminology Server is potentially used in two different ways:

- To support operational systems at run time with dynamic interpretation and encapsulation of codes, natural language expressions, and references.
- To support the development and maintenance of systems by providing a repository of concepts and terms and a means of extending this repository coherently and co-operatively.

When used to support operational systems, the Terminology Server must help applications in their interactions with end-users, and will primarily be asked questions. When used in development it must support editing and knowledge acquisition programmes in their interaction with knowledge engineers and other specialised users, and will frequently be told new information. The facilities and ergonomics of the two situations are markedly different. Whether they can be achieved within a single framework remains to be seen.

When used with operational systems, the Terminology Server will be an important part of a ‘mediation service’ to assist in access to existing heterogeneous databases. However, the larger payoff should be the use of the Terminology Server as a repository for concepts to enable the development of groups of coherent systems which can work together and build on each other cumulatively. More importantly, it should provide a means of maintaining and updating such groups of systems coherently as new information and new concepts need to be incorporated.

#### 3.1.2. Terminology Server Requests

A typical pattern of interaction by an application, whether as part of development or at run time, is to *connect* to the Terminology Server, perform a series of *requests*, and then to *disconnect*. This series of events is known as a *connection session.*

A request made of the Terminology Server by an application is specified in three parts: an operation, its input, and its required output(s). An important feature of the Terminology Server is that of mapping between different external representations (languages and coding schemes); as this is a common operation, we provide an invisible, automatic coercion mechanism. This mechanism performs the mapping to GRAIL concept entities from codes on input, and allows the caller to specify a series of required output formats which are then produced from the underlying concept entity which is typically the result of a Terminology Server call. This mechanism has the advantages of ease-of-use for the application developers, and of minimising the number of requests; because the Terminology Server is a networked resource, there is a fixed overhead per call.

The input or output types for the Terminology Server may be any of the following forms:

- ‘References’ — *e.g.* pointers — to (elementary or complex) concept entities. References can be combined into a specification of any complex concept entity.
- ‘Linguistic expressions’ which can be generated from (and potentially translated into) GRAIL expressions, but which are not in general unique. A major function of the Multilingual Module is to provide a buffer between the intrinsic ambiguity of natural language and the unambiguous formal representation in GRAIL of the CORE Model.
- ‘External expressions’ such as from coding and classification systems, database schemata, etc. which can be mapped into or out of the CORE Model. Mapping expressions and coping with the problems of mismatches, partial matches, and differences in granularity is the task of the Code Conversion Module.

In general the Terminology Server will accepts input objects in any of these forms and likewise will produce answers in any of these forms.

Where necessary, the Terminology Server will perform internal 'coercion' on input arguments and output results, by making calls of the individual modules invisibly to the user. For example, an application may have hold of an ICD code for Ulcer, and may wish to use the knowledge in the CORE model to produce a list of the possible relationships along which this may be refined. Informally: "what can I say about ulcers to describe them further?" The results are required to be put up on a screen, so we need output in a natural language (say French for this example), but we also require efficient handles for the results (see Section 3.3), so we can use them in subsequent requests to the Terminology Server. The request, in this example, will have the following form:

| | | |
|---|---|---|
| Input: | ICD531.9 | *an ICD Code* |
| Operation: | refiningRelationships | *a terminological operation implemented by the Terminology Module within the Terminology Server* |
| OutputSpecification | <asNaturalLanguage(french), asVolatileReference> | *an array of output specifications; this says that the output is required both as French natural language, and as a volatile reference for use in future requests* |

### 3.1.3. Reference to Concept Entities: Managing Persistence

One of the permitted input and output forms that the Terminology Server supports are 'references' to concept entities. References can be of three forms:

| | |
|---|---|
| Volatile | valid only during single application's connection session with a particular Terminology Server. These references (or 'handles') are the cheapest form of reference, but have the most limited lifetime. They are of fixed length. |
| Local | local to a particular Terminology Server at a particular site and its extensions. They are also of fixed length. Local identifiers are typically used for communication between applications that may connect to the same Terminology Server, or for local, long-term, data storage within applications. |
| Global | valid across all Terminology Servers containing a specific version of the CORE Model. Global identifiers have the widest applicability, though are the most expensive, and they are of variable length. They are used to communicate between different, geographically distinct, Terminology Servers. |

Applications may construct complex concept entities - GRAIL expressions - using any combination of global, local, and volatile references. A single reference may be thought of as an elementary GRAIL expression.

The Terminology Server can generate a single volatile or local reference from any GRAIL expression which is sanctioned by the CORE Model. However, there may not be an elementary global reference corresponding to a particular expression. Therefore, requests for global references may be of variable length and are thus expressions rather than pointers.

## 3.2 Questions which the Terminology Server can answer

### 3.2.1. What does this reference or expression mean?

The Terminology Server can be presented with an expression (for example made up of concept entity references) which may or may not correspond to one or more legal concept entities sanctioned by CORE Model. If sanctioned, a concept entity may or may not already have been generated by the Concept Module. The first task of the Terminology Server is to examine the expression, convert it to a sanctioned GRAIL expression if possible, and then see if either a corresponding concept entity exists, or if not to generate and classify the new concept entity required. Note however that the external application is unaware of which of these actions has been taken. The Terminology Server can then answer questions such as:

- Is this a legal expression, and what is its simplest form (e.g. with any redundancies removed)?
- If it is legal, how is it classified— what more general concepts subsume it? What more specialised concept entities does it subsume?
- What is known about this concept entity conceptually from the CORE Model? What other extrinsic information has been said about this concept entity?

### 3.2.2. What can be said about this concept entity?

A major function of the Terminology Server is to tell applications what further can be sensibly said about a concept entity — to support a user interface to help clinicians enter the information; to assist a bibliography system refine a query; or to assist a natural language system to disambiguate candidate phrases. Correspondingly, much of the information in the CORE Model is not about what is true but about what can sensibly be said. Once concept entities are generated and classified, the Terminology Server can therefore answer questions such as:

- What statements can sensibly be made about this concept entity? What are its sensible modifiers and relations?
- How can this concept be specialised according to given criteria? — *e.g.* anatomically, functionally, according to clinical indications or effects.
- What are the 'sensible' ways in which this set of concepts can be combined into a single larger concept?

Answers to any of these questions may involve generating further new entities. For example, the CORE Model does not store information about every phalanx of every finger explicitly, but it can respond to questions such as the parts of the "left fourth finger" by generating entities representing concepts such as the "first phalanx of the left fourth finger", "proximal interphalangeal joint of the left fourth finger", "second phalanx of the left fourth finger", etc.

### 3.2.3. What are the nearest representations to this in some other representation?

Another major use of the Terminology Server is to convert between external representations including both coding and classification systems and natural languages. (Although initially its ability to convert from natural language will be limited). Exact conversion or translation is not always possible because the corresponding concepts may not have representations in the target system. The Code Conversion and Multilingual modules are responsible for providing applications with a variety of strategies for coping with inexact matches. However, in many situations the best that can be done is to provide the application with the information on the potential matches and details about the imperfections in the matching process. It is then up to the application program to decide how to deal with this information according to its own particular requirements.

Conversion using the Terminology Server is always a two-stage process — first map the expression into the CORE Model and then map it back into the target external or linguistic representation. At the same time, the Multilingual and Code Conversion Modules maintain extra information to enable them to answer specific questions relating to the external representations. Combining and encapsulating these techniques the Terminology Server can respond to questions such as:

- What are the external expressions for this concept entity in a particular external system? What is the preferred term for this concept entity in that system?
- What are the natural language expressions for this concept in a particular language? What is the preferred form for a particular 'clinical linguistic group'.

- Are these two concept entities derived from two different external representations the same? If not, how do they differ? What information would have to be added or removed from each to make them the same?
- Find all of the expressions in a given external representation which correspond to children of this concept entity, i.e. all of the codes which this concept entity subsumes. This is a particularly important question for information retrieval. It allows the Terminology Server to compensate for the deficiencies in the organisation of external coding systems. For example, forms of heart disease are found in at least five different chapters of ICD-9.

#### 3.2.4. Encapsulation and the transformation of terms

Different applications may require information to be encapsulated in different forms. In general, applications want to store the information which they manipulate locally and to encapsulate information which they do not expect to need. Therefore a surgical system might want to record the approach, instruments, and method of anaesthesia separately whereas a general practice system might want to encapsulate these details into a single code for a surgical procedure.

Furthermore, most systems use relational technology which is based on fixed length identifiers for most fields. The variable length recursive structures from the CORE Model itself fit badly into relational schemes. One of the functions of the Terminology Server is to encapsulate complex expressions into fixed length references and to provide alternative sets of such references representing different degrees of encapsulation — *e.g.* a single reference for a surgical procedure for a general practice system or separate references for the main procedure, approach, anaesthesia, and instruments used for a surgical database.

One special case is that systems differ as to which ‘nominalisation’ they wish to use to encapsulate particular information — whether to record the “fracture of the femur” or the “femur which is fractured”. Applications also vary as to which of various dualities they wish to regard as primary — *e.g.* whether to record “the ulcer” or “the process of ulceration”. The terminology server provides a number of special purpose functions to deal with the technical issues and can respond to requests such as:

- Transform this concept entity into an alternative nominalisation or an alternative form within one of the recognised dualities.
- Provide a volatile, local or global reference for this concept entity.
- Encapsulate these concept entities according to a given format for an application as a set of references or a set of external expressions.

#### 3.2.5. What other extrinsic information has been attached to this concept besides the indefeasible terminological knowledge?

Strictly speaking, the CORE Model contains only concrete conceptual knowledge which is indefeasible and true ‘by definition’. However, a major function of the CORE model is to provide a framework with which to organise other, more general information. Holding such information and retrieving the most specific information in a certain category available — *e.g.* concerning drug interactions, clinical procedures or diagnostic methods— is so useful that additional operations are provided to support these functions directly. There are three primary operations:

- Find the most specific information in a given category about a concept entity.
- Find all of the information in a certain category about a concept entity and all of its parents.
- Find all the children of a particular concept entity such that a particular piece of extrinsic information holds.

### 3.3 Things the Terminology Server can be Told

One of GALEN’s major goals it to support local extensions and flexible development within an overall coherent framework provided by the CORE Model. Local sites and applications must therefore be able to add information to the Terminology Server in a number of different ways. These functions are still under development as we gain experience with using and developing the Terminology Server, and the different types of knowledge it contains.

#### 3.3.1. To extend the existing model

Local site may need to extend the model itself to fit their needs in a number of different ways. The goal is that many changes can be made locally without prior reference to the central management of the CORE Model. Some of those changes may eventually be incorporated into the global model and distributed more widely; others may remain strictly local.

- By giving new local names to existing or potential concept entities. Adding local names does not increase the range of things which can be expressed by the model, but it can make the model much easier to use by simplifying what would otherwise be complex expressions. New local names can always be given without reference to the central co-ordinators.
- By adding new primitive concept entities. The range of primitive concept entities may not include things which are important locally. For example, a surgical system might not include names for all of the surgical instruments used at a particular site. New detailed concept entities in existing categories can normally be provided locally but need to be notified to the co-ordinators so that any potential conflicts with other users working in the same area can be monitored and reconciled where necessary. New major categories require more careful control and co-ordination.
- By adding new attributes and associated sanctions so that new things can be said. As with adding new primitive concept entities, the range of attributes may not support sufficient detail for local use. Detailed extension within the overall framework can normally take place locally. More global changes require central co-ordination.
- By adding new sanctioning statements so that existing attributes and concept entities can be used in new ways. It is often the case that the sanctions in the CORE model are too specific for local use and may have to be extended. However, the feeling that the sanctions need to be relaxed often indicates misunderstandings concerning the intended use of the model. Therefore, except where sanction are being extended trivially to cover new primitives, changes to sanctions need to be made with care and notified to the co-ordinators.
- By adding new statements of conceptually necessary facts. Making a conceptually necessary statement about a concept entity may cause that entity to be classified in an additional way, or even cause two entities which were previously distinct to coalesce into a single entity. As in the previous examples, local changes which simply increase the available detail can be made locally but need to be notified to the central Co-ordinator. More drastic changes must be carefully monitored. Any changes which cause two entities to coalesce need to be verified centrally.

#### 3.3.2. To add to or modify the mappings to external representations in the Code Conversion module

Many of the changes to the Terminology Server involve adding to an existing external representation or adding a complete new external representation. The status of these changes depends on the status of the external representation. Addition of a complete local external representation — a local coding system or database schema is obviously a local matter. Changes to the mapping to ICD or SNOMED need to be made with great care and probably indicate errors which should be notified centrally.

#### 3.3.3. To add to or modify the linguistic information in the Multilingual Module

The structure of concept entities maintained by the Concept Module within the Terminology Server is language-independent. The Multilingual Module maintains the data and the functionality required to map any concept entity into (potentially) any natural language. Local users may want to add to the translations in the Multilingual Module or change details about the preferred use of language locally. These changes can be made completely locally without affecting the other users of the Terminology Server. However, major linguistic developments may be of much wider interest and should probably be notified to the central co-ordinators.

#### 3.3.4. To add or modify the additional extrinsic information attached to concept entities.

As discussed in Section 2.2, an important facility the Terminology Server offers is to annotate the conceptual model with extrinsic information. The Terminology Server provides facilities for adding such annotations, and a series of operations to find such annotations from any concept entity (e.g. one sanctioned but never before seen) using the conceptual classification maintained by the Concept Module within the Terminology Server.

## 3.4 Global Operations on the Model

In addition to operations on individual concept entities and expressions, there are operations which can be performed on the model as a whole. Normally the operations are performed either after making changes to the model or centrally a part of the overall maintenance function for the CORE

Model. These functions are still under development as our experience with the Terminology Server, and the amounts of knowledge held within it grow.

#### 3.4.1. Coherence checking

Total checking of the CORE Model is probably computationally intractable. However, a wide range of checks can be performed both on individual concept entities and on the model as a whole. Global checking may only be practical at central computing sites with large computing resources.

#### 3.4.2. Providing information on the editorial status of items

The Terminology Server maintains information on the editorial status of the model and on the background and expected usage of the concept entities in it. These are currently maintained as special meta annotations and text comments; the range of facilities is growing rapidly as experience with the model grows.

#### 3.4.3. Managing updates

There is a great deal of 'housekeeping' to be done to manage the integration, distribution, and acceptance of updates and the notifications between various users of the Terminology Server and the CORE Model. The Terminology Server requires functions to manage these changes both centrally and in each co-operating centre. The best means, organisationally and technically, are under vigorous investigation.

#### 3.4.4. Local coding schemes

One of the important features of the Terminology Server is to be able to 'compile out' sections of the conceptual model in a form recognisable to existing applications. This is equivalent to building a 'local coding scheme' dynamically, and makes the functionality of the Terminology Server more widely available. Whilst it is not possible to take advantage of all the facilities of the Terminology Server by using such local coding schemes, it does make the knowledge available to a wider range of applications. Furthermore any data collected using such a 'local scheme' can always be referred back to the Terminology Server with which it remains consistent, if additional analyses are required.

## 4. The Terminology Server Architecture

GALEN's approach is to divide the tasks and information usually summarised under the heading of 'terminology' into several semi-independent pieces, to develop well defined techniques for dealing with each, and then to present all of the services addressing these tasks through a uniform interface — the Terminology Server's Applications Programming Interface. The overall architecture is shown in two different levels of detail in figures 1 and 2.

### 4.1 High Level Architecture

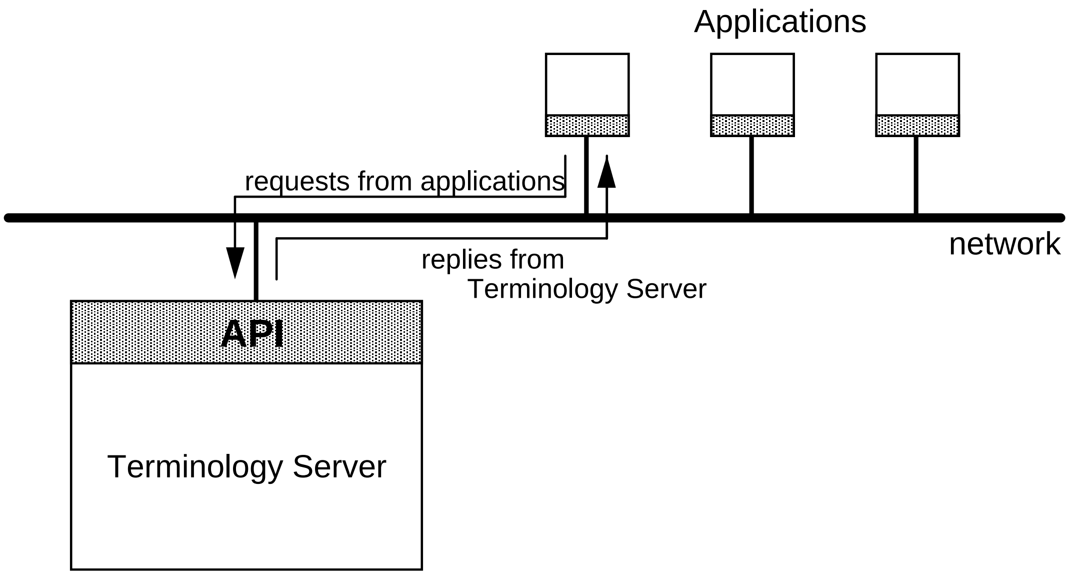

Figure 1: The high level architecture of the Terminology Server.
The Terminology Server provides a networked resource for applications.

The Terminology Server provides a uniform applications programming interface (API) to its modules. It provides a common query language so that applications can transparently make complex requests involving more than one module , and provides a uniform means of specifying the forms of input supplied and output requested.

The Terminology Server’s applications programming interface is designed in such a way that new modules can easily use it as a means to export their services to the network. This is appropriate where a module’s function is so closely tied to the other services of the Terminology Server that applications developers find it convenient to have them bundled together.

### 4.2 Internal Architecture of the Terminology Server

Figure 2 presents an overview of the internal architecture of the Terminology Server. Externally, the Terminology Server presents a modularised view of terminology to external applications. This pattern of modularisation is echoed in its internal architecture. The overall task of managing terminology has been modularised into different aspects - conceptual, linguistic, coding, and extrinsic - which are implemented by separate modules within the Terminology Server. The Terminology Server combines these modules, adds reference and coercion mechanisms, and exports individual module services, via the API, to applications. The Terminology Server's reference management makes it easy for external applications to reference and store concept entities, for example as part of a patient record system. The Terminology Server's coercion mechanism provides efficient ways of combining multiple module services and relieves applications of needing to know how specific requests are handled.

A flexible interface has been developed so that individual modules may 'export' their services, via the API, to external applications, so additional functionality can be made available very quickly. Maintaining modularity within the Terminology Server provides the additional advantage in software engineering terms of allowing different development groups to proceed with developments and enhancements in which they are expert, with the confidence that integration into the Terminology Server is straightforward.

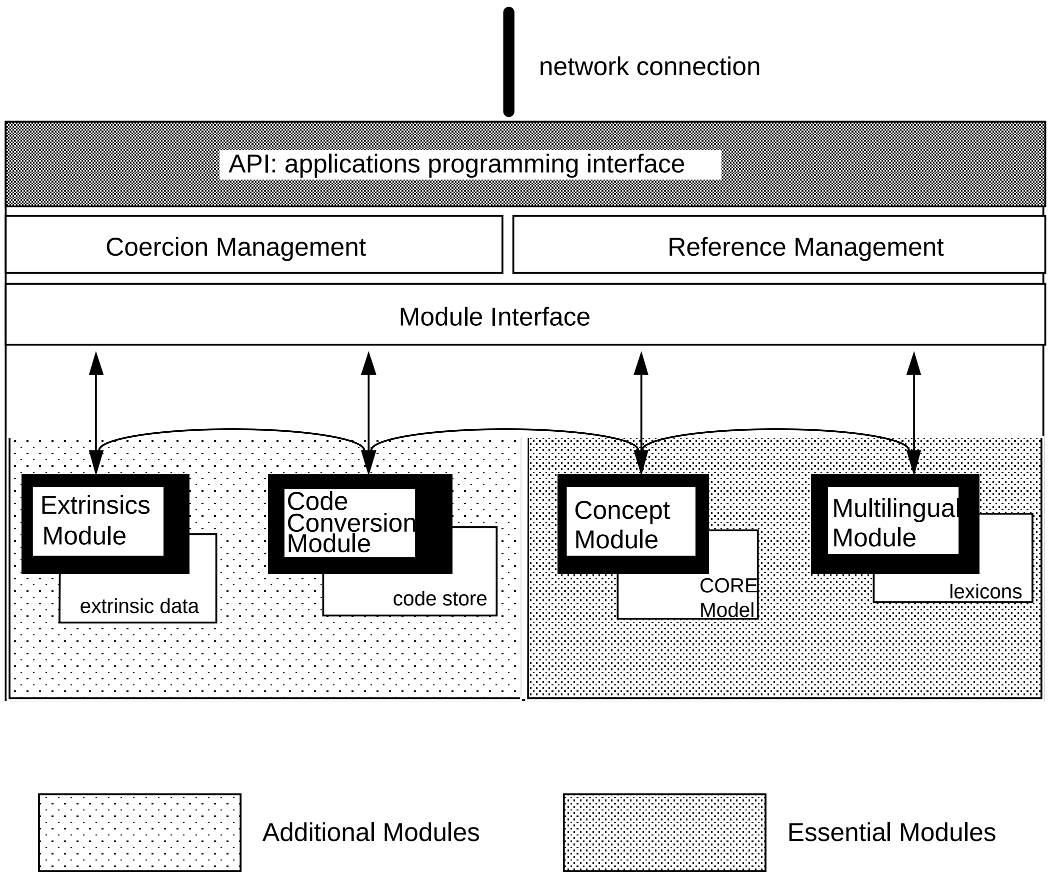

Figure 2: An overview of the internal architecture of the Terminology Server. The Coercion Management Layer performs coercion on a request's inputs and outputs, and provides facilities for specifying how such coercion should take place. The Reference Management layer mediates between the external references by which applications may refer to concept entities, and the internal representation of concept entities which is managed by the Concept Module.

The central task of concept modelling is addressed by the ‘Concept Module’ which interprets the ‘Concept Reference’ (CORE) Model[2]. The CORE Model serves as an interlingua amongst medical nomenclatures, vocabularies, and the terminological aspects of database schemata. When performing conversions all terms from external coding systems, nomenclatures, database schemata or other external representations are first converted into the CORE model and classified. Any other processing requested is performed on the CORE Model representation. If a response is required in a form other than an expression from CORE Model itself, then the result is converted into the required output forms (*i.e.* ‘projected’ onto the output schemata). A basic assumption is that the CORE model will be at least as detailed as the union of the supported external representations. The goal is that translation amongst n external representations requires maintaining only *n* mappings to and from the CORE model rather than *n(n-1)/2* 2-way mappings between all possible pairs of representations.

The Multilingual Module provides lexicons and grammatical information for expressing, and eventually understanding, phrases in natural languages. The goal is that external representations need not supply their own translations to various natural languages but can depend on the Multilingual module to translate the CORE Model expansion of their representation. (It is also, however, necessary to support official translation of particular coding schemes via individual mappings from external representations to their official translations.) The concept entities within the Concept Module are language-independent; the Multilingual Module maintains and presents linguistic interpretations of these concept entities. A minimally functional Terminology Server must contain at least a Concept Module, and a Multilingual Module to provide a linguistic interpretation. Further functionality is added, as described, by adding a Code Conversion Module and Extrinsic Information Module.

The Code Conversion Module maintains the external representations themselves, along with special information related to their structure and browsing, *e.g.* information on the cross referencing in SNOMED or the dagger-asterisk mechanism and exclusions in ICD-9/10. The Code Conversion Module also provides the functionality concerned with resolving ambiguities and conflicts when there is not an immediate one-to-one correspondence between the GALEN CORE Model and the target external representation or when there — for example when the expansion of a term from one external coding system has no direct representation in a different external coding system.

The Extrinsic Information Module provides a repository in which applications or sites can store detailed information about the clinical criteria for using concepts in the Terminology Server. These definitions are ‘hung onto’ the classification structure of the CORE Model but are not part of it. Different clinical linguistic groups have different criteria for diagnosing diseases such as rheumatoid arthritis or schizophrenia. The concept model is not intended to be a diagnostic decision support system, much less a normative model of care for Europe. However, one of the expected uses of the Terminology Server is the support of applications that wish to test or enforce such conventions locally.

## 5. The CORE Model: Requirements for Re-use — avoiding application specific decisions

Much of the success of the Terminology Server will depend on the adequacy of the CORE Model and the formalism in which it is represented, the GRAIL Kernel. The GRAIL Kernel is described in detail elsewhere [6, 18, 19] , but two considerations in its design should be re-iterated.

- The CORE Model aims at application-independence and re-use. This means that the information in the CORE Model held in the Terminology Server will usually be greater and more detailed than needed by any single client application. Client applications must be able to

[2] Sometimes known in early documents as the ‘Terminology Engine’ and ‘COding REference’ model, respectively.

address the Terminology Server in different ways appropriate to their own situation. Wherever possible, users should be protected from detail that does not affect them.

- The Terminology Server does not provide a complete reasoning system. Applications are expected to provide additional inference or other processing capabilities. What is provided is a service for classifying and harmonising the concepts and terminology used.

Making the CORE Model application independent means avoiding application specific decisions. This near tautology leads to an analysis of where application-specific decisions occur in knowledge representation systems. That analysis leads to key features of The GRAIL Kernel:

- Constructs in the language which promote clean, homogeneous taxonomies which are recombined through composition and generation, including constructs to co-ordinate part-whole relations with subsumption, plus a modelling style which exploits this constructs.
- A view of the category-individual (class-instance) distinction which avoids arbitrary choices. Choices concerning what level of detail should constitute an 'instance' such as those described graphically by Brachman in [20] are avoided by restricting individuals only to concrete instances in the real world and their properties. In this respect it is closely analogous to Sowa's treatment of types (corresponding to GRAIL categories) as lambda abstractions over individuals and hence fundamentally different from them. GRAIL models the 'level of specification' required by individual applications as explicit 'external' knowledge about those applications.
- Support for generation of implied concept entities, which means only that the basic model (roughly equivalent to the 'basis' in Conceptual Graphs [21]) be represented explicitly. Other concept entities are generated as needed. This allows the Terminology Server to behave as if it contained an indefinitely large number of concepts while physically representing only a compact model.
- Features which facilitate alternative encapsulations and which bridge the different levels of detail required by different applications.
- Recognition that the model can never be complete, and that it therefore functions in an open rather than a closed world with corresponding constructs and restrictions on the formalism.
- Restrictions on the range of constructs supported to those deemed 'terminological'.

## 6. Discussion

## 6.1 A Terminology Service rather than a Terminology

The idea of a "Terminology Server" represents a new way to view the role of terminology in information systems. Previously, terminologies have been static and used only during development or 'compile time'. A terminology was something which could be written down or at least stored in a straight forward database. Any manipulation of the terminology was left to individual applications. SNOMED-III, the READ Codes and ICD-9 all provide one degree or another of prescriptive advice about how the coding system is to be used, but they are defined in terms of the structure rather than the functions performed. In contrast a terminology server delivers *terminological services*, that provide high level functionality to applications.

Used in this way we believe that the 'terminology' can become a potent integrating force helping to mediate between different systems and different applications, providing a consistent linguistic service for many different applications. The server also provides a way of encapsulating one aspect of the variability and complexity of clinical data in forms more palatable to conventional information services.

The idea of a terminology service rather than a terminology has several further ramifications.

### 6.1.1. Separation of responsibility and limitations

The idea of a Terminology Server embodies the separation of responsibility between the terminology or concept-modelling functions and other functions in applications. Up to a point, this separation reflects current practice — coding and classification systems are developed separately from the medical records and hospital information systems in which they are used. However, this is a purely static separation and the line between the application and the terminology may be blurred with many values and functions being handled procedurally within the application.

Providing a separate Terminology Server requires that the nature of that service be well defined. Potential client applications must know what they can and cannot ask of the Server. This requires that the limitations of what is considered 'terminological' must be carefully defined. There is always a tendency to increase the bounds of any technology, pushing it to the limits of what it can do rather than establishing what it can do best. GALEN has attempted to be rigorous in limiting the Terminology Server to a set of functions which involve indefeasible reasoning about the 'intrinsic' conceptual characteristics of concepts. While it has proved convenient to provide limited facilities to record other 'extrinsic' characteristics which applications wish to organise using the CORE Model hierarchies, the Terminology Server performs no inference with such extrinsic information. There is a strong tradition for this division going back to Brachman and Levesque's KRYPTON [22].

#### 6.1.2. Mediation, Re-use and Knowledge Sharing

The GALEN Terminology Server is an important component in a strategy for mediation between heterogeneous applications. It is not a complete mediation service — some differences may require extensive computation for conversion, *e.g.* between different numerical scales and co-ordinate systems. However, a large class of problems in mediating between medical information sources concerns the semantic content of those sources.

Because of the ability to transform between different forms and re-encapsulate structures in different ways, GALEN's approach to knowledge sharing and re-use requires less rigid adherence to a single standard than fixed coding systems such as ICD 9/10 [23], the Read Codes [3] , or even SNOMED III [2]. Alternative representations which are appropriate to individual applications can be used. Conversion can occur either dynamically at run time or the system can be used during development to assist in static translations. On the other hand, to those willing to make that commitment during development, the CORE model through the Terminology Server is intended to provide a source of concepts which will make the independent development of coherent systems much easier.

Of other knowledge sharing efforts, the DARPA Knowledge Sharing Effort [11, 13, 24] focuses primarily on translation between ontologies and the provision of standard ontologies during development. It normally requires a minimal commitment in advance to a single shared ontology and provides no support at run-time. The GALEN Terminology Server is perhaps more analogous to some applications envisaged for Cyc [25-27] in which Cyc would provide a general substructure and services to many applications. However, GALEN's CORE Model is strictly limited to medical conceptual knowledge, whereas Cyc's knowledge base aspires to general common sense knowledge of the world.

#### 6.1.3. Distributed development

The ability to separate running applications across sites offers the potential to distribute development effort across multiple centres. Distributed development remains a long term goal to be pursued. However, achieving successful distributed development requires implementing sophisticated strategies for notification, locking, and version maintenance which are still only in the planning stages.

#### 6.1.4. Architecture and Extensions

One of the successes of the project has been the development of an architecture into which new modules and services can be easily incorporated. GALEN is experimental, and it is premature to determine which services will be best packaged together. It is important to modularity that the developers resist the temptation to extend the server without limit. On the other hand, when services are encountered which are tightly coupled to the terminological functions they can be incorporated as needed.

### 6.2 Evaluation

Ultimately, the Terminology Server will be judged by how well it supports applications,. There are at least two broad areas for evaluation:

- Does the terminology server and model support individual applications effectively?
- Can the same terminology server and model support several communicating applications?

These questions can be further separated into two groups:

- Those which concern the idea of a Terminology Server and its functions *per se;*
- Those which concern the CORE Model and the facilities of the GRAIL Kernel modelling language.

This paper has concentrated on the functions of the Terminology Server *per se* the key issue for which must be whether the separation and architecture are convenient for the development of applications, given an effective CORE Model.

The GALEN project itself contains applications to test the clinical effectiveness of the combined Terminology Server and CORE Model, including clinical user interfaces, medical records and knowledge based systems. Further collaborative developments are planned. Initial results are promising, but definitive results will have to await further experience. (There are also a range of procedures in place to evaluate the CORE Model itself.)

### 6.3 Current Status

GALEN is a long term project to demonstrate the feasibility of the approach both to the architecture of the terminology engine and to the techniques of concept modelling. Current progress is promising but does not yet constitute definitive evidence of that feasibility. As of June 1994, initial versions of the Terminology Server have been implemented and applications for clinical user interfaces, medical records and knowledge based systems are now being developed. Portions of the CORE Model have been compiled for gastro-intestinal diseases, arthroscopy proceedings and findings, and urinary tract and respiratory tract infections. A general framework for a model of anatomy has been developed and the broad shallow model of anatomy is nearing completion. The concept of a client-server architecture has been tested, and it has been shown that applications and the server can interact successfully running on different machines linked across either local area networks or across the Internet.

## 7. References

1. Lindberg D, Humphreys B, McCray A. The Unified Medical Language System. In: van Bemmel J, ed. 1993 Yearbook of Medical Informatics. Amsterdam: International Medical Informatics Association, 1993: 41-53.
2. Côté R, Rothwell D. SNOMED-3.Chicago: College of American Pathologists, 1993
3. Read J. The Read Clinical Classification. In: NHS Centre for Coding and Classification, Loughborough, UK, 1993:
4. Evans DA, Cimino J, Hersh WR, Huff SM, Bell DS, The Canon Group. Position Statement: Towards a Medical Concept Representation Language. *Journal of the American Medical Informatics Association* 1994;1 (in press).
5. Cimino JC, Hripscak G, Johnson S. Knowledge-based approaches to the maintenance of a large controlled medical terminology. *Journal of the American Medical Informatics Association* 1994;1(1):35-50.
6. Rector A, Nowlan W, Glowinski A. Goals for Concept Representation in the GALEN project. 17th Annual Symposium on Computer Applications in Medical Care (SCAMC-93). McGraw Hill, 1993: .
7. Rector A, Nowlan W, Kay S. Conceptual Knowledge: The Core of Medical Information Systems. In: Lun K, Degoulet P, Pierre T, Rienhoff O, (ed). *Seventh World Congress on Medical Informatics,* MEDINFO-92. Geneva: North-Holland Publishers, 1991: 1420-1426.
8. Rector A. Marking up is not enough. *Methods of Information in Medicine* 1993;32(4):272-273.
9. Lenat DB, Guha RV, Pittman K, Pratt D, Shepherd M. Cyc: toward programms with common sense. *Communications of the ACM* 1990;33(8):30-49.
10. Lenat RGaDB. Re: CycLing paper reviews. *Artificial Intelligence* 1993;61(1):149-74.
11. McGuire JG, Kuokka D, Weber JC, Tenenbaum JM, Gruber TR, Olsen GR. SHADE: Technology for knowledge based collaborative engineering. *Jounral of Concurrent Engineering: Applications and Research (CERA)* 1993;1(2).
12. Neches R, Fikes R, Finin T, et al. Enabling Technology for Knowledge Sharing. *AI Magazine* 1991;(Fall 1991):37-54.
13. Patil RS, Fikes RE, Patel-Schneider PF, et al. The DARPA Knowledge Sharing Effort: Progress Report. Principles of Knowledge Representation and Reasoning, Third International Confrence. Cambridge MA: Morgan Kaufman, 1992: .
14. Musen M. Dimensions of knowledge sharing and reuse. *Computers and Biomedical Research* 1992;25:435-467.
15. Walther E, Eriksson H, Musen MA. Plug-and-Play: Construction of task-specific expert-system shells using sharable context ontologies. AAAI Workshop on Knowledge Repreentation Aspects of Knowledge Acquision. San Jose CA: , 1992: 191-198.

16. Schreiber A, van Heijst G, Lanzola G, Stefanelli M. Knowledge organisation in medical KBS construction. In: Andreassen S, Engelbrecht R, Wyatt J, (ed). Fourth Conference on Artificial Intelligence in Medicine Europe. Munich: IOS Press, 1993: 394-405.
17. Masarie Jr F, Miller R, Bouhaddou O, Giuse N, Warner H. An interlingua for electronic interchange of medical information: using frames to map between clinical vocabularies. *Computers in Biomedical Research* 1991;24(4):379-400.
18. Rector A, Nowlan W. The GALEN Representation and Integration Language (GRAIL) Kernel, Version 1. In: The GALEN Consortium for the EC AIM Programme. (Available from Medical Informatics Group, University of Manchester), 1993:
19. Rector AL, Nowlan W. A Reusable Application Independent Model of Medical Terminology: GALEN's GRAIL. KR-94. Berlin: Morgan Kaufmann, 1994: (in press).
20. Brachman RJ, McGuinness DL, Patel-Schneider PF, Resnick LA, Borgida A. Living with Classic: When and how to use a KL-ONE-like language. In: Sowa J, ed. Principles of Semantic Networks: Explorations in the representation of knowledge. San Mateo, CA: Morgan Kaufmann, 1991: 401-456.
21. Sowa J. Conceptual Structures: Knowledge Representation in Mind and Machine.New York: John Wiley & Sons, 1985
22. Brachman R, Fikes R, Levesque H. An essential hybrid reasoning system; knowledge and symbol level accounts of KRYPTON. International Joint Conference on Artificial Intelligence (IJCAI-85). Morgan Kaufman, 1985: 532-539.
23. World Health Organisation. International Clasification of Diseases.Geneva: World Health Organisation, 1989
24. Fikes R, Cutkosky M, Gruber T, Baalen jV. Knowledge Sharing Technology -- Project Overview. In: Stanford University, Knowledge Sharing Laboratory, 1991:
25. Lenat DB, Guha RV. Building Large Knowledge-Based Systems: Representation and inferenc in the Cyc Project.Reading, MA: Addison-Wesley, 1989:372.
26. Lenat D, Guha R. Ideas for applying Cyc. In: MCC, 1991:
27. Guha R, Lenat D. Cyc: a midterm report. *AI magazine* 1990;11(3):32-59.