Historical Records Processing in the HiTeX System

J.N. Oliveira [*]
A.S. Araújo [†]
A.M. Silva [‡]

August 1991

Abstract

This paper presents an outline of HITEX, an open software system for recording, organizing and maintaining historical data. Particular attention is paid to a formalization of the model chosen to support data organization in HITEX, which is a hybrid between the AI *semantic network* model and the *object oriented* computational model.

The notion of a *re-usable historical component* is put forward as a basis for standardization and exchange of machine readable historical data.

Practical aspects concerning historical knowledge acquisition and source transcription in the current HITEX prototype are discussed.

In *Yesterday*, 149--168 (Editors: Hans J. Marker and Kirsten Pagh), Proceedings of AHC'91 -- 6^{th} International Conference of the Association of History and Computing, Odense, Denmark, 28--30 Aug. 1991, Odense University Press, 1994.

[*] Dept. de Informática, UM/INESC, Rua D.Pedro V, 88-3, 4700 Braga, Portugal.
[†] Arquivo Distrital de Braga, Largo do Paço, 4719 Braga Codex, Portugal.
[‡] Universidade do Minho, Largo do Paço, 4719 Braga Codex, Portugal.

1 Introduction

The main purpose of this paper is to describe the motivation behind the conception of HITEX, a software system for recording, organizing and maintaining historical data. This system is currently being developed for the District Archives of Braga (ADB), Portugal, in collaboration with the Computer Science Department of Minho University at Braga and IBM Portugal, under the financial support of the Calouste Gulbenkian Foundation (Lisbon).

The paper's structure is as follows: we start by reviewing currently open issues concerning the mechanical handling of historical records. This is followed by an outline of the project's evolution and of the mathematical model chosen to support data organization in HITEX. The user's interface with the system and details concerning its pratical use are presented in the sequel. A review of related work followed by a few conclusions and an outline of future work end the paper.

1.1 Computing and the Historical Disciplines

Many institutional problems concerning the organization of modern society are still hard to treat by computer. Although many others have known a successful mechanization in the recent past (bank accounting is a paradigmatic example), there is a general feeling that a significant gap lies between the available technology and the enormous complexity of some non-trivial problems one would like to tackle by computer. Typical non-trivial application domains are *legal systems*, *linguistics* and *historical research* (to quote only a few of them).

A most striking shortcoming of computers is made evident wherever one endeavours to replace human activity by automatic processing — computers are unable to cope with vagueness or ambiguity, two essential aspects of human nature. This may explain why human-computer interaction is so relevant an area of R&D in present day computer technology.

As a matter of fact, computers understand the real world only through "stylized" *models* from which every subjective subtlety has been ruled out. Therefore, any automated solution for a given "human" problem presupposes the conception of an *unambiguous* model of the problem domain. Such unambiguous models are *formal* in the sense that they possess some objective *mathematical* structure, which can be emulated by an automaton — by a piece of computer software, for instance.

Such an emulation is affected by yet another "epistemological gap" between humans and computers — however fast and efficient, the latter are grotesque automata whose "language" is hard to read and reason about by the former. As a consequence, software design is normally split in two phases according to the dicotomy between so-called *specifications* (formal models of real world problems written in some mathematical notation) and *implementations* (machine-readable descriptions of such models written in some programming language). Current software technology is much concerned with strategies for reliably deriving the latter from the former.

In summary, the reliable application of computing technology to areas such as historical research is faced with (at least) two levels of technical complexity:

- “disambiguation” (= build specifications)
- implementation (= build programs).

as described above. We write “at least” because further elaboration is required, arising from the evolutive nature of the historical data themselves. There is hardly a *stable*, consistent specification of one’s historical “knowledge”, which is typically under permanent revision. To worsen things, many questions which arise in historical research involve processing of huge amounts of (presumably inconsistent) data.

Lack of information stability is determinant to the kind of software technology adequate to the historical disciplines, ruling out conventional data-processing tools based on rigid database schemata. On the one hand, it forces formal models developed in each application domain to step from “compile-time” to “run-time”, in order to become modifiable throughout the application’s lifetime. The need for interactive processing (*i.e.* user assistance) is therefore obvious. On the other hand, efficient connection to the original information source (*e.g.* manuscripts, record books *etc.*) is desirable, suggesting the need for multi-media environments. For huge amounts of documentation one may resort to high capacity memory devices such as optical disks. Furthermore, the dependence of traditional historical research on publishing source transcriptions suggests that text processing tools should be made available for computerized typesetting.

1.2 The Hypertext Paradigm

Hypertext systems [4, 12] meet most of the above requirements and have been successfully applied to small-scale historical case-studies, *cf. e.g.* reference [16]. However, it is difficult to assess commercial hypertext systems because of their “ad hoc” semantics hidden behind sophisticated user-interface layers.

A helpful formalization of the hypertext paradigm has been developed at T.U. Denmark by Lange [12] using the Vienna Development Method (VDM) [10]. His model for hypertext consists of a networked collection of attributed units of information, named *nodes*, together with *links*, the “glue that holds hypertext together”.

As Fountain *et al* [8] point out, hypertext techniques give rise to further usability problems that have yet to be resolved. Currently available hypertext packages are basically *closed systems* which do not communicate bi-directionally with other software packages. Documents are becoming available in computer readable form far faster than they can be converted to hypertext. Standard hypertext systems add structure to documents by means of links or tags hard-coded into their textual representation. Such links cannot be added or revised on read-only media (*e.g.* CD-ROM).

Although some hypertext systems partly solve these problems, it is clear that there is room for technical innovation in the area. HITEX attempts to provide an alternative system philosophy based upon an *open architecture* communicating with pre-existing software tools such as text processors, knowledge/data-bases, image scanning software, optical disk software packages *etc*. HITEX's open philosophy is suggested by the system name itself — HITEX— which comes from TEX[11], the well-known open document preparation system designed by Donald Knuth, chosen to be HITEX's "default" text processing tool. Such a motivation to build HITEX is shared by other research communities, notably by the LACE and MICROCOSM projects at Southampton University [19, 8], which will be related to HITEX in section 6.

1.3 The Need for Re-usability

A concern related to system openness is *re-usability*. A well-known, negative syndrome tends to affect the uncontrolled fragmentation of computing resources in independent personal workstations — incompatible models of the same problem may proliferate (everybody doing it in their own way!) rendering future integration very hard, if not impossible.

The AHC'91 expected topic *"Standardization and exchange of machine readable data in the historical disciplines"* expresses some concern about this syndrome reaching the historians' community, where many personal databases are being built independently, with little concern for compatibility.

Re-use is the "magic word" waved in the computer industry against this long identified incompatibility epidemic. Both the hardware and software industries have learnt to fight it by producing *re-usable components*, that is, standard system building-blocks with appropriately defined interfaces making for the exchange of equivalent components [1]. Larger systems become available by mere integration of pre-existing system components, instead of being built from scratch. Productivity is thus much improved.

The HITEX system attempts a parallel of this strategy by introducing the notion of a *re-usable historical component* (RHC). RHCs are archived according to a standard *taxonomy* of historical concept classes expressing a *subsumption* ordering on historical knowledge. As detailed later on (*cf*. section 3), RHCs are tolerant towards *incomplete* information, a final aspect of practical relevance to one's incremental acquisition of historical knowledge.

2 HITEX Antecedents and Project Evolution

The HITEX project was launched in January 1989, arising from two confluent directives of ADB management concerning the effective use of computing resources formerly installed in the Archives by IBM Portugal. On the one hand, a software infra-structure was needed for incrementally organizing and accessing the vast collections of documents entrusted to the Archives. On the other hand, ADB's editorial board wanted to upgrade its publishing technology whilst launching a new editorial series of commented source

[1] Due to its relative technical under-development, software has been slower than hardware to adopt such a production strategy, *cf*. *e*.*g*. reference [22].

transcriptions of particularly interesting originals. The intended software system was furthermore expected to absorb reports issued by pre-existing databases [2].

The first phase of the HITEX project was concerned with designing a simple to use mark up language (the *HiTeX format*) satisfying ADB's editorial requirements. A laborious task in manually editing source transcripts is the compilation of *indices* which historians traditionally use as "textual databases" — *e.g. chronological indices* (recording the occurrence in the original source of dates or other chronological data), *toponymical indices* (referring to toponyms or other geographical data), *anthroponymical indices* (collecting every reference to people) *etc*. Therefore, special emphasis was put on automatic compilation of these indices.

LATEX [11] was chosen as the underlying text preparation system for several reasons [3]:

- LATEX itself is a structured mark up language;
- LATEX has widespread over UNIX, MS/DOS and **MacIntosh** enviroments;
- its underlying formatting system (TEX) is of a good standard;
- its openness suggests its use as a versatile target language.

The *HiTeX format* front-end processor is written in LEX,YACC [23] and generates standard LATEX for both text and indices. The latter are compiled from textual *labels* accepted by the *HiTeX format* (marking the ocurrence of index items) but expunged from the output LATEX code.

A description of the *HiTeX format* and its processor can be found in reference [17].

The project's second phase was devoted to *knowledge representation*. The indices produced by the *HiTeX format* processor could have been regarded as a form of (textual) knowledge representation. We agree with Rahtz *et al* [19] about books being *"the primary repository of past knowledge; any new form of presentation must have some continuity with existing methods of knowledge representation"*. However (and again agreeing with the same authors), such indices are flat, unstructured records of textual information which ignore the intertwined texture of human knowledge. Moreover, their quality standard would remain highly dependent on authorship.

Bearing compatibility in mind, the system's design evolved towards regarding such indices not as knowledge sources, but rather as by-products of a knowledge-based system. That is, we expected indices to be automatically produced as filtered, textual "dumps" of some knowledge-base.

Design of the HITEX knowledge-base (HKB for short) was influenced by both the AI *semantic network* paradigm [6] and the recently so much favoured *object oriented* paradigm [7]. The resulting *data model* chosen to mould historical knowledge in HITEX is described in section 3. Special attention was paid to its mathematical definition, due in

[2] For instance, the report of the *Inquirições de Génere* database, recording so far $\simeq$ 30% of about 80,000 ordainment processes (17c–19c).

[3] We became aware later on of a similar choice in the LACE project [19].

part to the unsecure ‘ad hoc’ presentation of such paradigms in the computing literature (Wolczko’s work [24] is among the outstanding exceptions).

The original formal specification of HKB (written in the VDM notation [10]) can be found in reference [17]. It has remained a standard reference document throughout project design. The impact of later revisions was carefully analysed and this analysis greatly simplified by the unambiguous semantics of the formal specification. This was tested via its corresponding functional prototype executable on the XMETOO shell [18].

Recently, a more user-friendly prototype was encoded in SMALLTALK [3] which takes advantage of this language’s object-oriented semantics [7]. This prototype’s user interface (described in section 5) provides on-line access to source image files obtained from an IBM 3117 scanner and archived in an IBM 3363 optical disk device, *cf.* reference [20].

3 The HITEX Data Model

As suggested above, the architecture of the HITEX knowledge-base (HKB) resembles, in many respects, a hierarchical semantic-network or an object-oriented system. Historical information is recorded in terms of information “granules” which have a unique identity, that is, which can be referred to by quoting their unique name. The RHC acronym (for “re-usable historical component”) will be used to denote such information granules. RHC is HITEX’s most primitive entity for building up historical knowledge.

We will write the following expression,

$$HKB = Name \hookrightarrow RHC \tag{1}$$

as a shorthand for the sentence *“the* HITEX *knowledge-base (HKB) maps* Names *to* RHCs*”*. The $\hookrightarrow$ -symbol is intended to mean that not every name n in $Name$ has an associated $RHC\ r_n$ (think of people, by analogy: it is easy to think of a name which is not the name of anybody).

It is up to the user to decide how fine or coarse information units RHCs should be. Consider the following text fragment which starts folio 1 of the first volume of the ADB *Index das Gavetas* (part of a collection of manuscripts once belonging to the chapter of Braga’s cathedral church):

> *Certidão da doação que o arcebispo de Braga D.Martinho de Oliveira fez ao Cabido de Braga [. . .] Ano de 1300.*
>
> (Certificate of the donation by Martinho de Oliveira archbishop of Braga, to the chapter of Braga [. . .] Year 1300.)

A meticulous analysis of this sentence fragment reveals the presence of the following information granules:

- *Martinho de Oliveira* was the archbishop of Braga in 1300;

- this archbishop endowed the *chapter of Braga's cathedral* with something, in the same year;
- the chapter of Braga's cathedral kept a *certificate* of this endowment;
- such a certificate is archived in the ADB *Gavetas do Cabido* collection;
- a reference to the same certificate can be found on fol.1,vol.1 of the corresponding *Index* (a series of six volumes compiled in the 18c).

How should we record these RHCs in a flexible, incremental way? We need to know what an RHC actually "is".

HiTeX follows the object-oriented philosophy of associating *types* or *classes* to RHCs. Let us attempt a classification for the above items:

- *Martinho de Oliveira* → *Arcebispo* (Archbishop);
- *cabido de Braga* → *Cabido* (Chapter);
- *doação* ... → *Contracto* (Contract);
- *certidão da doação* ... → *Certidão* (Certificate).

Now our question is: what "is" a class? For instance, what is a *Certificate*? Every certificate is surely a *document*. But it is a *document* of a particular kind — a formal (public) document. Other documents may be referred to in the sequel of the above text fragment which are not formal, *e.g.* private letters. So we are tempted to introduce a few more classes helping us to "frame" our (so far sketchy) notion of a certificate:

$$\begin{array}{l}\textit{Documento}\\ \text{(Document)}\end{array} \left\{ \begin{array}{l} \begin{array}{l}\textit{Particular}\\ \text{(Private)}\end{array} \left\{ \begin{array}{l} \vdots \\ \begin{array}{l}\textit{Carta}\\ \text{(Letter)}\end{array} \\ \vdots \end{array}\right. \\ \begin{array}{l}\textit{Formal}\\ \text{(Formal)}\end{array} \left\{ \begin{array}{l} \vdots \\ \begin{array}{l}\textit{Notarial}\\ \text{(Public)}\end{array} \left\{ \begin{array}{l}\vdots\\ \begin{array}{l}\textit{Certidão}\\ \text{(Certificate)}\end{array}\\ \vdots\end{array}\right. \\ \vdots \end{array}\right. \\ \vdots \end{array}\right. \tag{2}$$

In summary, we have sketched a class *taxonomy* for documents, much in the same way biological species have been classified by biologists in the past. Of course, we do not claim that (2) is the "best" taxonomy framing the notion of a certificate! We simply want to show that building an RHC taxonomy may be a helpful way of telling what a given RHC "is" and "is not", due to the implicit *subsumption ordering*. In this way, we may talk about *subclasses* subsumed under *superclasses* (*e.g*. *Certificate* is a subclass of *Public*, *Document* is a superclass of *Private*).

Taxonomies can be established by recording the intended relationship between *class names*. The HiTeX block which records the system's current taxonomy is $HTax$, whose model is

$$HTax = CName \hookrightarrow CName \tag{3}$$

That is, $HTax$ records, for each class name ($CName$), the name of its immediate class ancestor. The overall subsumption ordering is obtained by transitivity. For instance, $HTax$ will map *Certificate* to *Public* and *Public* to *Formal*. It follows that *Certificate* transitively "maps" to *Formal*, and so on.

Is a class name hierarchy such as (3) enough to make up a taxonomy in the usual sense of the word? From Biology or Linguistics (the linguistic counterpart of a *class* is a *syntatic category*) we know that classes are not atomic entities but rather *feature bundles* (as originally suggested by Chomsky) [21]. Some classes exhibit features which others do not. For instance, a *Public* document is produced by a known *Notary-public*, but it is meaningless to say the same thing about a private *Letter*. That is, features provide the actual differentiation between classes. Subclasses inherit all features of their superclasses, and may exhibit additional features of their own. For instance, if we associate a *date* to every document of class *Document*, then such a feature, or attribute, will be inherited by every class in (2). The *Notary-public* feature of *Public* is inherited by *Certificate*, but *Certificate* may have specific features which cannot be found in every *Public* document. In this "feature factorization" resides the essence of a class taxonomy.

Under the above *inheritance assumption*, it suffices to revise $HTax$ (3) by adding, for each class name, the set of features it exhibits, which are not exhibited by any of its ancestor classes:

$$HTax = CName \hookrightarrow CName \times Features \tag{4}$$

The $CName \times Features$ subexpression means that each class name is associated to *not only* its immediate ancestor class name ($CName$) *but also* to its specific features ($Features$).

Now we need an expression formalizing how features are prescribed for each class:

$$Features = FName \hookrightarrow CName \tag{5}$$

According to (5), a set of features is a classified collection of *feature names* ($FName$), that is, an association of a feature name fn to the class name c_{fn} of its expected values. For instance, class name *Public* will be mapped not only to its immediate ancestor class name *Formal*, but also to the association of its feature name *#Notary* to a new class, *Notary-public* [4]. We will write

$$\begin{array}{l} Cl \\ \quad \#\mathit{Feat_1} : Cl_1 \\ \quad \vdots \\ \quad \#\mathit{Feat_n} : Cl_n \end{array}$$

wherever we want to stress that class name Cl has n features of its own, $Feat_1, \ldots, Feat_n$ ranging over classes $Cl_1, \ldots, Cl_n$ respectively, *e.g.*

$$\begin{array}{l} Public \\ \quad \#\mathit{Notary} : \textit{Notary-public} \end{array}$$

or

$$\begin{array}{l} Document \\ \quad \#\mathit{Date} : Date \end{array}$$

Let us now come back to the HiTEX knowledge-base HKB (1) and see a simple way of classifying RHCs (recall that this has been our motivation to build up a taxonomy). To each RHC identifier we attach the name of its intended class, obtaining

$$HKB = Name \hookrightarrow RHC \qquad (6)$$

$$RHC = CName \times \cdots \qquad (7)$$

That is, every RHC may be regarded as a particular *instance* of a class (which must be a valid class, *i.e.* archived in $HTax$).

It remains to fill in the "$\cdots$" in (7) showing how different instances of the same class are recorded. Think of a class *Indivíduo* (*Individual*) endowed with features such as

$$\begin{array}{l} Individual \\ \quad \#\mathit{BirthDate} : Date \\ \quad \#\mathit{Name} : CharList \\ \quad \#\mathit{Sex} : MaleFemale \end{array}$$

Two particular individuals will naturally differ from each other *wrt.* the particular values they exhibit for the features allowed by their common class (*Individual*), *e.g.* a different sex, same sex but different name, a different birth-date *etc.* [5]. So the core of an RHC of class Cl is its particular collection of specific values instantiating features exhibited by Cl in the underlying taxonomy. So we may resort to the mapping ($\hookrightarrow$) notation once again and complete (7) as follows:

$$RHC = CName \times Particularities$$

[4] For improved readability, feature names will be prefixed by character "#".

[5] Of course, different individuals may happen to exhibit exactly the same values for all features, but this only means that the available features provide *too abstract* a characterization of *Individual*; *passport number* is an example of a feature which would provide a finer distinction.

where $Particularities$ is defined by

$$\begin{array}{rcl} Particularities & = & FName \hookrightarrow Value \\ Value & = & AtomicValue + Name \end{array} \tag{8}$$

The "$+$" symbol in (8) means alternative (*either ... or ...*) definition: a particular value ($Value$) is associated to a feature name ($FName$) which is *either* an atomic value ($AtomicValue$, *e.g.* a number, a character *etc.*) *or* an RHC name ($Name$), *i.e.* a reference to another RHC.

In summary, RHCs may refer to each other by means of feature instantiation. In this resides the "networked" (or "linked") structure of HKB, *cf.* semantic networks [6].

A few comments on (8) are needed:

- *Type checking*: it is illegal for a particular RHC to provide instantiations for features which are absent from its class definition.
- *Incomplete information*: by contrast, a particular RHC need not provide instatiation for *every* feature present in its class definition, since some of its values may be yet unknown; the missing data may eventually become available from historical evidence gathered later on.
- *Definedness*: it is illegal for a particular RHC to refer to other non-existing RHCs; potential problems arising from circular cross-references are solved by incremental feature instantiation (*i.e.* declare RHCs first and add features later on).
- *Interfacing*: RHC "re-usability" arises naturally from the underlying taxonomy, which provides classified *standards* for the existing RHCs.

Our brief presentation of the HITEX data model ends by writing

$$HiTeX = HTax \times HKB \times \cdots$$

conveying the idea that HITEX is made up of at least two consistent basic blocks, a *taxonomy* (4) and a *knowledge-base* (6). Putting all definitions together we obtain

$$\begin{array}{rcl} HiTeX & = & HTax \times HKB \times \cdots \\ HTax & = & CName \hookrightarrow CName \times Features \\ Features & = & FName \hookrightarrow CName \\ HKB & = & Name \hookrightarrow RHC \\ RHC & = & CName \times Particularities \\ Particularities & = & FName \hookrightarrow Value \\ Value & = & AtomicValue + Name \end{array} \tag{9}$$

This collection of definitions may be regarded as a simplified *formal definition* of part of the HITEX system. In fact, the symbols $+$, $\times$, $\hookrightarrow$ *etc.* are not casually chosen — they belong to the specification notation SETS [14] and have a very precise, mathematical meaning (intentionally ignored above) which can be reasoned about [6].

[6] The SETS notation is similar to the VDM notation.

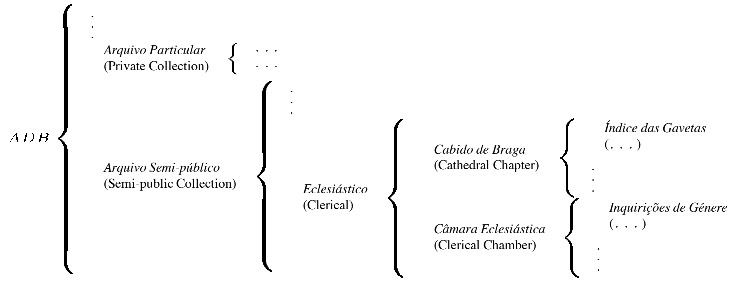
Mathematical reasoning is the only way of guaranteeing the correctness and compatibility of HITEX's data when ported to other implementation environments. Thus our emphasis on formally defining HITEX.

The actual formal definition of HITEX can be found in [17]. It is — of course — much larger than the above model, which was made simpler for ease of explanation [7].

Some of the technical elaborations of the actual system's model which are of some interest to the historian are briefed below.

- *Multiple instantiation*: RHCs may be instances of more than a single class, each class mirroring different views, complementary descriptions or even distinct historical phases of the same entity — *e.g.* a nobleman who is a notary-public, a bishop who becomes a member of some cathedral church chapter *etc*.
- *Unification* (a particularly useful device for incremental gathering of historical knowledge): if two seemingly different RHCs are found eventually to be complementary views of the same entity, this device provides for their consistent aglutination.
- *Structured features*: practical application of HITEX has suggested that some features possess some internal structure, that is, their values are not single entities as above but rather *collections of Values* which dynamically vary from RHC to RHC within the same class (*cf. e.g.* many-to-many relationships such as property limits). In the actual HITEX system, a $Value$ may take the form of a *set* or *list* of $Values$, or even a mapping from $keys$ to $Values$.
- *Cross-references*: a final facet of the HITEX model is source cross-referencing. Every class is by default endowed with a set-valued feature which is intended to record the identifiers of all sources which are known to refer to a particular RHC. Wherever a source transcription is performed in the HITEX environment, updating of such features is automatic, see section 5.

Note that the HITEX's taxonomy itself can be used for catalogue/cross-referencing purposes, that is, the archivist's data and the historian's data may be merged together in the same conceptual framework. See below a sketch of ADB's structure expressed by a HITEX taxonomy fragment:



[7] Many formal properties of HITEX other then *definedness*, *type checking etc*. have been left out.

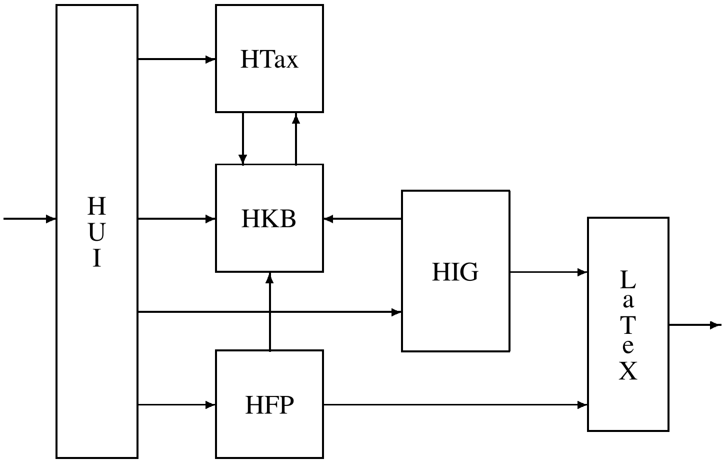


Figure 1: Block diagram of the HiTeX prototype.

Every archival reference may be regarded as an instance of class

$$\begin{array}{l} ArchReference \\ \quad \#To : Source \\ \quad \#InFolio : Folio \end{array}$$

where

$$\begin{array}{l} Source \\ \quad \#InCollection : ADB \\ \quad \#NrOfFolios : Number \end{array} \left\{ \begin{array}{l} LooseDocument \\ BoundVolume \left\{ \begin{array}{l} LooseVolume \\ VolumeInSeries \\ \quad \#VolNumber : Number \end{array} \right. \end{array} \right.$$

and so on.

4 The HiTeX System Structure

The structure of the HiTeX system's current prototype is shown in the block diagram of Figure 1.

User communication with the system is provided by the HUI block (HiTeX User Interface), see section 5 for details. Much of the user's input is concerned with creating and maintaining the system's RHC taxonomy ($HTax$) and knowledge-base (HKB) where the bulk of HiTeX's data is archived (*cf.* section 3).

The other system blocks are concerned with source transcription. HFP is the *HiTeX format* syntax processor. It outputs LaTeX code (recall that LaTeX is the default text preparation system of HITEX). All RHC names (labels) annotating a HITEX source file are removed but their information is used to update archival cross-references in HKB (*cf.* section 3). Finally, the HIG block (HITEX Indices Generator) may be invoked to generate a LaTeX encoding of a hierarchical, $HTax$-driven traversal of HKB. This collects textual descriptions of all RHCs whose names have been found as labels in a particular source transcription. This corresponds to the generation of textual indices referred to in section 2. Reference [15] contains an illustration of index generation concerning the first volume of the *Índice das Gavetas*.

5 The User's Perspective of HITEX

The current prototype version of the HITEX system (encoded in SMALLTALK and C) provides for user interface facilities concerned with

- Knowledge-base house-keeping.
- Access to document facsimiles from optical disk memory.
- Source transcription.

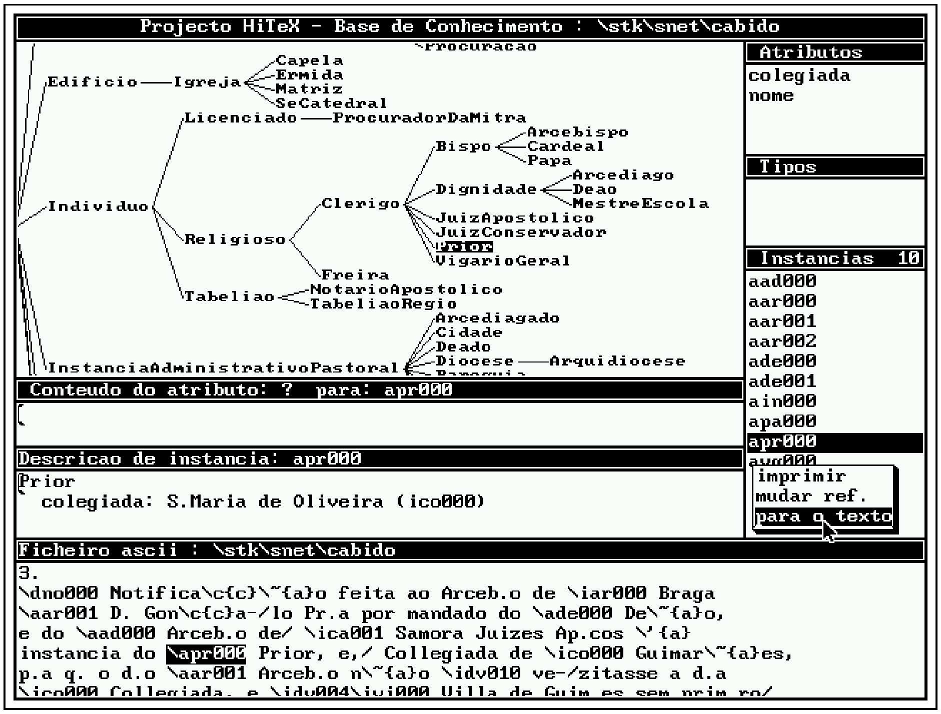
Knowledge-base house-keeping comprises facilities such as

- pictorial browsing of the system's taxonomy;
- search for a given RHC name;
- enter a new RHC of class currently selected from the taxonomy (RHC names are automatically generated);
- instantiate features in currently selected RHC;
- specialize/abstract a given RHC (*i.e.* push it down/up the taxonomy);
- add a new class to the system's taxonomy;
- add new features to currently selected class;
- query the knowledge-base (type-assisted construction of syntactically correct queries which may be run to obtain knowledge-base reports [8]).

etc.

Figures 2 and 3 depict snapshots of the prototype. The upper-left window depicts the system's current taxonomy. Conversation with the user is driven by pop-up menus.

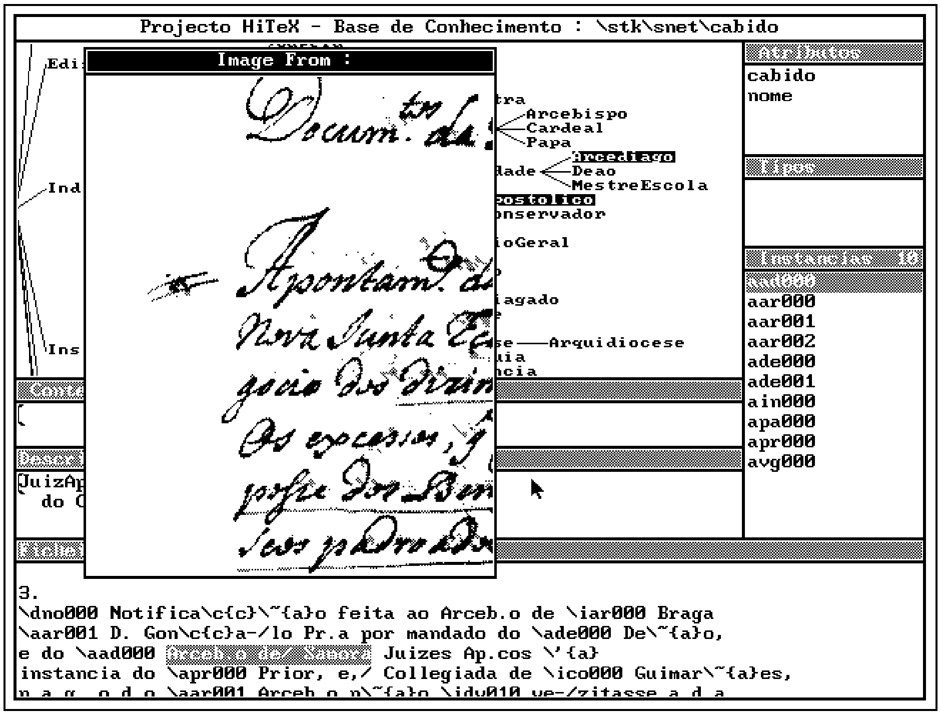
[8] This facility, available in the XMETOO prototype, has not yet been incorporated in the SMALLTALK prototype.



```
Descricao de instancia: apr000
Prior
  colegiada: S.Maria de Oliveira (ico000)
```

```
Ficheiro ascii : \stk\snet\cabido
3.
\dno000 Notifica\c{c}\~{a}o feita ao Arceb.o de \iar000 Braga
\aar001 D. Gon\c{c}a-/lo Pr.a por mandado do \ade000 De\~{a}o,
e do \aad000 Arceb.o de/ \ica001 Samora Juizes Ap.cos \'{a}
instancia do \apr000 Prior, e,/ Collegiada de \ico000 Guimar\~{a}es,
p.a q. o d.o \aar001 Arceb.o n\~{a}o \idv010 ve-/zitasse a d.a
\ico000 Collegiada, e \idv004\ivi000 Villa de Guim es sem prim ro/
```

Figure 2: Snapshot of the HITEX prototype.



```
3.
\dno000 Notifica\c{c}\~{a}o feita ao Arceb.o de \iar000 Braga
\aar001 D. Gon\c{c}a-/lo Pr.a por mandado do \ade000 De\~{a}o,
e do \aad000 Arceb.o de/ Samora Juizes Ap.cos \'{a}
instancia do \apr000 Prior, e,/ Collegiada de \ico000 Guimar\~{a}es,
n a q  o d o \aar001 Arceb.o n\~{a}o \idv010 ve-/zitasse a d a
```

Figure 3: Access to image files from the HiTeX prototype.

The lower window is concerned with source transcription. It may be regarded as a simple *text-editor* which communicates with the knowledge-base wherever required. Note that source files may be edited using other text editors and then up-loaded by this HiTeX editor, provided that they do not contain obscure character controls. However, cross-reference labelling is semi-automatic using the system's editor: leaving the cursor in the required position, the user may search the knowledge-base for the relevant RHCs. A mouse click (*cf.* the `para o texto` *i.e.* "to text" menu entry in Figure 2) will be enough for the selected RHCs' names to be automatically inserted in the text-window right after the cursor position.

Source transcription texts must obey the standard *HiTeX format* syntax [17]. No syntax-direction is provided in the current prototype yet. The invoking of HiTeX's HFP unit (see section 4) will automatically pick up the current source transcription file, which is then passed on to the *HiTeX format* translator which generates LaTeX code, as well as an internal commands file which may be run to automatically update cross-references in the HiTeX knowledge-base, affecting the involved RHCs.

6 Related Work

Computers have often been used to help historians. Significant use has been made of traditional databases and statistical packages, *cf. e.g.* references [1, 2]. However, the facilities provided by traditional tools are still weak when handling the complexity of information which is relevant to the historian.

The development of the HITEX software is one among many R&D efforts to design more helpful tools for the historical disciplines. An approach similar to HITEX's in many respects is followed by the MICROCOSM system [8]. Departing from the hypertext paradigm, MICROCOSM's philosophy is that "*hypermedia links in themselves are a valuable store of knowledge*". Such as in HITEX, complex links are not stored *in* the documents but rather separated out into a knowledge-base. MICROCOSM's global *universe of discourse* bears some relationship with HITEX's *taxonomy*. The MICROCOSM project has produced two parallel implementations, one in MS-windows and another in the Actor object-oriented language [8].

Another system design sharing HITEX's motivations is LACE [19]. This is another *open* hypertext front end to documents, running on Sun workstations and consisting of LaTeX (the underlying medium for document creation), NeWs (a PostScript-based windowing display system) and a (hypermedia) document library server.

Work related to HITEX has also been developed by Barroca [5] on object-oriented database design in archæology. Her emphasis is more on *method* than on *system design*, in a way similar to the HITEX's discipline for incrementally managing historical data.

7 Conclusions and Future Research

The HITEX project at the District Archives of Braga has nearly reached the end of its second phase by delivering a system software prototype encoded in the SMALLTALK and C languages. An earlier prototype is also available which "animates" the system's standard formal specification. This prototype was straightforward to derive from the specification but it is hard to use by the historian because of its poor user-interface. Moreover, its performance degrades exponentially as the system's knowledge-base grows up. Testing this prototype over realistic data could hardly be made on a Sun SPARC workstation (see reference [15] for a brief performance evaluation).

We believe the SMALLTALK prototype to be actually usable by the historian, since its user-interface is much improved. However, it is hard to predict how far we can go with this prototype *wrt*. knowledge-base size. The transcription of the *Índice das Gavetas* 6 volume series is in progress and will provide a definitive test of the system.

The HITEX project's third phase will start as soon as this prototype is extensively tested and assessed by the project's historians team. Some improvements are likely to be needed concerning the system's *deductive power* (*cf.* the logics of *aggregation*, *composition*, *grouping etc*. proposed by Hsieh [9]) and *textual index* production (fully automatic natural language generation will require an investment in computational linguistics [21]).

This last phase is planned to encompass a further significant step in integrating conventional technology in the system, *e.g.* relational (large) database support, access to SPSS [13] *etc*. Its novelty will reside in the application of formal methods to the implicit implementational challenge (recall the gap between specifications and implementations). We intend to "calculate" the underlying database schemata directly from HITEX's taxonomies, by using the SETS formal calculus [14], which guarantees correctness and compatibility. The main problem will probably reside in achieving a stable taxonomy for historical knowledge, widely accepted by the historians' community.

Acknowledgements

The authors express their thanks for the support of Dr. Maria Assunção Vasconcelos (ADB Director) and of Mrs. Leónida Gomes (ADB staff). Special thanks go to the project software developers Mané, Ramalho and Pedro.

The HITEX software has been developed for the IBM Microcomputer Laboratory of ADB. Access to the network of Sun workstations of UM/INESC is gratefully acknowledged.

The HITEX Project has been funded by the Calouste Gulbenkian Foundation under grant Nr.E/75/88.

References

[1] Actes du IV^{e} Congrès "History and Computing'89" *L'Ordinateur et le Métier d'Historian*. CNRS, Travaux et Documents 2, Maison des Pays Ibériques 1990.

[2] Amado J.P., Cardoso J.C., Neves A.M. *Hϵródoto 1.0 — Estação de Trabalho Informática em História e Arquivística*. Fac. de Ciências Sociais e Humanas da U.N.Lisboa, 1990 (in Portuguese).

[3] Andrade P. Relatório de Estágio, LESI, U.Minho, Braga 1991 (in preparation).

[4] Association for Computing Machinery. *Special Issue on Hypertexts*. ACM Comm., 31(7), Jul. 88.

[5] Barroca L. *Object-oriented Database Design in Archæology*. Dept. of Computer Science, University of Minho, 1991.

[6] Brachman R.J. *On the Epistemological Status of Semantic Networks*. in *Associative Networks: Representation and Use of Knowledge by Computers*. N. V. Findler, New York: Academic Press, 1979.

[7] Goldberg A., Robson D. *Smalltalk'80: the Language and Its Implementation*. Addison-Wesley 1983.

[8] Fountain A., Hall W., Heath I., Davis H. *MICROCOSM: An Open Model for Hypermedia With Dynamic Linking*. Technical Report CSTR 90-12, Dept. of Electronics and Comp. Science, Univ. of Southampton, 1990.

[9] Hsieh D. *A Logic to Unify Semantic Network Knowledge Systems with Object-oriented Database Models*. Technical Report SRI-CSL-90-15, SRI International, Dec. 1990.

[10] Jones C.B. *Systematic Software Development Using VDM*. Prentice-Hall, 1986.

[11] Lamport L. LaTeX — *A Document Preparation System*. Addison-Wesley Publishing Company, 5^{th} edition, 1986.

[12] Lange D.B. *A Formal Approach to Hypertext Using Post-prototype Formal Specification*. Lecture Notes in Computer Science, Vol.428, 99-121, Springer-Verlag, 1990.

[13] Norman H.N., Hull C.H., Jenkins J.G., Steinbrenner K., Bent D.H. *SPSS—Statistical Package for the Social Sciences*. McGraw Hill, 2nd edition, 1975.

[14] Oliveira J.N. *A Reification Calculus for Model-Oriented Software Specification*. Formal Aspects of Computing, Vol.2, 1-23, 1990, Springer-Verlag.

[15] Oliveira J.N. *Projecto HiTeX — 2º Relatório de Progresso*. Arquivo Distrital de Braga, Univ. do Minho, Aug. 1990 (in Portuguese).

[16] Paiva J.P. *O Que é o HyperTexto*. In Boletim da Assoc. Portuguesa de História e Informática, 1, 4-9, Mar. 1989 (in Portuguese).

[17] Pereira, A.M. *HiTeX — Um Sistema para Transcrição Documental em Larga Escala*. Relatório de Estágio, LESI, U.Minho, Braga 1991 (in Portuguese).

[18] Pinto A. *xmetoo — metoo Estendido sobre a Máquina XLISP*. Manual de Utilização, ver.2, 1989 (in Portuguese).

[19] Rahtz S., Carr L., Hall W. *New Designs for Archæological Reports*. Science and Archæology, Vol.31, 20-34, 1989.

[20] Ramalho J.C. Relatório de Estágio, LESI, U.Minho, Braga 1991 (in preparation).

[21] Ravera S.B. *Information-Based Linguistics and Head-Driven Phrase Structure*. In *Natural Language Processing*, Lecture Notes in Artificial Intelligence, Vol.476, 55-101, Springer-Verlag, 1990.

[22] ROSE Consortium — E.S.F. *The ROSE Subproject*. Presentation at the Workshop on Reuse of Software Development Components, Joint Organization CEC-ESF, Berlin, 25-26 October 1989.

[23] Tare R.S. UNIX *Utilities*. McGraw-Hill International Editions, 1986.

[24] Wolczko M. *Semantics of Object-oriented Languages*. Tech. Report UMCS-88-6-1, Univ. of Manchester, 1988.