INTERVISION: A New Hypermedia System  
Focusing on Dynamic Media  

Hideto IKEDA*, Fumio KITAGAWA*, Masaki NAKAMURA** and Kenzo UCHIYAMA**  

* Information Processing Center, Hiroshima University, 1-1-89 Higashi-Senda, Naka-ku, Hiroshima 730 Japan  
** Information Systems Development, Tokyo Electric Power Co., INC., 1-4-10 Iriyama, Chuo-ku, Tokyo 104 Japan  

Abstract  
In order to implement storage and manipulation of multimedia information, especially full-motion videos, a new data model and a new language are required, because they should represent complex data of not only traditional data, characters and numerals, but also new media such as images, long texts, sounds and full motion videos. Data model B5 that is proposed in this paper is a large sheet with fixed five columns, that is, value, data type, entity, attribute and time. Queries, integrity and security for data in B5 are expressed first-order predicates. In spite of such a simple structure, it has very flexible and strong capabilities for data expression and manipulation. This paper describes how to express various data in B5 including complex data and propose a new language INTERVISION which is specially designed for implementing interactive videos on B5. A prototype system of INTERVISION is also presented.  

1 Introduction  
Developers of hypermedia systems face many design issues [6]. Some of these issues have been discussed in Conklin's summary [1] of the hypertext field and in papers describing specific systems, such as Intermedia[3], NoteCards[4], Neptune[2], HyperTIES[4] and KMS[6]. From these discussions, we can summarize the current problems of hypermedia systems to be as follows:  

- Data Model Issue  
Hypermedia is a generalization of the hypertext concept, now over 40 years old. Although there is no generally accepted definition for hypermedia, the main concept concerns the kind of data model used to implement hypermedia systems. Since a data model determines the internal structure of the software system, it strongly influences the complexity, capability, performance, and cost of system development. The problems are compounded when one must cope with multiple systems with inconsistent data models. Therefore, one of the most important issues for designers is to establish a simple and comprehensive data model. Data model issues include a general view of data (data structure), integrity control, query, editing, versioning, and extensibility.  

- User Interface Issue  
A good user interface is very important for users to communicate with computer interactively. The user interface is closely related to the data model from the point of implementation and performance. It, however, should be independent from functional capabilities of the data model. The user interface provides users a view of the system, a means of search and manipulation of information, and a way to control of system behavior. The user interface also gives users a certain degree of flexibility to tailor the application to suit their own needs. Support for collaborative work is also a significant issue of user interface.  

- Implementation Issue  
In order to implement hypermedia systems, some different devices other than traditional computer peripherals and accessories are needed. Some of implemented systems use DVI, CD-I, IVD and CD-XA as interactive physical media, and digital PBX for multimedia data communication. But, this equipment is not enough to implement large databases of dynamic media, especially with joint authoring capability and shared by multiple users. These problems are left for the future research and technology development.  

Here, we introduce a prototyping hypermedia system - INTERVISION which specifically focuses on authoring application and interactive use of dynamic media information. The initial version of INTERVISION was an experimental system to evaluate an unique data model B5 which was proposed in 1988 [8, 10]. The current INTERVISION is equipped with an object-oriented language to edit interactive video. This paper introduces the data model B5, a user interface for the authoring system, and an implementation of INTERVISION. It also discusses the remaining problems of hypermedia systems in brief.  

2 Data Model B5  
INTERVISION system has a database which consists of a set of records with the same format as:  

\[
(VALUE, [DATA-TYPE], [ATTRIBUTE], [ENTITY], [PERIOD])
\]

Each record expresses a fact of a real world. For ex-
ample, record (‘IKEDA’, [C], [NAME], [DASFAA’91-ATTENDEE], [04/02/1991-04/04/1991]) means that “character string ‘IKEDA’ is a name of an attendee of DASFAA’91 and the fact is true during April 2, 1991 to April 4, 1991”. This data structure can accept all information of the real world as a collection of records with this fixed format. Each of the five columns is essential to express information with a definite meaning. By digital technologies, we can express any information as a bit string. Such a bit string is stored in VALUE-column. But a bit string alone is not enough because two or more different types of information may have the same bit string. So, specification of DATA-TYPE is necessary. In order to express the meaning of the value in the real world, ENTITY and ATTRIBUTE are essential. (This is one of significant results of database researches for 25 years.) PERIOD is also important to support historical databases, versioning, time series and dynamic media information. Therefore, none of the above five columns can be eliminated. This data structure is referred as B5 (Basic 5-column table) in the following.

A possible value of DATA-TYPE is recursively defined from the set $T_0$ of all basic data types and three basic operations. These basic data types are: A(address), C(character string), N(numeral), L(logical), P(time), O(operation), G(vector graphics), I(bit-mapped image), S(sound), and M(full-motion video). These three basic operations are set $\{*,*, \ldots\}$, list $(*,*,\ldots)$, and combination $(*,*,\ldots)$. The definition of data type is as follows:

**Definition 1 (Data Type)**

1. Each element of $T_0$ is a data type,
2. If $t$ is a data type, $\{t\}$ and $(t)$ are data types,
3. If $t_1, t_2, \ldots, t_n$ are data types, $(t_1, t_2, \ldots, t_n)$ is a data type.

The set $T_0$ of all basic data types must include $A, C, N, L, P$ and $O$. The other basic data types can be specified depending upon applications. Some other basic data types can also be defined such as $W$(double-precision real number) and $V$(analog full-motion video), and so on. Since some basic data types depend on the physical devices available in the system, the discussion of data model B5 will not be limited to a specific set $T_0$. The set of all data types is denoted by $DT$ in this paper.

Assume that the set of all possible values, denoted by $D_0(t)$, is associated to each basic data type $t$. By using set $\{D_0(t) \mid t \in T_0\}$ and the above three operations, the set of all possible values $D(t)$ for a general data type $t$ is defined as follows:

**Definition 2 (Domain)**

1. For a basic data type $t$, $D(t) = D_0(t)$.
2. For a data type $t$, $D(\{t\}) = 2^{D_0(t)} \text{, i.e., the set of all subsets of } D_0(t)$.
3. For a data type $t$, $D((t)) = \bigcup_{k=1}^{n} D(t_k) \text{, i.e., the set of all lists of } D(t_k)$.
4. If $t_1, t_2, \ldots, t_n$ are data types, $D(t_1, t_2, \ldots, t_n) = \prod_{k=1}^{n} D(t_k) \text{, i.e., the product set of } D(t_1), D(t_2), \ldots, D(t_n)$.

A user can define data type by a character string, say $d$, that is not included in $T(\cup_{t \in T_0} D(t))$ and associated to a subset of $D(t)$ for some data type $t$.

ENTITY and ATTRIBUTE are expressed by character strings. PERIOD is described by a disjoint time interval which specifies the starting time and the ending time, e.g., 04/02/1991 - 04/04/1991. Symbol $*$ is used for an unspecified time in time expressions. If the starting time and the ending time are equal, the period can be described by the starting time only, e.g., 04/02/1991 for 04/02/1991-04/02/1991.

In addition, expressions of DATA-TYPE, ATTRIBUTE, ENTITY, and PERIOD are also kinds of values. All values should be stored in VALUE-columns of the database. Redundancy should, however, be eliminated. In order to meet these requirements, which seems to be contradicted, an address of each column is stored. The address indicates where the actual value is. In the previous example of a B5 record, we used a symbol $[*]$ for expressing the address of record in B5 where value $*$ is stored.

**Definition 3 (B5 structure and records)**

A B5 database is defined as a collection of records satisfying the following axioms:

**<Axiom 1: Data Type>** If a record $(v, t, a, e, p)$ is in B5 then $v \in D(t)$ and $t, a, e, p \in A$.

**<Axiom 2: Self-Description>** If there is a record $(v, t, a, e, p)$ in B5, there is a list of records in B5 as follows:

1. $(a, [C], [COMPONENT], [SYSTEM], [p])$,
2. $(a, [C], [INSTANCE], [DATA-TYPE], [p'])$,
3. $(p, [P], [INSTANCE], [PERIOD], [p'])$.

**<Axiom 3: System record>** When B5 is initialized, there are some records in B5 which are called system records. Each system record cannot be changed or erased by users. A value of a system record is called a reserved word. Representative system records are listed in Table 1. Besides system records, there are also user records.
If a record \((v, t, a, [ENTITY], p)\) is in B5, then \(a = [INSTANCE]\) and \(t = [C]\).

If there is a record \((v, t, a, [ATTRIBUTE], e, [p])\) in B5, then \(t = [C]\) and there is a record \((c, [C], [INSTANCE], [ENTITY], b)\) in B5 so that \(p'\) covers \(p\). Furthermore, if there are two records \((v', t', [w], e', [p'])\), \((v', t', [v], e'', [p''])\) then \(t' = t'' = [C]\) and \(e' = e''\).

If there is a record \((v', t', a', e', [FUNCTION], p')\) in B5 and \(p'\) covers \(p\). Furthermore, if a record \((v', t', a', e', [RELATION], b)\) in B5 for each \(i = 1, 2, \ldots, n\) in B5, then \(v(v_i, v_i, \ldots, v_n)\) is an element of \(D(t'')\).

If there is a record \((v, t, a, [RELATION], p)\), then \(t = [C]\) and \(a = [INSTANCE]\) and there are determined uniquely four records
\[(v', [C], [DOMAIN], [v], [p]),
(v'', [C], [DATA-TYPE], [v], [p]),
(t', [DT], [DATA-TYPE], [v'], [p]),
(t'', [DT], [DATA-TYPE], [v''], [p]).\]

Furthermore if a record \((v', t', a', e', [FUNCTION], [RELATION], p)\) is in B5 and \(p'\) covers \(p\), then \(v'(u, v, v, \ldots, v_n)\) is an element of \(D(L)\).

The set of all value formulas on B5 is denoted by \(F\). If a value \(f\) is a value formula on B5 and \(x\) is an open variable, then \(f(x)\) is a value formula on B5. Among \(w_i's\), an element which is included in \(X\), is called an open variable of the relation.

1. If \((f, [C], [DATA-TYPE], [v], [p])\) is in B5 and \(w_i\) is in \(D(t_i(f))\) \(X\) for \(i = 1, 2, \ldots, n\) and \(f(w_1, w_2, \ldots, w_n)\) is a value formula on B5. Among \(w_i's\), an element which is included in \(X\), is called an open variable of the relation.

2. If \(f\) and \(f'\) are value formulas on B5, \(f \land f', f \lor f'\), \((f)\) and \(~f\) are value formulas on B5.

3. If \(f\) is a value formula on B5 and \(x\) is an open variable of \(f\), then \((\forall x)f\) and \((\exists x)f\) are value formulas on B5. In these value formulas, \(x\) is called a closed variable.

4. Each value formula on B5 is derived from finite applications of 1 through 3.

The set of all value formulas on B5 is denoted by \(F\). If a value formula has open variables, it is called an open value formula, else a closed value formula. A value formula is used to express the integrity and security in B5.

### Extension of Definition 3 (Integrity and Security)

#### <Axiom 9: Integrity>
If there is a record \((v, t, a, [INTEGRITY], p)\), then \(t = [C]\) and \(a = [INSTANCE]\) and there is a unique record \((v', [V], [CONDITION], [w], [p])\) in B5.

#### <Axiom 10: Security>
If there is a record \((v, t, a, [SECURITY], p)\), then \(t = [C]\) and \(a = [INSTANCE]\), and there are four records

### Table 1: List of Representative System Records

<table>
<thead>
<tr>
<th>Value</th>
<th>Data Type</th>
<th>Attribute</th>
<th>Entity</th>
<th>Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>#001</td>
<td>[SYSTEM]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#002</td>
<td>[COMPONENT]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#003</td>
<td>[VALUE]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#004</td>
<td>[DATA-TYPE]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#005</td>
<td>[ATTRIBUTE]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#006</td>
<td>[ENTITY]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#007</td>
<td>[PERIOD]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#008</td>
<td>[INSTANCE]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#009</td>
<td>[IDENTIFICATION]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#010</td>
<td>[FUNCTION]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#011</td>
<td>[RELATION]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#012</td>
<td>[INTEGRITY]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#013</td>
<td>[SECURITY]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#014</td>
<td>[CONDITION]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#015</td>
<td>[USER]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#016</td>
<td>[NAME]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#017</td>
<td>[CHARACTER]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#018</td>
<td>[NUMBER]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#019</td>
<td>[LOGICAL-VALUE]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#020</td>
<td>[OPERATION]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#021</td>
<td>[ADDRESS]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#022</td>
<td>[SOUND]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#023</td>
<td>[MOTION-VIDEO]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#024</td>
<td>[DOMAIN]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#025</td>
<td>[RANGE]</td>
<td>#051</td>
<td>#002</td>
<td>#001</td>
</tr>
<tr>
<td>#050</td>
<td>['*']</td>
<td>#051</td>
<td>#008</td>
<td>#007</td>
</tr>
<tr>
<td>#051</td>
<td>['C']</td>
<td>#051</td>
<td>#008</td>
<td>#004</td>
</tr>
<tr>
<td>#052</td>
<td>['N']</td>
<td>#051</td>
<td>#008</td>
<td>#004</td>
</tr>
<tr>
<td>#053</td>
<td>['L']</td>
<td>#051</td>
<td>#008</td>
<td>#004</td>
</tr>
<tr>
<td>#054</td>
<td>['G']</td>
<td>#051</td>
<td>#008</td>
<td>#004</td>
</tr>
<tr>
<td>#055</td>
<td>['T']</td>
<td>#051</td>
<td>#008</td>
<td>#004</td>
</tr>
<tr>
<td>#056</td>
<td>['P']</td>
<td>#051</td>
<td>#008</td>
<td>#004</td>
</tr>
<tr>
<td>#057</td>
<td>['VF']</td>
<td>#051</td>
<td>#008</td>
<td>#004</td>
</tr>
<tr>
<td>#058</td>
<td>['DT']</td>
<td>#051</td>
<td>#008</td>
<td>#004</td>
</tr>
<tr>
<td>#059</td>
<td>['O']</td>
<td>#051</td>
<td>#008</td>
<td>#004</td>
</tr>
<tr>
<td>#060</td>
<td>['A']</td>
<td>#051</td>
<td>#008</td>
<td>#004</td>
</tr>
<tr>
<td>#061</td>
<td>['S']</td>
<td>#051</td>
<td>#008</td>
<td>#004</td>
</tr>
<tr>
<td>#062</td>
<td>['M']</td>
<td>#051</td>
<td>#008</td>
<td>#004</td>
</tr>
<tr>
<td>#090</td>
<td>['=']</td>
<td>#051</td>
<td>#008</td>
<td>#010</td>
</tr>
</tbody>
</table>
Definition 5 (Record Formula)

A record formula is recursively defined as follows:

1. Any record in B5 is a record formula.

2. If \((v, t, a, e, p)\) is in B5, a tuple derived by substituting some of five columns to variables is a record formula.

3. If \(x\) is a variable included in a record formula \(r\), then \((\forall x)r\) and \((\exists x)r\) are record formulas.

4. If \(r\) and \(r'\) are record formulas, \(r \land r'\), \(r \lor r'\), \((r)\) and \(\neg r\) are record formulas.

The set of all record formulas is denoted by \(RF\). Without using record formulas, we can specify any subset of B5 by value formulae. But record formulas make it easier for users to specify data manipulations of B5. In such sense, we use the word 'formula' for a record formula or value formula.

Definition 6 (Select and Delete Operations)

Both a select operation and a delete operation are defined as a record formula and/or a value formula.

For the selection and deletion, the security and integrity conditions are checked respectively.

Although there can be many different syntactical expressions of queries, the essential components are the same as defined above. If a SQL-like expression is used, query "Show me the position of every employee whose name is IKEDA and age is greater than 40" as follows:

```
SELECT ALL RECORDS
WHERE ALL X ('IKEDA', *, *, X, *)
AND (X, X, 'IDENTIFICATION', 'EMPLOYEE', X)
AND (Y, X, 'POSITION', X, *)
AND AGE(x) > 40
```

Definition 7 (Insert and Update Operations)

An insert operation is defined as a list of records to be stored in B5. For an update operation, a pair of a list of records and a condition are expressed by a formula. The security and integrity conditions are also checked.

To define a theoretical data model, it is necessary to make more detailed specifications. Essential concepts, however, are presented as above. The basic purpose to develop INTERVISION is to verify that data model B5 whether it is sufficient to support hypermedia systems including interactive video.

3 Editing Interface of INTERVISION

A typical hypermedia editing interface is the desktop metaphor. The desktop metaphor as reigning human-computer interaction paradigm, however, attains its limitation when it is used to develop interactive video applications. To provide non-professional editors an easy way to edit interactive video data, an authoring language is designed in INTERVISION. It tries to use state-of-the-art technologies, and, at the same time, to simulate the traditional video editing methods.

The syntaxes of the authoring language are defined as follows. Figure 1 illustrates the basic structure of a SCENARIO.

(1) Basic Data Structure

```
<SCENARIO>

DEFINE SCENARIO
SC-NAME: scenario-name.
ENTRY:
scene-name IF condition GOTO scene-name,
ELSE condition GOTO scene-name,

...;

ELSE condition GOTO scene-name;
scene-name IF condition GOTO scene-name,
ELSE condition GOTO scene-name,
```

Figure 1: Basic Structure of a SCENARIO
ELSE condition GOTO scene-name;

scene-name IF condition GOTO scene-name,
ELSE condition GOTO scene-name,

ELSE condition GOTO scene-name.

PARAM:

parameter-name [= initial-value ] data-type;
parameter-name [= initial-value ] data-type;

parameter-name [= initial-value ] data-type.

<SCENE>

SCENE is defined by a collection of CUTs and PROCEDURES. A PROCEDURE is a function of SCENE.

DEFINE SCENE

S-NAME: scene-name.
SCREEN: screen-name.
CUT:

cut-name [- cut-name ] FROM albumn-name
ON window-name
BY how-to-playback;
cut-name [- cut-name ] FROM albumn-name e
ON window-name
BY how-to-playback;

PROC:

procedure-name(parameter, ...) ON window-name
IF condition,
procedure-name(parameter, ...) ON window-name
IF condition;

procedure-name(parameter, ...) ON window-name
IF condition.

<ALBUMN>

An ALBUMN is a library of CUTs with the same data type. So, each ALBUMN has a unique name in a SCENARIO.

DEFINE ALBUMN

AR-NAME: albumn-name
DT-TYPE: data-type.
ID-TERM: Identification.
DEVICE: device-name FROM address TO address,
DEVICE: device-name FROM address TO address,
DEVICE: device-name FROM address to address.

(2) Control Procedure of INTERVISION

The control procedure in INTERVISION is designed in the following way.

<SCENARIO>

After setting initial values to the parameters, the initial SCENE is loaded and shown. The system waits until one of the conditions is satisfied. If so the control transfers to the corresponding SCENE. If EXIT SCENE is reached, then the system halts.

<SCENE>

Starting a SCENE, the procedures with START condition are executed and then all CUTs are loaded and displayed in the SCENE in the corresponding windows. After that, the system waits until one of the conditions is satisfied. If satisfied, the corresponding procedure is executed. If the procedure with condition of “END” is executed, the control is returned to the SCENARIO.

4 Implementation of INTERVISION

(1) Representation of SCENARIO on B5

In order to implement INTERVISION, it is necessary to store scenarios into B5. It is important for us to evaluate data model B5 how efficient to support hypermedia systems such as INTERVISION. A scenario has a basic structure as follows:

( scenario-name,
  {(( scene-name, {(( condition, scene-name) ))}),
  {(( parameter-name, initial-value, data-type))})
Figure 2: SCENARIO - "SELECT-PROGRAM" in B5

For example, a SCENARIO named "SELECT-PROGRAM" with 3 scenes called MENU, PROGRAM-1, and PROGRAM-2, and 2 parameters P1 and P2 has a basic structure as

('SELECT-PROGRAM',
 ( MENU, ((P1 = 1, PROGRAM-1),
 (P2 = 2, PROGRAM-2 ))),
 PROGRAM-1, ( (TRUE, PROGRAM-2)),
 PROGRAM-2, ( (TRUE, MENU)) ),
 { (P1, 0, N),
 (P2, 0, N) })

Figure 2 is the way to store SCENARIO "SELECT-PROGRAM" in B5.

(2) Hardware and Software Configuration

There are two different types of INTERVISION. One is a stand alone system. The other is a multi-user system. The later is a server which is connected to clients. To concentrate on the major issues of this paper, the configuration of a stand alone INTERVISION is shown in Figure 3.

Hardware Configuration

In this configuration, the functions of each device is as follows.

<Control Computer> executes the software which controls each device. The detail is shown in the followings.

<Optical Video Disk Recorder> accepts NTSC Video signals, records 24,000 - 100,000 frames( 20 minutes through 2 hours for a full-motion video), and plays back each frame within 0.5 second. This device is used to implement ALBUMNs of still videos and full-motion videos.

<Magnetic disk device/optical disk device> are used to store software and data other than videos.

<Monitor> has input/output ports for NTSC and RGB, superimposer and RS232C interfaces.

<Video scanner> converts NTSC signal to digital data stream to generate image data in the frame memory.

<Frame memory> stores and converts digital image. It is controlled by the computer.

<Stream data device> is used to store and to backup data in magnetic disk.

<Voice input/output device> digitizes voice and transfers the data to the computer. And it also converts digital data into voice.

<Video switcher> selects input video signals.

<Port selector> switches RS-232C signals.

<Other AV devices> are used for original image input and playback.

By using hardware configuration in Figure 3, it is possible to develop application and to use them. Devices for editing and playback are indicated by * and + respectively.
Software Configuration of INTERVISION

On the basis of hardware configuration in the previous section, we will show the software configuration in Figure 4.

<table>
<thead>
<tr>
<th>Editing and data management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definitions of application</td>
</tr>
<tr>
<td>Text editor</td>
</tr>
<tr>
<td>ALBUM creation</td>
</tr>
<tr>
<td>Compiler &amp; Debugger</td>
</tr>
<tr>
<td>SCREEN compiler &amp; debugger</td>
</tr>
<tr>
<td>SCENE compiler &amp; debugger</td>
</tr>
<tr>
<td>SCENARIO compiler &amp; debugger</td>
</tr>
<tr>
<td>PROCEDURE compiler &amp; debugger</td>
</tr>
<tr>
<td>Dictionary management</td>
</tr>
<tr>
<td>Initialization/Registration/Update/Delete/</td>
</tr>
<tr>
<td>List/Copy/Login/Report</td>
</tr>
</tbody>
</table>

CUT editor
- Word processor
- Spreadsheet
- Graphics
- Image editor
- Audio editor
- Video editor

Figure 4: Software Configuration of INTERVISION

5 Conclusion and Discussion

This paper proposes a new data model B5 and a hypermedia system INTERVISION which is supported by B5 structure. It also discusses how to implement INTERVISION with currently available technologies.

For advanced applications, it is essential to establish a powerful data model that can support various requirements of hypermedia systems as well as traditional information systems. In addition, theoretical formulation of database concepts which we obtained historically is also very important, because there would have no more progress without it.

Data model B5 enables us to express complex data structure derived by operation, setting, listing, and combination. Such structural hierarchy plays very important role to express hypermedia information.

In addition to the representability of complex structure, it is also important to express the meaning of information. Each piece of information of B5 is attached to what it is by specifying attributes and is stored as a collection of atomic parts. This approach enables to realize high reusability of information. The structure of B5 make it simple to implement B5 as hardware also.

This paper is a trial to establish such a data model. The B5 data model can be used for management of hypermedia databases as well as traditional business databases. The description of B5 in this paper is not sufficient theoretically, and the detail will be presented in future papers.

References