An Example of a Business Application using the Search Function of the Cellular Data System

Tosiyasu L. Kunii
Morpho Inc.
Tokyo, Japan
kunii@ieee.org

Yoichi Seki
Software Consultant
Tokyo, Japan
gamataki51@hotmail.com

Abstract— In the era of cloud computing, data is processed in a “Cloud”, and data and its dependencies between systems or functions progress and change constantly within the cloud, as a user’s requirements change. Such information worlds are called cyberworlds. We need a more powerful mathematical background which can model the cyberworlds in the "cloud" as they are. We consider the Incrementally Modular Abstraction Hierarchy (IMAH) to be appropriate for modeling the dynamically changing cyberworlds by descending from the most abstract homotopy level to the most specific view level, while preserving invariants. We have developed a data processing system called the Cellular Data System (CDS) based on IMAH. In this paper, we introduce numerical value identifier processing into condition formula processing, which is the main search function of CDS, as an applied function on the presentation level of IMAH. This function is effective in business application development because subsets, including numerical values from formulas, can be taken out and be calculated by the map. We show its effectiveness through examples of a calculation system for a Bill of Materials (BOM), solving the difficulty arising from the changes of component configuration and inconsistent spelling of parts.

Keywords-component: incrementally modular abstraction hierarchy, formula expression, numerical value calculation, exponential calculation

I. INTRODUCTION

Cyberworlds are information worlds formed in a cloud either intentionally or spontaneously, with or without design. As information worlds, they are either virtual or real, and can be both. In terms of information modeling, the theoretical ground for the cyberworlds is far above the level of integrating spatial database models and temporal database models. They are more complicated and fluid than any other previous worlds in our history, and are constantly evolving. The number of organizations that conduct business in a cloud is increasing and the market is growing remarkably. On the other hand, in general business application system, as the scale of systems becomes larger and the specifications of systems changes more frequently, development and maintenance becomes more difficult, leading to higher costs and delays. In some cases, a huge system as the mainstay system in a large company, where the number of program steps is the hundreds of millions, needs several years to develop. Increases in development and maintenance cost squeeze management. Such situations arise because of combinatorial explosions. The era of cloud computing requires a more powerful mathematical background to model the cyberworlds and to prevent combinatorial explosions. In the cloud, every business object and business logic should be expressed in a unified form to eliminate discontinuity between systems or functions and to meet changes in user requirements. The needed mathematical mechanisms are: 1. disjoint union of spaces by an equivalence relation; 2. change in spaces to preserve invariants; 3. attachment of different spaces by an equivalence relation; 4.space with dimensions as a special case. We consider the Incrementally Modular Abstraction Hierarchy (IMAH) that one of the authors (T. L. Kunii) proposes able to satisfy the above requirements, as it models the architecture and the dynamic changes of cyberworlds from a general level (the homotopy level) to a specific one (the view level), preserving invariants while preventing combinatorial explosion [1]. It also benefits the reuse of information, guaranteeing modularity of information based on the mechanism of disjoint union. Unlike IMAH, other leading data models do not support the disjoint union or the attaching function by equivalence relation. In this research, one of the authors (Y. Seki) proposed a finite automaton called Formula Expression as a development tool to design spaces in IMAH. One of authors (T. Kodama) has actually designed spaces and implemented the data processing system called the Cellular Data System (CDS) using Formula Expression. In this paper, we first design an image of a numerical value identifier in the condition formula processing map, which is the main search function of CDS, on the presentation level, and then implement it (Section 4). Using this new design and processing, subsets of a numerical value can be processed. We demonstrate the effectiveness of a numerical value identifier by developing general business applications for core processing of BOMs used in manufacturing, thereby abbreviating the process of developing application programs and improving the maintainability of them.
II. IMAH AND FORMULA EXPRESSION

A. The Incrementally Modular Abstraction Hierarchy

The following list is the Incrementally Modular Abstraction Hierarchy to be used for defining the architecture of cyberworlds and their modeling:
1. The homotopy (including fiber bundles) level
2. The set theoretical level
3. The topological space level
4. The adjunction space level
5. The cellular space level
6. The presentation (including geometry) level
7. The view (also called projection) level

In modeling cyberworlds in cyberspaces, we define general properties of cyberworlds at the higher level and add more specific properties step by step while climbing down the Incrementally Modular Abstraction Hierarchy. The properties defined at the homotopy level are invariants of continuous changes of functions. The properties that do not change by continuous modifications in time and space are expressed at this level. At the set theoretical level, the elements of a cyberspace are defined, and a collection of elements constitutes a set with logical calculations. When we define a function in a cyberspace, we need domains that guarantee continuity such that the neighbors are mapped to a nearby place. Therefore, a topology is introduced into a cyberspace through the concept of neighborhood. Cyberworlds are dynamic. Sometimes cyberspaces are attached together, an exclusive union of two cyberspaces where attached areas of two cyberspaces are equivalent. It may happen that an attached space is obtained. These attached spaces can be regarded as a set of equivalent spaces called a quotient space that is another invariant. At the cellular structured level, an inductive dimension is attached spaces can be regarded as a set of equivalent spaces called a quotient space that is another invariant. At the cellular structured level, an inductive dimension is.

Formula Expression is a finite automaton as a communication tool that has been developed in order to guarantee universality in communication between subjects by expressing states of things in a formula. This is very effective in solving the frequent problems arising from misunderstandings between providers and suppliers in business application system development. Formula Expression was invented over a number of years by pursuing the greatest simplicity possible. As a result, spaces could be designed on each level of the cellular model using Formula Expression, as shown in the previous paper [2], while this was not possible using other tools. It is thought that Formula Expression, by following the levels of the cellular model, can reflect any space that humans create and their operations. In other words, it reflects human thought.

2) Definition

In modeling cyberworlds in cyberspaces, we define general properties of cyberworlds at the higher level and add more specific properties step by step while climbing down the Incrementally Modular Abstraction Hierarchy. The properties defined at the homotopy level are invariants of continuous changes of functions. The properties that do not change by continuous modifications in time and space are expressed at this level. At the set theoretical level, the elements of a cyberspace are defined, and a collection of elements constitutes a set with logical calculations. When we define a function in a cyberspace, we need domains that guarantee continuity such that the neighbors are mapped to a nearby place. Therefore, a topology is introduced into a cyberspace through the concept of neighborhood. Cyberworlds are dynamic. Sometimes cyberspaces are attached together, an exclusive union of two cyberspaces where attached areas of two cyberspaces are equivalent. It may happen that an attached space is obtained. These attached spaces can be regarded as a set of equivalent spaces called a quotient space that is another invariant. At the cellular structured level, an inductive dimension is attached together, an exclusive union of two cyberspaces where attached areas of two cyberspaces are equivalent. It may happen that an attached space is obtained. These attached spaces can be regarded as a set of equivalent spaces called a quotient space that is another invariant. At the cellular structured level, an inductive dimension is introduced into each cyberspace. At the presentation level, each space is represented in a form which may be imagined before designing cyberworlds. At the view level, the cyberworlds are projected onto view screens. An example of the homotopy level is shown in Figure 1.

![Figure 1. The fiber bundle to model and visualize business trading.](image)

B. Formula Expression

1) Outline

Formula Expression is a finite automaton as a communication tool that has been developed in order to guarantee universality in communication between subjects by expressing states of things in a formula. This is very effective in solving the frequent problems arising from misunderstandings between providers and suppliers in business application system development. Formula Expression was invented over a number of years by pursuing the greatest simplicity possible. As a result, spaces could be designed on each level of the cellular model using Formula Expression, as shown in the previous paper [2], while this was not possible using other tools. It is thought that Formula Expression, by following the levels of the cellular model, can reflect any space that humans create and their operations. In other words, it reflects human thought.

2) Definition

Formula Expression in the alphabet is the result of finite times application of the following (1)-(7).

1) \( a (\in \Sigma \star) \) is Formula Expression
2) unit element \( e \) is Formula Expression
3) zero element \( \phi \) is Formula Expression
4) when \( r \) and \( s \) are Formula Expression, addition of \( r+s \) is also Formula Expression
5) when \( r \) and \( s \) are Formula Expression, multiplication of \( r \times s \) is also Formula Expression
6) when \( r \) is Formula Expression, \( \{r\} \) is also Formula Expression
7) when \( r \) is Formula Expression, \{r\} is also Formula Expression

Strength of combination is the order of (4) and (5). If there is no confusion, \( \times, (, ) \) can be abbreviated. Unit element \( e \) is expressed as “()” and zero element \( \phi \) as “{}” in Formula Expression, but the characters \( e \) and \( \phi \) are used in this paper to prevent misunderstandings.

3) The generating grammar

The grammar which generates Formula Expression is the following.

We assume that \( \Sigma \) is a set of ideograms and its element is \( w (\in \Sigma \star) \).

\[ G = (\Sigma, \Delta, \Sigma \star, +, (, ) , \{, \}, P, E) \]
\[ P = \{E \rightarrow T|E+T, T \rightarrow F|T \times F, F \rightarrow (E)|\{E\}|\{, \}|id, id \rightarrow w\} \]

Here, \( E \) is called a formula, \( T \) is called a term, \( F \) is called a factor, \( id \) is called an identifier, \( + \) is called a separator, which creates a disjoint union (= addition operation) and is expressed as \( \Sigma \) specifically, and \( \times \) is called a connector which creates a direct product (= multiplication operation) and is also expressed as \( \Sigma \). The Parentheses (\()\) mean a set where an order of elements is not preserved and braces \( \{\} \) an ordered set where the order of elements is preserved. In short, you can say "a formula consists of an addition of terms, a term consists of a multiplication of factors, and if \( \} \) or \{ is added to a formula, it becomes recursively the factor".

4) The Meaning of Formula Expression
The language L(r) (r ∈ Σ*) that Formula Expression r expresses is defined as:

1. L(α) = {α} (α ∈ Σ*)
2. L(ε) = {ε}
3. L(φ) = φ
4. L(r + s) = L(r) ∩ L(s)
5. L(r × s) = L(r) ∩ L(s)

5) The algebraic structure of Formula Expression r, s, t, u follow the following algebraic structure.

(1) r + (s + t) = (r + s) + t, r × (s × t) = (r × s) × t
(2) r + s = s + r
(3) r × q = q × r = r
(4) r ×/g307 = /g307 × r = /g307, r+/g307 = r
(5) r ×/g304 = /g304 × r = r
(6) r ×/g314 s = /g314 s × r = r ×/g314 s
(7) s ×/g314 s = /g314 s × s = s ×/g314 s
(8) s ×/g314 s = /g314 s × s = s ×/g314 s
(9) /g520* (r × s) = /g520* (s × r)
(10) r ×/g314 s ×/g314 s = /g314 s ×/g314 s ×/g314 s
(11) /g520* (r × s × t) = /g520* (r × t × s)
(12) /g520* (r × s × t × u) = /g520* (r × t × s × u)
(13) /g520* (r × s × t × u × v) = /g520* (r × t × s × u × v)

C. The Conditional Formula Processing Map

In Formula Expression, several basic maps were defined in the previous paper [9]. A function for specifying conditions defining a condition formula utilizing basic maps is supported in CDS. This is one of the main functions, and the map is called a condition formula processing map. A formula created from these is called a condition formula. "!'" is a special factor which means negation. Recursivity by () in Formula Expression is supported, so the recursive search condition of a user is expressed by a condition formula. The condition formula processing map f is a map that gets a disjoint union of terms that satisfies a condition formula from a formula. When condition formula processing is considered, the concept of a remainder of spaces is inevitable. A remainder acquisition map g is a map that has a term that doesn’t include a specified factor.

If you assume x to be a formula and p, !p, p+q, p*q, !p+q, !p*q, (p+q), (x, p*q), (x, !p+q), (x, !p*q) to be condition formulas, the images of (x, p+q), (x, p*q), (x, !p+q), (x, !p*q) by f, g are the following:

f(x, p) = g(x, p)
f(x, !p) = g(x, p)
f(x, p+q) = f(x, p) + f(g(x, p), q)
f(x, p*q) = f(f(x, p), q)
f(x, !p+q) = g(g(x, p), q)
f(x, !p*q) = g(f(f(x, p), q))

Fig. 2 shows each image by the condition formula processing map f.

D. The Properties of Numerical Value Calculation and Exponential Calculation

If we assume that p, q, r are arbitrary numerical factors, and that s, t, u are arbitrary factors, the numerical value calculation in Formula Expression has the following properties:

1. s × /g314 s = 1
2. s × 0 = 0
3. ps = s*p

4. s ×/g307 p + s × q = s × (p + q)

5) The numerical value and exponential calculation map f is defined based on the above mentioned properties. If you assume the entire set of formulas, including the numerical factors, to be A, f: A → A and f is the followings:

f: s → s ×/g314 l
f: s ×/g314 p → s ×/g314 p
f: s ×/g314 p ×/g314 q → s ×/g314 (p ×/g314 q)

And if we assume that T is an arbitrary term, and that E is an arbitrary formula, f is:

f(T ×/g314 T) = f(T) ×/g314 f(T)
f(T +/g314 T) = f(T) +/g314 f(T)
f(E ×/g314 E) = f(E) ×/g314 f(E)

An example of the map f is shown below.

f(cat + dog + rabit + dog + cat + rabit + dog + rabit + mouse) = cat × 2 + dog × 2 + rabit × 3 + mouse × 1
As a further example, when three boxes of cigarettes (which are $5 per box) are bought, the formula for calculation and processing by the map \( f \) is:

\[
f(\text{cigarette} \times 5 \times \{1 (\text{box}) \times 3 \times \text{box})
= \text{cigarette} \times 5 \times 1 \times \text{box} \times 3 \times \text{box}
= \text{cigarette} \times 15 \times \$
\]

III. THE INTRODUCTION OF NUMERICAL VALUE IDENTIFIER INTO THE CONDITIONAL FORMULA PROCESSING

A. The design of a conditional formula processing map with numerical value(s)

New definitions are introduced into the conditional formula processing map \( f \) (II.C) to consider numerical value(s) on the presentation level of the cellular model. The map consists mainly of two maps: 1. A quotient acquisition map \( h_1 \) that gets a term that includes a specified factor. If we assume that \( p, q, r \) \((p<q<r)\) are arbitrary numerical identifiers and that \( s \sim x \) are arbitrary factors, they follow these rules:

\[
h_1: s, q \rightarrow \varphi \text{ (when } r \text{ doesn’t include a numerical identifier)}
\]

\[
h_1: s \times p \times t, q \rightarrow s \times p \times t
\]

\[
h_1: s \times r \times t, q \rightarrow s \times q \times t
\]

\[
h_1: s \times (t + u) \times p \times v \times x, q \rightarrow s \times u \times p \times v \times x
\]

\[
h_1: s \times (t + u) \times r \times v \times w \times x, q \rightarrow s \times u \times q \times v \times w \times x
\]

\[
h_1: s \times \{t + u \times p \times v \times w \} \times x, q \rightarrow s \times \{q + u \times p \times v \} \times x
\]

\[
h_1: s \times \{t + u \times r \times v \times w \} \times x, q \rightarrow s \times \{q + u \times r \times v \times w \} \times x
\]

\[
h_2: s, q \rightarrow s \text{ (when } r \text{ doesn’t include a numerical identifier)}
\]

\[
h_2: s \times p \times t, q \rightarrow \varphi
\]

\[
h_2: s \times r \times t, q \rightarrow s \times (r-q) \times t
\]

\[
h_2: s \times (t + u) \times p \times v \times w \times x, q \rightarrow s \times (t + w) \times x
\]

\[
h_2: s \times (t + u) \times r \times v \times w \times x, q \rightarrow s \times (t + r \times w) \times x
\]

\[
h_2: s \times (t + u) \times p \times v \times w \times x, q \rightarrow s \times (t + q \times w) \times x
\]

\[
h_2: s \times (t + u) \times r \times v \times w \times x, q \rightarrow s \times (t + r \times q \times v \times w) \times x
\]

A simple example, which shows the rest after some coins have been removed from a purse, using the map \( f \) is shown below.

\[
f(\text{Purse}(1 \text{- dollar} \times 3 \text{ piece} + 25 \text{- cent} \times 5 \text{ piece} + 50 \text{- cent} \times 4 \text{ piece}), !(1 \text{- dollar} \times 2 \text{ piece} + 25 \text{- cent} \times 2 \text{ piece}))
= \text{Purse}(1 \text{- dollar} \times 1 \text{ piece} + 25 \text{- cent} \times 3 \text{ piece} + 50 \text{- cent} \times 4 \text{ piece})
\]

B. Implementation

This system is a web application developed using JSP and Tomcat 5.0 as a Web server. The client and the server are the same machine. (OS: Windows XP; CPU: Intel Pentium 3, 1.2GHz; RAM: 1.1Gbyte; HD: 20GB) The following is the coding for the calculation of numerical value and exponential calculation. The focus is the recursive process (line 7, in bold) that is done if a coming numerical calculation is of the type (). The explanation is abbreviated due to space limitations.

1. term = null; factor = null;
2. while(factor is not null){
3. term = getTerm(factor);
4. while(term is not null){
5. factor = getFactor(term)
6. if(factor is of the type ()){
7. factor = calculate(the contents);
8. }
9. factor = getNumericalFactor(factor);
10. LetteFactor = getLetteFactor(factor);
11. newNF = newNF \times NumericalFactor;
12. newLF = newLF \times LetteFactor;
13. newTerm = newNF + newLF;
14. newFormula = newFormula + newTerm;
15. return newFormula;
}

IV. DEVELOPMENT OF A CORE CALCULATION SYSTEM FOR A BOM (BILL OF MATERIALS) IN MANUFACTURING

A. Outline

We take up an example of a core calculation system for a BOM (Bill of Materials). A BOM is a list of the parts or components that are required to build a product in manufacturing. When the component configuration of a product changes frequently, such as division of a part, integration of parts, or inconsistent spelling of the same components (i.e. synonyms), it is difficult to manage configuration information consistently. This is because the application programs of a system have to be developed every time the component configuration changes or inconsistent spelling of a component emerges. To solve the difficulties, we have applied CDS to the development of a core calculation system for BOMs. Firstly we make a general design of a topological space for one part of a product and the changes using CDS. Secondly, formulas for the parts in each phase are created. Thirdly, the formulas are processed by the maps of CDS to meet user requirements. Here, data is also simplified without losing generality.

B. The design of a formula for a topological space for parts and Data input according to the design

We design a formula for topological spaces for one part of a product \( P_{n,m} \) in \( m \) th layer \( n \). \( P_{n,m} \) is defined as the multiplication of a part id \((id_{a,n})\), the disjoint union of topological spaces for the subordinate parts \((\Sigma P_{i,m-1})\), the quantity of part \( num_{n,m} \) and the price of part \( price_{a,n} \) where a numerical value identifier is used. The formula for \( P_{n,m} \) is designed as follows:

\[
P_{n,m} = a \text{ part id}_{a,n} \times (\Sigma P_{i,m-1}) \times num_{n,m}
\]

where \( P_{1,m} = a \text{ part id}_{a,1} \times price_{a,1} \) and \( a \text{ part id} \) a factor which identifies one part of a product

\( num \) a factor which shows the quantity of a part

\( price \) a factor which shows the price of a part
Next, we design a part price calculation map $j$. Map $j$ is
defined using the mask map $k$ ([14]) and the calculation
map $f$ (II.D) to calculate the price $p_{n,m}$ of $P_{n,m}$.

\[
\begin{align*}
j(P_{n,m}) &= j(k(P_{n,m}, [1...0]+S)) \\
&= (\Sigma(\Sigma(price_{n,i} \times num_{n,m}) \times \text{num}_{n,m}) \times \text{num}_{n,m}) \\
&\times num_{n,m} + \text{num}_{n,m}
\end{align*}
\]

Next, we design a factor of topological spaces for the
synonyms of parts. If a part id has the synonyms of $id1, id2, \ldots$,
the factor of part id is expressed simply below.

\[
(id1+id2+ ... )
\]

We take up a simple example of $P$, including synonyms for
a part (=Ps). In the above formula in phase 1, 2 (Fig.3, 4),
we assume that A part has synonyms $A1$ and $A2$.

\[
Ps_{1,2} = (A1+A2)(B \times 3 \times S5+C \times 2 \times S4+D \times 5 \times S6) \times 1
\]

The price of $P$ is calculated using map $j$:

\[
j(P_{1,2}) = (3 \times S \times 5+2 \times 4+5 \times 6) \times 1 = 53 \times S
\]

We take up a simple example of $P'$, in which the name of
the part C in Fig.3 changes to E, whose price is $5 each.

\[
P_{1,2}' = (A1+A2+A3)(B \times 3 \times S5+C \times 2 \times S4+D \times 5 \times S6) \times 1 + (A1+A2+A3) \times C \times 2 \times S4 \times 1+(A1+A2+A3) \times E \times 2 \times 1 \times S5
\]

C. Data output

If a user wants to answer the question “What is the structure
of parts in phase 2?”, you calculate $Ps'_{1,1}$ using the
calculation map $g$ (II.D) and get the image of it. The result
is below.

\[
g(Ps'_{1,1}) = (A1+A2)(B \times 3 \times S5+E \times 2 \times S5+D \times 5 \times S6) \times 1
\]

From the result of the calculation, you know the structure
of parts in phase 2, as seen in Fig. 7.

Next, if a user wants to answer the question “How
much is the unit price of part A1 in phase 2?”, you use the
map $f$ (III.A) to get the image of $Ps'_{1,1}$, and then calculate
it using the part price calculation map $j$.

\[
\begin{align*}
j(f(Ps'_{1,1}, 'A1')) &= j('A1(B \times 3 \times S5+E \times 2 \times S5+D \times 5 \times S6) \times 1') \\
&= 175
\end{align*}
\]
From the result, you know the unit price of A1 is $55.

Next, if a user wants to answer the question “How much is the total price of parts except part E in phase 2?”, you use the map \( f \) to get the image of \( Ps'_{1,1} \) and then calculate it using the map \( j \).

\[
f(f(Ps'_{1,1}, !E')) = j((A1+A2)(B \times 3 \times $5 + D \times 5 \times $6) \times 1') = 45 \times $\]

From the result, you can know the total price of parts except part E in phase 2 is $45.

D. Considerations

In this example, the changes in component configuration are generally designed as a disjoint union of topological spaces using numerical value factors in addition to the design of a topological space for parts using CDS. Therefore, information about the parts in a phase is expressed as the union of information about changes in the parts from an initial value of parts, such that \( P' \) is accumulated information about changes from \( P \). Moreover, synonyms for parts are generally designed as a factor of the sum of words that are synonyms in the formula. The information can then be processed consistently by the maps of CDS, such as the condition formula processing map or the calculation map, according to user requirements. This modeling can reduce the amount of development of application programs and improve maintainability of the system.

V. RELATED WORKS

The distinctive features of our research are the application of the concept of topological process, which deals with a subset as an element, and that the cellular space extends the topological space, as seen in Section 2. Relational OWL as a method of data and schema representation is useful when representing the schema and data of a database [2][5], but it is limited to representation of an object that has attributes. Our method can represent both objects: one that has attributes as a cellular space and one that does not have them as a set or a topological space.

Many works applying other models to XML schema have been done. The motives of most of them are similar to ours. The approach in [8] aims at minimizing document revalidation in an XML schema evolution, based in part on the graph theory. The X-Entity model [9] is an extension of the Entity Relationship (ER) model and converts XML schema to a schema of the ER model. In the approach of [6], the conceptual and logical levels are represented using a standard UML class and the XML represents the physical level. XUML [10] is a conceptual model for XML schema, based on the UML2 standard. This application research concerning XML schema is needed because there are differences in the expression capability of the data model between XML and other models. On the other hand, objects and their relations in XML schema and the above models can be expressed consistently by CDS, which is based on the cellular model. That is because the tree structure, on which the XML model is based, and the graph structure [3][4][7], on which the UML and ER models are based, are special cases of a topological structure mathematically. Entity in the models can be expressed as the formula for a cellular space in CDS. Moreover, the relation between subsets cannot in general be expressed by XML.

Although CDS and the existing deductive database apparently look alike, the two are completely different. The deductive database [11] raises the expression capability of the relational database (RDB) by defining some rules. On the other hand, CDS is a new tool for data management, and has nothing to do with the RDB.

VI. CONCLUSIONS

We have developed a data processing system called the Cellular Data System (CDS) based on the Incrementally Modular Abstraction Hierarchy (IMAH). In this paper, we introduced the numerical value processing map into the search function as an applied function on the presentation level of CDS. Using this function in business application development, you can deal with a subset of a numerical value of a formula, so that you can output the data which a user requires more flexibly. Consequently, system development becomes simpler, business processes more visible, and business applications more flexible to changes in business conditions. As a result, use of CDS can make developers more creative, while preventing combinatorial explosions and reducing the need for intensive testing. This function has already been put to practical use in many companies in Japan, and further strides are being made every day. We are sure that CDS has the potential to bring great social impact in the era of cloud computing.

REFERENCES


