A New Method for Developing Business Applications: The Cellular Data System

Toshio Kodama
Maeda Corporation
2-10-26 Fujimi, Chiyoda-Ku, Tokyo
102-0071 Japan
kodama.ts@jcity.maeda.co.jp

Tosiyasu L. Kunii
IT Institute, Kanazawa Institute of Technology
1-15-13 Jingumae, Shibuya-ku, Tokyo
150-0001 Japan
tosi@kunii.com, kunii@iti.kanazawa-it.ac.jp

Yoichi Seki
Software Consultant
3-8-2 Sakaemachi, Hino City, Tokyo
191-0001 Japan
yseki@bbl@hinocatv.ne.jp

Abstract

Currently, in the development of business applications, difficulties occasionally arise between the customer side and the supplier side, causing delays and higher development costs. To deal with this, we propose a new development method for business applications to validate them by invariant preservation based on equivalence relations. We have realized it by designing an algebraic expression called Formula Expression and applying the algebra to the design of each space in the cellular model. The system is called a Cellular Data System (CDS). It can become a common tool for both sides to make agreements efficiently in the future. Moreover, because you can manage data which has any form by using CDS, it is appropriate when dealing with data not only in business application development but in the Cyberworld where no data manager exists.

Keywords: Formula Expression, Cellular model, Topological space, Cellular space

1. Introduction

As Web technology develops, the era has come when immense amounts of information are constantly being processed, and various computers and devices around the world are connected around the clock on the Web. Consequently, a flexible development method and high maintainability, suitable for a situation where business requirements are ever changing, are needed in the development of a business application system. But under the current system, the customer side is concerned only with requirements analysis, while the specialized techniques of architectural design, detailed design, coding, module testing and system testing are generally left to the supplier side. (Fig.1) In that
situation, modification work is often required because of ambiguous requirements analysis, changes to specifications during development, differences in product quality, et al., leading, as a result, to delays in development time and increases in development costs. The same is true for maintenance of a developed system. From the viewpoint of a data model, no popular data models, such as the relational model, the object oriented model, the entity relationship model, XML, etc. are based on the concept of a disjoint union of information that provides information modularity. In addition, they don’t support the mechanism of attachment by equivalence relation to guarantee and validate invariant preservation at the cellular modeling level. This inevitably results in increases in the amount of application development in cost and time and difficulty of maintenance. [1][2][3]

To solve the problem, we designed an algebraic expression named Formula Expression and its maps as a common business application development tool for both the customer side and the supplier side. In addition, we adopted the cellular model, which is a novel data model that can make up for the demerits of the above-mentioned data models, and we designed each space in the cellular model by Formula Expression. According to the design, we implemented the prototype system and confirmed the effectiveness of it by using simple samples. We call this a "Cellular Data System".

Our research is the first attempt in the field of cellular model data engineering to develop a prototype system and, though still at an early stage, we are certain that it will have a major social impact in data management.

In Chapter 2 we explain the design of Formula Expression, and in Chapter 3 we explain with simple examples the design of each space in the cellular model by using Formula Expression as a tool. In Chapter 4 we show the flowchart of the main algorithm of the prototype system, and in Chapter 5 we verify the effectiveness of the system by samples. We conclude and consider future work in Chapter 6.

2. The design of Formula Expression

We explain the design of Formula Expression, which is unpublished work created by Y. Seki, and currently undergoing revision.

2.1. The definition of Formula Expression

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

(1) a (a ∈ Σ) is Formula Expression
(2) unit element ε is Formula Expression
(3) zero element φ 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×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 (5) and (4). If there is no confusion, ×, (), { } can be abbreviated. And Formula Expression is often simply called formula.

2.2. The grammar that generates Formula Expression

The grammar that generates Formula Expression is defined as follows:

We assume that Σ is a set of ideograms and its element is w (w ∈ Σ).

G = ({E, T, F, id}, Σ → {+, ×, (, ), {, }}, P, E),
P = {E → T | E + T, T → F | T × F, F → (E) | {E} | id, E → w}

Here, E is called a formula, and T is called a term, and F is called a factor, and id is called an identifier; + is called an addition operation, × is called a multiplication operation, () is called a 1st bracket, and {} is called a 2nd bracket. If you paraphrase the above-mentioned grammar, you can say "a formula consists of an addition of terms, a term consists of a multiplication of factors, and if the 1st or 2nd bracket is added to a formula, it becomes recursively the factor". And when a term is a component of a formula, we say that the formula has the term. And when a factor of the bracket that includes a term is a component of a formula, we say that the formula includes the term. It is the same with a term and a factor. An example is shown below.

The term "a(b+c){d+e}" has factors "a", "(b+c)", "{d+e}" and includes factors "b", "c", "d", "e".

2.3. The meaning of Formula Expression

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

(1) L(a) = {a} (a ∈ Σ)
(2) L(ε) = {ε}
(3) \( L(\varphi) = \{\} \)
(4) \( L(r+s) = L(r) \cup L(s) \)
(5) \( L(rs) = \{rs\} \)

2.4. The algebraic structure of Formula Expression

Formula Expression \( r, s, t, u \) follow the following algebraic structure.

(1) \( r+(s+t) = (r+s)+t \)
(2) \( r \times (s \times t) = (r \times s) \times t \)
(3) \( r + r = r \)
(4) \( r \times \varphi = \varphi \times r = \varphi \)
(5) \( r \times (s+t) = r \times s + r \times t \)
(6) \( r \times r = r \)
(7) \( \{r+s\} \times \{t+u\} = r \times t + s \times u \)

2.5. Expansion map \( f \)

An expansion map \( f \) is a map that takes off brackets from a formula according to the following rules. If you assume the entire set of formulas to be \( A \), \( f : A \mapsto A \), and arbitrary terms \( r, s, t, u \) follow these rules:

\[
\begin{align*}
    f(r+s) = f(r) + f(s) \\
    f(r) = r & \text{ (when } r \text{ doesn't include any bracket)} \\
    f(r \times (s+t)) = f(r) \times f(s+t) \\
    f(r \times s + r \times t) = f(r) \times f(s) + f(r) \times f(t) \\
    f(r \times s \times t) = f(r) \times f(s) \times f(t) \\
    f(r \times s + t \times u + v) = f(r \times s) + f(t \times u + v) \\
    f(r \times s + t \times u) = f(r \times s) + f(t \times u) \\
\end{align*}
\]

Operation by the expansion map \( f \) is called expansion of a formula. A synthetic map that repeats the expansion map \( f \) until the formula doesn't include the brackets is called an whole expansion map \( f^n \).

2.6. Bind map \( g \)

A bind map \( g \) is a map that changes a formula to a disjoint union of a term that has a specified factor and a term that doesn't. If you assume the entire set of formulas to be \( A \), the entire set of factors to be \( C \), \( g : B \times C \mapsto A \). Arbitrary terms \( r, s, t, u, v \) and an arbitrary factor \( p, q \) follow these rules:

\[
\begin{align*}
g(r+s, p) = g(r, p) + g(s, p) \\
g(r, p) = r & \text{ (when } r \text{ doesn't have } p) \\
g(r \times t \times s \times p \times u \times v, p) = \{r+t\} \times \{s+u\} \times \{v+w\} \\
g([r+t] \times [s+u] \times [v+w], p) = [r+t+v] \times [s+u+w] \\
\end{align*}
\]

2.7. Division map \( h \)

A division map \( h \) is a map that divides a term into a disjoint union of a term that has a specified factor and a term that doesn't. If you assume the entire set of formulas to be \( A \), the entire set of terms to be \( B \) and the entire set of factors to be \( C \), \( h : B \times C \mapsto A \). Arbitrary terms \( r, s, t, u, v \) and an arbitrary factor \( p, q \) follow these rules:

\[
\begin{align*}
h(r, p) = r & \text{ (when } r \text{ doesn't include } p) \\
h(r \times s, p) = r \times s \\
h(r \times (s+t \times p \times u + v) \times w, p) = r \times (s+t \times p \times u \times v) \times w \\
h([r+t] \times [s+u] \times [v+w], p) = [r+t] \times [s+u+w] \times [v+w] \\
\end{align*}
\]

Operation by a division map \( h \) is called division of a term by the factor \( p \), and the term that has \( p \) is called a quotient term; the term that doesn't have \( p \) is called a remainder term. An example of a division map \( h \) is shown below.

\[
g(a \times p \times b + e \times f + c \times p \times d, p) = \{a+c\} \times p \times \{b+d\} + e \times f
\]

2.8. Attachment map \( i \)

An attachment map \( i \) is a map that creates a term from two terms by corresponding the factor of one term to the factor of another term. If you assume the entire set of formulas to be \( A \), the entire set of terms to be \( B \) and the entire set of factors to be \( C \), \( i : B \times C \times C \mapsto A \). Arbitrary terms \( r, s \) and arbitrary factors \( p, q \) follow these rules:

\[
i(r, s, p, q) = r \times s & \text{ (when } r \text{ doesn't have } p, \text{ or when } s \text{ doesn't have } q) \\
i(r \times p \times s, t \times q \times u, p, q) = \{r+t\} \times \{p+q\} \times \{s+u\}
\]

Operation by an attachment map \( i \) is called attachment of terms by the factors \( p, q \) and the term that has \( \{p+q\} \) after attachment is called an attachment term.

3. The design of the cellular model

The cellular model is a novel information model that T. L. Kunii created. [4] More actual phenomena can be modeled with the cellular model. In this chapter, we design the spaces on each level (3.1. a set level, 3.2. a topology level, 3.3. an adjunction space level and 3.4. a cellular space level) in the cellular model by using Formula Expression. The advantages of the use of Formula Expression are: 1. spaces on each level can be designed, and 2. a disjoint union of spaces can be expressed by the addition. This designed system is called a "Cellular Data System (CDS)".

3.1. A set level

As a set is a space that consists of its elements, the formula which expresses a set is as follows. Union is simply represented by the sign \( \Sigma \).

\[
id \times (\Sigma \text{ an element}), \text{ or } (\Sigma \text{ an element}) \times id
\]

"id" is the factor that identifies a set, and an element is always a term. The example of the set that represents "a family" is shown below.

If there are a father, a mother, a son and daughter in a family, it is expressed as the formula for the set below.

\[
fam(father + mother + son + daughter)
\]

Fig.3.1 Formula for the set "a family"

3.2. A topology level

A topological space extends the set and is a space that can express an element as a subset. Hence, the formula for a subset is as follows:

\[
\Pi(\Sigma \text{ an element}) \text{ or } \{\Sigma \text{ an element}\}
\]

And the formula for a topological space is:

\[
id \times (\Sigma \text{ a subset})
\]

If there is no confusion, the bracket ( ) can be abbreviated. An example is shown below.

If there is a topological space of a family, and the parents have a son, the formula for the topological space is as follows:

\[
fam\{father+mother\}son
\]

Fig.3.2-1 Formula for the topological space "parents with a son"

Next, if the parents have a daughter, the formula for the topological space about the daughter is added to the former formula. The formula for the family is as follows:

\[
fam\{father+mother\}son + fam\{father+mother\}daughter
\]

Fig.3.2-2 Formula for the topological space "parents with a son and parents with a daughter"

If we bind the formula by the factor \{father+mother\}, the formula changes to the following:

\[
fam\{father+mother\}(son+daughter)
\]

Fig.3.2-3 Formula for the topological space "parents with children"
The topological space differs from the set, and it can express the relation between the subset of parents and the subset of children like above-mentioned examples.

### 3.3. An adjunction space level

A quotient space is created by the division of a topological space according to equivalence relations. An adjunction space is created by attaching according to equivalence relations between quotient spaces. Hence, the formula for a topological space changes to the formula for a quotient space via the division map by the factor that expresses equivalence relations, and the quotient terms that have each factor expressing equivalence relations change to the formula for an adjunction space by attaching via the attachment map.

Let X and Y be the formulas of topological spaces. If a factor x that is included in X and a factor y that is included in Y are equivalent, namely \( x \sim y \), the formulas of quotient spaces are created via division map by the factors x and y. If you assume each quotient term of X, Y to be \( t^{x\times s}, t^{y\times u} \), the formula for quotient spaces is as follows:

\[
\text{a remainder term of } X + \text{a remainder term of } Y
\]

Next, the formula for the adjunction space is created by attaching quotient terms of X and Y via attachment map by x and y.

\[
\text{a remainder term of } X + \{t+s\} \times \{x+y\} \times \{t+u\}
\]

An example is shown. You assume that there is the formula for topological spaces of a family X and a family Y.

\[
\text{familyX}\{\text{fatherX}+\text{motherX}\}\{\text{sonX}+\text{daughterX}\}
\]

\[
\text{familyY}\{\text{fatherY}+\text{motherY}\}\{\text{sonY}+\text{daughterY}\}
\]

If an equivalence relation "marriage" happens between the daughter of the family X and the son of the family Y, the formula for the quotient spaces is created by division map.

\[
\text{familyX}\{\text{fatherX}+\text{motherX}\}\{\text{sonX}\}
\]

\[
+\text{familyX}\{\text{fatherX}+\text{motherX}\}\{\text{daughterX}\}
\]

\[
+\text{familyY}\{\text{fatherY}+\text{motherY}\}\{\text{sonY}\}
\]

\[
+\text{familyY}\{\text{fatherY}+\text{motherY}\}\{\text{daughterY}\}
\]

Next, the formula for the adjunction space is created by attachment map.

\[
\text{familyX}\{\text{fatherX}+\text{motherX}\}\{\text{sonX}\}
\]

\[
+\text{familyX}\{\text{fatherX}+\text{motherX}\}\{\text{daughterX}\}
\]

\[
+\text{familyY}\{\text{fatherY}+\text{motherY}\}\{\text{sonY}\}
\]

\[
+\text{familyY}\{\text{fatherY}+\text{motherY}\}\{\text{daughterY}\}
\]

### 3.4. A cellular space level

A cellular space extends the adjunction space, and is a space that can express dimensions. In this case, the dimensions correspond with the attributes of a cellular space. Hence, the formula for a cellular space is as follows:

\[
id\times(\text{a primary key attribute}\times\{e+\Sigma\text{a subset that expresses the other attributes}\}\times(\Sigma\text{an instance}\id\times\{e+\Sigma\text{a subset that expresses values}\}))
\]
"id" is the factor that identifies a cellular space and "an instance id" is the factor that identifies an instance. An example of the cellular space of the family is shown below.

family{relationship{ε+name+birthday}(father{ε+Masao+1945/7/17}+mother{ε+Utako+1946/2/17}+daughter{ε+Kaori+1974/12/26}+son{ε+Toshio+1975/3/1})}

If there is the requirement of gaining the values of the attribute "name" as an example, the formula is expanded by whole expansion map first.

family×relationship×father
+family×relationship×name×father×Masao
+family×relationship×B.D.×father×1945/7/17
+family×relationship×mother
+family×relationship×name×mother×Utako
+family×relationship×B.D.×mother×1946/2/17
+family×relationship×daughter
+family×relationship×name×daughter×Kaori
+family×relationship×B.D.×daughter×1974/12/26
+family×relationship×son
+family×relationship×name×son×Toshio
+family×relationship×B.D.×son×1975/3/1

Next, if the formula is bound by the factor "name", the created bind term is as follows. This result can meet the requirement.

family×relationship×name×{father×Masao+mother×Utako+daughter×Kaori+son×Toshio}

4. Development of the prototype system

T. Kodama has developed the prototype system according to the design in chapters 2 and 3.

4.1. The environment

This system is a web application developed using JSP1.4 and Tomcat4.2 as a web server. The client and server are the same machine. (OS: Windows XP; CPU: Intel Pentium3 1.2 GHz; RAM: 512Mbyte; HD: 20 GB)

4.2. The algorithm

The flowchart of the algorithm of the expansion operation that is the core function of the prototype system is shown.
5. Case studies

To verify the effectiveness of the prototype system, examples of data management of information that has different forms are shown. Assume that you manage data of 1. two business cards of different types, 2. a letter and 3. memos in a memo pad. The actual sample data is shown in Fig. 5. Although the data forms of four pieces of information are totally different, as seen in Fig. 5, you can manage them by creating spaces using Formula Expression according to the design.

5.1. Input of data

The two business cards and the letter in Fig. 5 are considered as cellular spaces because the attributes of their values are specified or can easily be guessed. On the other hand, the memos in Fig. 5 are considered as a disjoint union of three topological spaces because there are three sentences in the memos and they have no forms of information. A formula for the cellular space of business card 1 is as follows:

```
BUSINESSCARD(CARD-ID{ε+NAME{FIRST+FAMILY}+COMPANY+DEP.}+ADDRESS+TEL
+FAX+E-MAIL+URL}{c1{ε+myName{Toshio+Kodama}+MaedaCorporation+InformationSystemDivision+J.CITY5-8Takamatsu...+81-3-5372-4894+
81-3-5372-4735+kodama.ts@...+http://www.maeda.co.jp}}
```

A formula for business card 2 is created in the same way, and added to the previous formula as follows:

```
...+BUSINESSCARD(CARD-ID{ε+NAME{FIRST+FAMILY}+HOME{ADDRESS+TEL+CELLPHONE{NUMBER+E-MAIL}}+HOMETOWN{ADDRESS+TEL}}{c2{ε+myName{Toshio+Kodama}+myhome{2-23-23ShimomeguroMeguro-ku...+81-3-3491-8829+myphone{090-4387-7951+toshio+Kodama}+myHometown{422HironishiWakaya
maCity...+81-734-62-2769}}}}
```

A formula for the cellular space of the letter is created and added as follows:

```
...+LETTER(LETTER-ID{ε+RECEIVER{NAME{FIRST+FAMILY}+ADDRESS+ZIP}+SENDER{NAME{FIRST+FAMILY}+ADDRESS+ZIP}}{l1{ε+son{son'sName{Toshio+Kodama}+2-23-23ShimomeguroMeguro-ku...+153-0064}+father{father'sName{Masao+Kodama}+422HironishiWakaya
maCity...+649-6339}}}}
```

A formula for the disjoint union of three topological spaces in the memos is as follows:

```
...+KodamaFamily{Father×Masao+Mother×Utako
}{Daughter×Kaori+Son×Toshio+Rescher×Tokyo+Kaori'sfamily{Kaori×JuniorHighSchoolTeacher+HerHusband}{FirstSon+SecondSon+Daughter}Hokkaido
<Advantages>
● inputted data is visible
In this system, system users can easily determine whether inputted data is correct or not after data input because inputted data is expressed as visible formulas. On the other hand, in the conventional way, other programs need to be developed to check the inputted data because it is not visible.
● no need for file design
In this system, there is no distinction between file design and data input, because the schema and its instances in a cellular space can be described by the same Formula Expression. 2.objects can be expressed as a disjoint union of spaces, and 3.the formula for
schema can express the nested form. Consequently, there is no need for the conventional file design.

- flexible data input
  In this system, if you create the formula for the set or topological space, you can input data with no attribute without designing its attribute. Namely, you can input data with no forms, such as memos, that can never be inputted in the conventional way.
- no need for input programs
  In this system, the creation of formulas equals data input. Hence, there is no need for the conventional development of input programs.

5.2 Output of data

Example1. When you output the values of the attribute "ADDRESS", you can get the values as a bind term, if you expand the inputted formula by the whole expansion map and bind the result by the factor "ADDRESS" via the bind map. The bind term is as follows:

\[
\{\text{BUSINESSCARD}\times\text{CARD-ID}\times\text{BUSINESSCARD}\times\text{CAR D-ID}\times\text{HOME}\times\text{BUSINESSCARD}\times\text{CARD-ID}\times\text{LETTER}\times\text{LETTER-ID}\times\text{RE CEIVER}\times\text{LETTER}\times\text{LETTER-ID}\times\text{SENDER}\}\times\text{ADDRESS} \{c1\times\text{J.CITY3}=8-Takamatsu\ldots+c2\times\text{myhome}\times2-23-23\text{ShimomeguroMeguro-ku\ldots}+c2\times\text{myHome town}\times422\text{HironishiWakayamaCity\ldots}+l1\times\text{son}\times2-23-23\text{ShimomeguroMeguro-ku\ldots}+l1\times\text{father}\times422\text{HironishiWakayamaCity\ldots}\}
\]

Example2. When you output the data related to "Kaori", you can get the data as an attachment term if you attach the terms of the topological spaces that include the factor "Kaori". The attachment term is as follows:

\[
\{\text{KodamaFamily}\{\text{Father}\times\text{Masao}+\text{Mother}\times\text{Utako}\}\times\text{Daughter}\times\text{Kaori'sfamily}\times\text{Kaori}\{q=\text{JuniorHighSchool}\times\text{Teacher(firstSon}\times\text{SecondSon}\times\text{Daughter)}\times\text{Hokkaido}\}
\]

\[<\text{Advantages}>
\]

- no need for output programs
In this system, if you change the shape of the inputted formula by the maps, you can output data that meets the requirements. Hence, there is no need for the development of output programs or applications that express the relevance between objects.

5.3 Maintenance

Example1. If there is a requirement to add data "recycled paper is being used" to business card 1, you only have to insert it into the formula for business card 1 as follows:

\[
\text{BUSINESSCARD}(\text{CARD-ID}\{\varepsilon+\text{NAME}+\text{FIRST+ FAMILY}\}+\text{COMPANY}+\text{DEP.}+\text{ADDRESS}+\text{TEL}+\text{FAX}+\text{E-MAIL}+\text{URL})\{c1\{\varepsilon+\text{Name}\{\text{Toshio+K odama}\}+\text{MaedaCorporation}+\text{InformationSystemDivision}\times\text{J.CITY5}=8-Takamatsu\ldots+81-3-5372-4894+81-3-5372-4735+\text{kodama.ts}@\ldots+\text{http://www.maeda.co.jp}+"\text{recycled paper is being used}"\})
\]

If you divide the above formula by the inserted data, you get the quotient term as follows:

\[
\text{BUSINESSCARD}\times\text{CARD-ID}\times c1\times"\text{recycled paper is being used}"
\]

Thus, because the instance id "c1" is multiplied to the data, you can consider the data to be being inserted correctly.

Example2. Next, if there is a requirement to add attribute "POSITION" and its value "manager" to business card 1, you only have to insert them into the formula for business card 1 as follows:

\[
\text{BUSINESSCARD}(\text{CARD-ID}\{\varepsilon+\text{NAME}+\text{FIRST+ FAMILY}\}+\text{COMPANY}+\text{DEP.}+\text{ADDRESS}+\text{TEL}+\text{FAX}+\text{E-MAIL}+\text{URL}+\text{POSITION}\{c1\{\varepsilon+\text{myName}\{\text{Toshio+Kodama}\}+\text{MaedaCorporation}+\text{InformationSystemDivision}\times\text{J.CITY5}=8-Takamatsu\ldots+81-3-5372-4894+81-3-5372-4735+\text{kodama.ts}@\ldots+\text{http://www.maeda.co.jp}+"\text{manager}"\})
\]

\[<\text{Advantages}>
\]

- no need for maintenance work when adding/changing attributes and values
In this system, you don't have to worry about the addition of attributes and values, and only have to create the formula for the data you want to input because there is no file design work. It is similar when changing them. Hence, conventional maintenance work, such as changes in the database design, the development of data convert programs, or the modification of I/O programs is unnecessary.
6. Related work

The distinctive features of our research are that application of the concept of topological process, which is dealing with a subset as an element, and that the adjunction space and the cellular space extend the topological space as you see in Chap 3.

Relational OWL as a method of data and schema representation is useful when representing schema and data of a database. [6] 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 doesn’t have as set or a topological space.

Many works applying other models to XML schema are being done. The motives of most of them are similar to ours. Approach in [11] aims at minimizing document revalidation in a XML schema evolution, based on a part of the graph theory. X-Entity model [12] is an extension of the Entity Relationship (ER) model and is converting XML schema to a schema of the ER model. In approach of [9], the conceptual and logical levels are represented using standard UML class and the XML represents the physical level. XUML [13] is a conceptual model for XML schema, based on the UML2 standard. These application researches concerning XML schema are needed because there are differences in expression capability between XML and other models. On the other hand, objects and their relation 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 XML model is based, and the graph structure, on which UML and the ER model are based, are special cases of a topological structure mathematically. And entity in the models can be expressed as the formula for a cellular space in CDS. Moreover, the relation between subsets, as we showed in 3.2, in general cannot be expressed by XML.

Although CDS is similar to the existing deductive database in expression, they are completely different. The deductive database [17] is the one that raises expression capability of the relational database (RDB) by defining some rules. On the other hand, CDS is a proposal for a new tool for data management and has nothing to do with the RDB.

Plenty of use case tools are currently available. But they support systems development according to the existing data models. The differences from CDS mainly are that we apply a novel model, the cellular model, for building CDS, and that the customer side can confirm the output by changing formulas using the defined maps after creating formulas as the input.

7. Conclusion and future work

We proposed a new method for developing business application. We call it a Cellular Data System (CDS). In this paper, first we designed an algebraic expression...
and its maps called Formula Expression. Next, we designed each space in the cellular model using Formula Expression as a tool. We then developed a prototype system according to the design, and verified the effectiveness of the system by considering simple samples of data. CDS can become a common tool for both the customer side and the supplier side in business application development and prove useful in making agreements between the two sides, preventing some difficulties that may arise from ambiguity in the agreement (Fig.7). This research is still in its infancy, but our proposal has the potential to dramatically change, not only the way of developing business applications, but the concepts of existing data management, where the existence of a data manager is inevitable.

![Development process using CDS](image_url)

**Fig.7 Development process using CDS**

Our future work is as follows.

- Developing maps that change the shapes of formulas
- There are various requirements for data output. Hence, application maps to meet them are necessary.
- Measuring processing speed
  
There is no doubt that the processing of the formulas is a heavy load on the machines. Realistic measures should be taken according to the scales of the formulas.

**References**


