A Step Toward Foundation of Class Diagram Algebra for Enterprise Service Systems

Hidekazu Enjo, Motonari Tanabu, and Junichi Iijima

***Abstract*—An enterprise service system is large and complex and manages big and complicated data. A large set of partial data models are used during designing an information system for a large enterprise because each service application consisting of an enterprise service system uses only a part of big and complicated data in most cases. The skill of modelers makes fluctuation and discrepancy among data models. It is necessary how to keep consistency among data models. There are several diagram methods to support data modeling. The Unified Modeling Language is a popular standard modeling language. A class diagram is describing a static view of data model. There are two kinds of inconsistency among data models described in class diagrams. One is inconsistency within a data model described in a class diagram. The other is inconsistency depending on operations for class diagrams. We present syntax of a class diagram describing a data model for syntactical foundation for class diagram algebra. Then we introduce syntactical merger, remain, complement and intersection operations on class diagrams. Consolidation and restoration conditions keep consistency of class diagrams syntactically during results of those operations.**

***Index Terms*—class diagram algebra, data modeling, Unified Modeling Language**

I. Introduction

An enterprise service system is large and complex and manages big and complicated data. It is very hard to handle big and complicated data. For example, it is difficult to describe a big and complicated data model of an enterprise service system into one diagram. A large set of partial data models are used during designing an information system for a large enterprise because each service application consisting of an enterprise service system uses only a part of big and complicated data in most cases. This situation raises a risk of embedding inconsistency caused by modelers' mistakes because the skill of modelers makes fluctuation and discrepancy among data models. And inconsistency within the set of partial data models decreases the quality of the data model. Without dependency of modelers' skill, formal operations like merger and remain are preventing inconsistency for manipulating partial data models. However, it is unclear which operations for partial data models have well properties keeping consistency among data models.

There are several diagram methods to support data modeling. The Unified Modeling Language (UML) [1] is a popular standard modeling language, especially for object oriented design. A class diagram, which is a type of UML diagrams, is describing a static view of data model. It is necessary to clear how to keep consistency among class diagrams.

There are two kinds of inconsistency among data models described in class diagrams. One is inconsistency within a data model described in a class diagram like differences between attributes, data types, or multiplicities of same name classes [2], [3], [4]. The other is inconsistency depending on operations like merger and remain for class diagrams. Because a class diagram as result of an operation isn't always consistent even if there is consistency in each data model as augments of an operation. In this paper, we focus the latter situation and propose consistent operations toward class diagram algebra.

II. Merger and Remain of Class Diagrams

It is easy to merge two parts of class diagram like class diagram X – Fig. 1 – and class diagram Y – Fig. 2 – and the result is class diagram Universe – Fig. 3 – because same classes are merged into one class in a result diagram and same associations are merged into one association of that.

A remain operation is one of the basic operations for dividing and analogized as difference of set. But remain operation is a little different from difference of set. If the class diagram Y is remained of class diagram Universe and class diagram X, class diagram X includes class Order and class OrderDetails but the remain result as class diagram Y also includes Order and OrderDetails. Because association PO in class diagram Y is relating those classes.

However, there are several ways to divide a large class diagram like Universe into parts of class diagram like X and Y or X and Z – Fig 4 – because the result of merging two parts of class diagram as class diagram X and class diagram Z is as same as class diagram Universe.

If class diagram Universe is the whole class diagram, class diagram Y is complement of class diagram X. Because class diagram Y is remained of class diagram Universe and class diagram X. Class diagram Y is complement of class diagram X and class diagram X is complement of class diagram Y. It means that class diagram X is double complement of class diagram X. However, class diagram Y is also a complement of

Manuscript received February 2 2009.

Hidekazu Enjo is with NTT DATA CORPORTATION, 3-3-9 Koto-ku, Tokyo 135-8671, Japan (phone: +81-50-5546-2297; fax: +81-30-3532-0488; e-mail: enjouh@nttdata.co.jp).

Motonari Tanabu is with Yokohama National University, 79-4 Tokiwadai, Hodogaya, Yokohama 240-8501, Japan. (e-mail: tanabu@ynu.ac.jp).

Junichi Iijima is with Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8550, Japan (e-mail: iijima.j.aa@m.titech.ac.jp).

class diagram Z and class diagram X is a double complement of class diagram Z. It means that class diagram Z isn't satisfied double complement law. We need to know the condition for satisfying double complement law because this property is important for algebraic foundation.

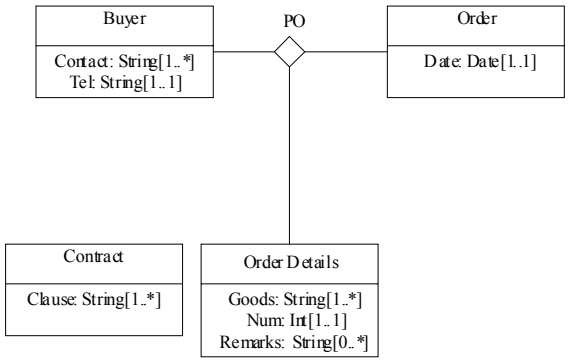


Fig. 1. Example of Class Diagram X

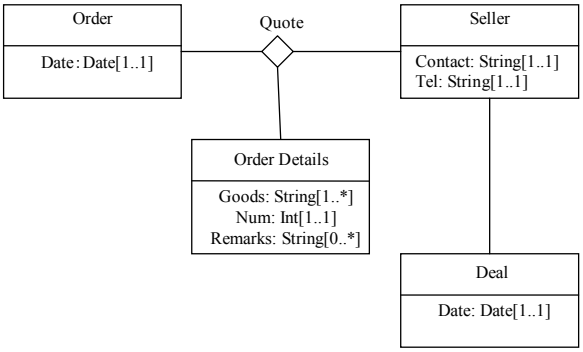


Fig. 2. Example of Class Diagram Y

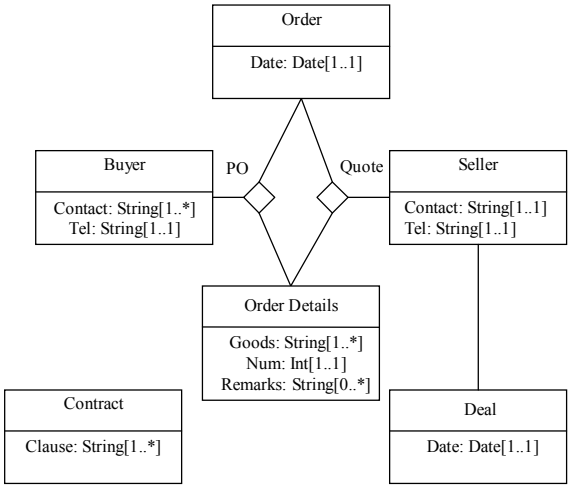


Fig. 3. Example of Class Diagram Universe

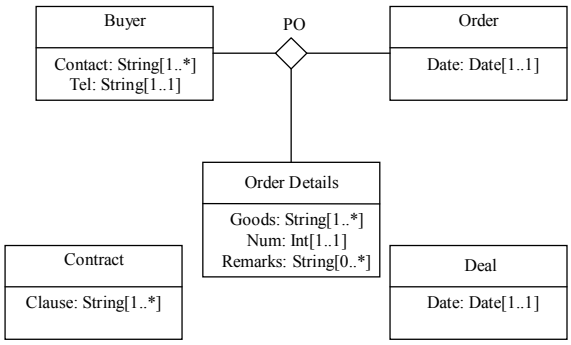


Fig. 4. Example of Class Diagram Z

III. RELATED WORK

There are several studies of mathematical foundation of class diagrams. Following are studies related consistency analyses for one class diagram. Berardi, Calvanese, and Giacomo [2] presented the correspondence between class diagrams and Description Logics, which enable us to utilize Description Logics based systems for reasoning on class diagrams. Szlenk [3] presented the mathematical definition of class diagrams and studied the consistency within a single class diagram. Kaneiwa and Satoh [4] proposed optimized algorithms that compute respective consistencies for class diagrams based on first order predicate logic. They discussed mathematical foundation of class diagram but didn't mention about algebraic structure.

Following studies are related algebra for class. Calvanese, Lenzerini, and Nardi [5] surveyed four type of knowledge representations -- description logic, frame based system, semantic data model, object-oriented data model and propose unifying framework. Bueher[6] formalized object-oriented data model based on Boolean algebra and query based on set theory. They discussed classes but didn't mention about class diagrams.

Sabetzadeh and Easterbrook [7] studied the merging of class diagrams to gain a unified perspective. However, they focused how to merge inconsistent and incomplete class diagrams for requirement engineering and they hadn't been clear detail of algebraic structure for a merging operation.

IV. NOTATION

A set $\boldsymbol{Name}$ is a set of name of all elements consist of class diagram including a class name, an association name, an attribute name and a role name. A set $\boldsymbol{DataType}$ is a set of data type, where type of all data including integer and character represented by $\boldsymbol{\{Int, Bool, Char, String, Date, ...\}}$. A set $\boldsymbol{Multiplicity}$ is a set of multiplicity $\boldsymbol{(m,n)}$ that is a pair of lower bound $\boldsymbol{m}$ and upper bound $\boldsymbol{n}$, where $\boldsymbol{0 \le m \le n, m \in N_0, n \in N_1}$. A set $\boldsymbol{N_0}$ is a set of integer grater than or equal to 0. A set $\boldsymbol{N_1}$ is a set of integer grater than or equal to 1.

V. SYNTAX OF A CLASS DIAGRAM

A class diagram is a type of UML and employed to model concepts in static views for an information structure of enterprise system consisting of classes and their interrelationships. A class diagram has been represented by graphical and abstract syntax. In this section, we focus abstract syntax for simplifying discussions. We define the abstract syntax of the class diagram as followed:

A. Syntactical definition of a class

A class c is a pair $(Name(c), Attrs(c))$, where $Name(c) \in Name$ is a name of the class c and $Attrs(c)$ is a list of attributes in the class c. An attribute list $Attrs(c)$ is a finite list $(Attr(c,1),...,Attr(c,n))$ of finite number n. An attribute $Attrs(c,i)$ is a triple of $(Name(c,i), Type(c,i), Multi(c,i))$, where $Name(c,i) \in Name$ is a name of i th attribute and a member of a set $Name$, $Type(c,i) \in DataType$ is a type of i th attribute and a member of a set $DataType$, $Multi(c,i) \in Multiplicity$ is a multiplicity of i th attribute and a member of a set $Multiplicity$. A set $Class$ is a set of all classes.

B. Syntactical definition of an association

An association a is a pair $(Name(a), Assocs(a))$, where $Name(a) \in Name$ is a name of the association a and $Assocs(a)$ is a list of associated classes in the association a. An list $Assocs(a)$ of associated classes is a finite list $(Assoc(a,1),...,Assoc(a,n))$ of finite number n. An associated class $Assoc(a,i)$ is a triple of $(Role(a,i), Multi(a,i), AC(a,i))$, where $Role(a,i) \in Name$ is a name of i th association, $Multi(a,i) \in Multiplicity$ is a multiplicity of i th association, $AC(a,i) \in Class$ is a related class of i th association. A length of the list $Assocs(a)$ must be at least 2. A set $Association$ is a set of all associations.

C. Syntactical definition of a class diagram

A class diagram is a pair (C, A), where $C \subseteq Class$ is a subset of $Class$ and $A \subseteq Association$ is a subset of $Association$. A class diagram (C, A) suffers following conditions.

Condition 1: if names of two classes are same, two classes are same.

$$\forall c_i \in C, \forall c_j \in C, Name(c_i) = Name(c_j) \Rightarrow c_i = c_j \quad (1)$$

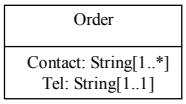
Condition 2: if names of two associations are same, two associations are same.

$$\forall a_i \in A, \forall a_j \in A, Name(a_i) = Name(a_j) \Rightarrow a_i = a_j \quad (2)$$

Condition 3: all associated classes of an association are included in the set of classes in the class diagram.

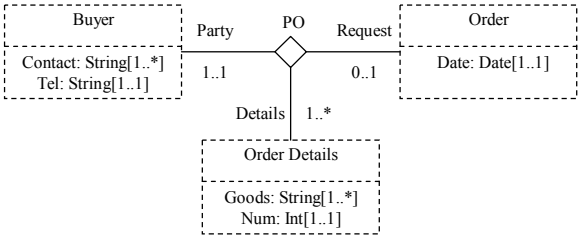
$$\forall a \in A, (r,m,ac) \in Assocs(a) \Rightarrow ac \in C \quad (3)$$

In order to reduce the complexity, we consider eliminating some components as operation and generalization. Because we discuss only behavior of data model rather than object model.



(Buyer,(Contact, String,(1,)),(Tel, String,(1,1))),*

Fig. 5. Example of a Class



(PO,((Party,(1,1),(Buyer,(Contact,String,(1,)),(Tel, String,(1,1)))),*
(Request,(0,1),(Order,(Date, Date,(1,1)))),
(Details,(1,),(OrderDetails,(Goods, String,(1,*)),(Num, Int,(1,1))))))*

Fig. 6. Example of an Association

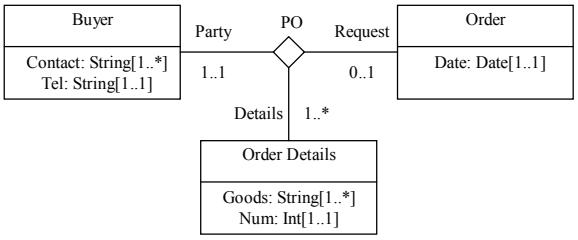


Fig. 7. Example of a Class Diagram

VI. SYNTACTICAL OPERATIONS FOR CLASS DIAGRAMS

A merger operation is defined in an analogous to union of set because same classes are merged into one class and same associations are merged into one association. A remain operation is defined in an analogous to difference of set but has little difference because the result of a remain operation should satisfy conditions of class diagram. Merger and remain operations are defined as followed:

A. Syntactical definition of merger operation

Given two class diagrams (C_X, A_X) and (C_Y, A_Y), a syntactical merger operation $\nabla : (C,A) \times (C,A) \rightarrow (C,A)$ is defined as $(C_X \cup C_Y, A_X \cup A_Y)$.

$$(C_X, A_X) \nabla (C_Y, A_Y) := (C_X \cup C_Y, A_X \cup A_Y) \quad (4)$$

Not all results of a merger operation are satisfied conditions of class diagram. Although we omit the proof because of space limitations, following conditions are formed.

Proposition 1

The syntactical merger operation (∇) is closed on class diagrams if and only if following consolidation condition is satisfied.

Consolidation condition: Let $(\mathbf{C_X},\mathbf{A_X})$ and $(\mathbf{C_Y},\mathbf{A_Y})$ be class diagrams. If names of any classes in $\mathbf{C_X}$ and $\mathbf{C_Y}$ are same, those classes are same. If name of any associations in $\mathbf{A_X}$ and $\mathbf{A_Y}$ are same, those associations are same.

$$\forall c_X \in C_X, \forall c_Y \in C_Y, Name(c_X) = Name(c_Y) \Rightarrow c_X = c_Y \quad (5)$$
$$\forall a_X \in A_X, \forall a_Y \in A_Y, Name(a_X) = Name(a_Y) \Rightarrow a_X = a_Y$$

B. Syntactical definition of remain operation

Given two class diagrams (C_X, A_X) and (C_Y, A_Y), a syntactical remain operation $-$: $(C,A)\times(C,A) \rightarrow (C,A)$ is defined as $((C_X \setminus C_Y) \cup C_{A_X \setminus A_Y}, A_X \setminus A_Y)$ where $C_X \setminus C_Y := \{c \mid c \in C_X, c \notin C_Y\}$, $A_X \setminus A_Y := \{a \mid a \in A_X, a \notin A_Y\}$ and $C_{A_X \setminus A_Y} := \{ac \mid a \in (A_X \setminus A_Y), (r,m,ac) \in Assocs(a)\}$.

$$(C_X, A_X) - (C_Y, A_Y) := ((C_X \setminus C_Y) \cup C_{A_X \setminus A_Y}, A_X \setminus A_Y) \quad (6)$$

Although we omit the proof because of space limitations, following proposition is formed.

Proposition 2

The syntactical remain operation (-) is closed on class diagrams.

C. Syntactical definition of complement operation

Given a class diagram $X = (C_X, A_X)$, a syntactical complement operation $\overline{X}$: $(C,A) \rightarrow (C,A)$ is defined as $U - X = (Class, Association) - (C_X A_X)$ where U is $(Class, Association)$.

$$\overline{X} := U - X = (Class, Association) - (C_X A_X) \quad (7)$$

Although we omit the proof because of space limitations, following lemma is formed.

Lemma 1

$$\overline{X} = U - X = (C_X{}^C \cup C_{A_X{}^C}, A_X{}^C) \quad \text{where}$$

$C_X{}^C := Class \setminus C_X = \{c \mid c \in Class, c \notin C_X\}$,
$A_X{}^C := Association \setminus A_X = \{a \mid a \in Association, a \notin A_X\}$ and
$C_{A_X{}^C} := \{ac_i \mid \forall a \in A_X{}^C, \forall (r_i, m_i, ac_i) \in Assocs(a)\}$.

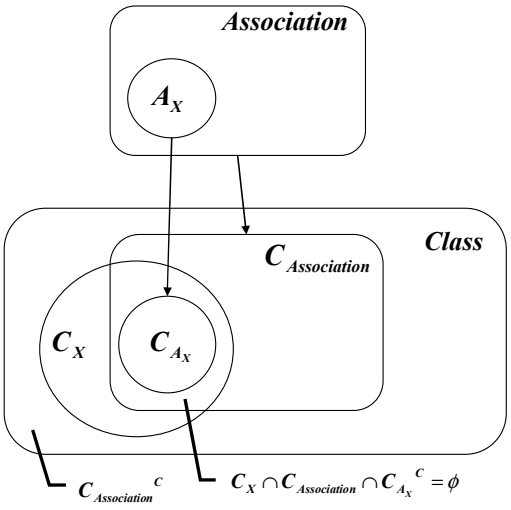


Fig.8. Restoration Condition

All results of remain operation are satisfied conditions of class diagram. However, not all results of that are satisfied double complement law. Although we omit the proof because of space limitations, following preposition is formed.

Proposition 3

The double complement law $\overline{\overline{X}} = X$ is satisfied if and only if following restoration condition is satisfied.

Restoration condition: Let (C_X, A_X) be a class diagram.

$$C_X \cap C_{Association} \cap C_{A_X}{}^C = \phi \quad \text{where}$$

$C_{Association} := \{ac \mid a \in Association, (r,m,ac) \in Assocs(a)\}$ and
$C_{A_X}{}^C := \{c \mid c \in Class, a \in A_X, (r,m,ac) \in Assocs(a), ac \neq c\}$

D. Syntactical definition of intersection operation

Intersection operation is defined with use of merger and complement operations as followed:

For any two class diagrams (C_X, A_X) and (C_Y, A_Y), a syntactical intersection operation Δ : $(C,A)\times(C,A) \rightarrow (C,A)$ is defined as $\overline{\overline{(C_X, A_X)} \nabla \overline{(C_Y, A_Y)}}$.

$$(C_X, A_X) \Delta (C_Y, A_Y) := \overline{\overline{(C_X, A_X)} \nabla \overline{(C_Y, A_Y)}} \quad (8)$$

Although we omit the proof because of space limitations, following two lemmas and one proposition are formed.

Lemma 2

The syntactical intersection operation (Δ) is closed on class diagrams if and only if consolidation and restoration conditions are satisfied.

Lemma 3

Following equations are formed if and only if consolidation and restoration conditions are satisfied.

$$\begin{aligned} X \Delta Y &= \overline{\overline{X} \nabla \overline{Y}} = U - ((U - X) \nabla (U - Y)) \\ &= (((C_X \cap C_Y) \setminus (C_{(A_X \cap A_Y)^C})) \cup C_{A_X \cap A_Y}, A_X \cap A_Y) \\ &= ((C_X \cap C_Y) \cap (C_{Association}{}^C \cup C_{A_X \cap A_Y}), A_X \cap A_Y) \end{aligned} \quad (9)$$

Proposition 4

Following equations are formed if and only if consolidation and restoration conditions are satisfied.

$$\begin{aligned} X - Y &= (C_X, A_X) - (C_Y, A_Y) = ((C_X \setminus C_Y) \cup C_{A_X \setminus A_Y}, A_X \setminus A_Y) \\ &= (C_Y{}^C \cup C_{A_Y{}^C}, A_Y{}^C) \Delta (C_X, A_X) = \overline{(C_Y, A_Y)} \Delta (C_X, A_X) = \overline{Y} \Delta X \end{aligned} \quad (10)$$

$$\overline{(X - Y)} = \overline{((C_X, A_X) - (C_Y, A_Y))} = (C_Y, A_Y) \nabla \overline{(C_X, A_X)} = Y \nabla \overline{X} \quad (11)$$

We can construct variety class diagrams with use of merger, intersection, remain and complement operations. Those operations have good properties for foundation of class diagram algebra where class diagrams are satisfied consolidation and restoration conditions.

It is safe to merge partial data models of applications describe in class diagrams with use of merger operation. It is also safe to eliminate a part of class diagram with use of remain operation. And it is easy to handle many partial data model during development of enterprise service systems.

VII. CONCLUSION

We present syntax of a class diagram describing a data model and introduce a part of foundation for class diagram algebra

with merger, remain, complement and intersection operations on class diagrams. Consolidation and restoration conditions keep consistency of class diagrams syntactically for results of those operations.

We propose a part of foundation for class diagram algebra but couldn't clear to generate any class diagrams as results of merger, remain, complement and intersection operations. We couldn't clear semantic of those operations.

REFERENCES

[1] UML 2.0 Superstructure Specification (formal/05-07-04), (2005), Object Management Group.

[2] D. Berardi, G. Calvanese, and G. D. Giacomo, "Reasoning on UML class diagrams", *Artificial Intelligence* Vol. 168, Issue 1, pp.70-118, 2005.

[3] M. Szlenk, "Formal Semantics and Reasoning about UML Class Diagram", in *Proceedings of the International Conference on Dependability of Computer Systems*, 2006, pp.51-59.

[4] K. Kaneiwa, and K. Satoh, "Consistency Checking Algorithms for Restricted UML Class Diagrams", in *Proceedings of the Fourth International Symposium on Foundations of Information and Knowledge Systems* (FoIKS2006), 2006.

[5] D. Calvanese, M. Lenzerini, and D. Nardi, "Unifying Class-based Representation Formalisms", *Journal of Artificial Intelligence Research*, vol.11, pp.199-240, 1999.

[6] D. J. Buehrer, "An Object Oriented Class Algebra", in *Proceedings of Seventh International Conference of Computing and Information* (ICCI'95), 1995, pp.669−685.

[7] M. Sabetzadeh, S. Easterbrook, "An Algebraic Framework for Merging Incomplete and Inconsistent Views", in *Proceedings of the 13th IEEE International Conference on Requirements Engineering*, 2005.

Hidekazu Enjo was born in Japan, on January 5, 1959. He received the B.Sc. and M.Sc. degree in computer science in 1981 and 1983 respectively from Tokyo Institute of Technology.

He has been working for NTT DATA CORPORATION in Tokyo, Japan. His research interests include data modeling, process modeling and XML.

Mr. Enjo is a member of ACM, a Member of Japan Society of Management Information, a member of Information Processing Society of Japan, a member of the Institute of Electronics, Information and Communication Engineers and a member of Japan Society for Software Science and Technology.

Motonari Tanabu was born in Japan, on December 15, 1969. He studied at Tokyo Institute of Technology, Tokyo, Japan. He received the B.Sc. degree in mathematics in 1993, M.Sc. and D.Eng. degrees in systems science in 1995 and 1998 respectively from Tokyo Institute of Technology.

He was a Lecturer at Faculty of Business Administration, Yokohama National University, Yokohama, Japan in 1998. He is currently an Associate Professor in MIS at International Graduate School of Social Sciences, Yokohama National University. His research interests include mathematical foundation of modeling and simulation, data mining and business gaming simulation.

Dr. Tanabu is a member of the Association for Information Systems, a member of the International Simulation & Gaming Association, a member of the Association for Business Simulation and Experiential Learning and a member of Japan Society of Management Information.

Iijima Junichi was born in Hokkaido, Japan on August 28, 1954.He has got Ph.D in systems theory from Tokyo Institute of Technology, Tokyo, Japan in 1983.

He was an assistant professor of the Department of Systems Science, associate professor of the Department Industrial Engineering and Management and currently, he is a professor of the Department of Industrial Management and Engineering, a Vice Dean of the Graduate School of Decision Science and Technology, Tokyo Institute of Technology, Japan. He is the author or co-author of many papers on systems theory and information systems including "Common Structure and Properties of Filtering Systems," PACIS2005 (Pacific Asia Conference on Information Systems) (2005), "Analytical framework for strategic alliances from the perspective of exchange of management resources," Int. J. Business Performance Management, Vol.6, No.1, 88-105(2004) and "Dependence Structure - Dependence as a binary relation," International Journal of General Systems, Vol.29, No.5, 655-670(2000). He also wrote several books on Information Systems, Systems Theory and Systems Integration. His major interests are Information Systems Integration, M-Business, IT Value and Systems Theory.

Prof. Iijima is the ex-president and currently the chair of the advisory board of JASMIN (the Japan Society for Management Information), which is one of the major societies related to Information Systems in Japan.