Lightweight Web-based Data Flow Diagraming Tool

Naravut Pattanotai[1]

[1]Faculty of Science and Technology
Rajamangala University of Technology Suvarnabhumi, Suphanburi Campus
Suphanburi, Thailand
naravut.p@rmutsb.ac.th

บทคัดย่อ—แผนภาพกระแสข้อมูล (Data flow diagram) ยังคงถูกใช้อย่างกว้างขวางในการพัฒนาซอฟต์แวร์ ข้อผิดพลาดทางไวยากรณ์ในแผนภาพกระแสข้อมูลสามารถเกิดขึ้นได้เมื่อแผนภาพนั้นไม่เป็นไปตามกฎของการสร้างแผนภาพ งานวิจัยนี้เสนอเครื่องมือในการสร้างแผนภาพกระแสข้อมูลบนเว็บขนาดเล็กซึ่งสามารถสร้างแผนภาพได้อย่างง่ายและรวดเร็ว จากโค้ดอย่างง่ายที่พัฒนาขึ้นโดยเฉพาะ เครื่องมือนี้ยังสามารถตรวจสอบความถูกต้องของแผนภาพตามกฎทางไวยากรณ์ได้อย่างอัตโนมัติ ตัวอย่างของผลลัพธ์แสดงให้เห็นถึงการใช้ประโยชน์ในการสร้างแผนภาพกระแสข้อมูล ทั้งยังช่วยลดข้อผิดพลาดทางไวยากรณ์ และสามารถนำไปใช้ในงานบนเว็บอื่นได้ต่อไป

คำสำคัญ: แผนภาพกระแสข้อมูล, กฎของแผนภาพกระแสข้อมูล, โปรแกรมสร้างแบบจำลองบนเว็บขนาดเล็ก

Abstract—Data flow diagram (DFD) is still widely used in the software development. The syntax errors in DFD can occur when the diagrams are not following its rules. This work purposes the lightweight web-based data flow diagraming tool which simply and quickly creates a DFD by the specific developed simple code. The tool also automatically validates the diagram based on formalized syntax rules of DFD. The example results show how to facilitate data flow diagramming and also to reduce syntax errors of DFD, which can use in the future other web-based works.

Keywords-data flow diagram; data flow diagram rules; data flow diagramming; lightweight web-based modeling tool

I. Introduction

Although the object-oriented analysis and design has introduced, the structured system analysis is still widely used in the software development. The most commonly used diagram in the structured system analysis is the data flow diagram (DFD). It illustrates all processes which transform the data within the system. Development project members can see all components of the system working together at once with DFD. Clients and end users can read and interpret DFD with minimal training. These are why DFD is still popular.

However, there is the rule for data flow diagramming. The errors in DFD can occur when the diagrams are not following its rules. There are many tools to data flow diagramming. Some tools are drawing tools and some are modeling tools. This work will present the lightweight web-based data flow diagraming tool to facilitate diagramming and also automatically validate it.

The rest of this paper is organized in the following manner: section II. discusses the related works on data flow diagramming followed by section III. that give an overview of DFD and its syntax rules. Section IV. is dedicated to the purposed tool and its result. Finally, the conclusion and future work are in section V.

II. Related Work

We can use a variety of tools to data flow diagramming, from freeware to commercial software. Most of the tools are the graphical drawing tools which cannot validate the diagram. However, there some tools which can validate the DFD while drawing. These tools were called modeling tool. The modeling tools help improve the quality of the result diagram, but most of them still cost highly.

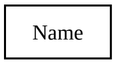
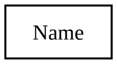
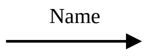
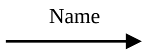
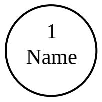
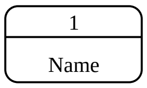
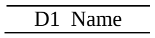
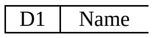
In 2010-2011, Ibrahim and Yen [1-3] purposed formalized the DFD rules for consistence check between the context diagram and level 0 DFD. They also developed the tool for drawing and checking the consistency of these two diagrams. Their tool is a desktop application. This work will present a web-based data flow diagraming tool which can use through a user's web browser.

In 2017, Pattanotai [4] purposed the pattern for DFD classes based on purposed formalized syntax rules of DFD. This facilitates data flow diagramming and also reduces syntax errors of DFD by using object-oriented paradigm. These pattern classes were implemented by using C# programming language and using JavaScript library, jsPlumb toolkit [5], to draw a data flow diagram. This work will present a text-based tool for quickly creating DFD and also automatically validate it.

III. DFD AND ITS SYNTAX RULES

DFD is a graphical system model which illustrates the movement of data between external entities and the processes and data stores within a system. [6] Truly, DFD focuses on the processes which are performed, not on data. There are four elements or symbols in DFD and there are two commonly used styles of each graphical symbol, one set developed by DeMarco [7] and the other by Gane and Sarsen. [8] These symbols were shown in Table I.

TABLE I. DFD SYMBOLS STYLES

DFD Element	**DeMarco Symbols**	**Gane and Sarsen Symbols**
External entity	 Name	 Name
Data flow	 Name	 Name
Process	 1 Name	 1 Name
Data store	 D1 Name	 D1 Name

An external entity is a person, organization, or other system which is external to the system, but supplies input data or accepts output data.

A data flow is a single piece of data, or a logical collection of several pieces of information.

A process is an activity or a function which is performed for some specific business reason.

A data store is a collection of data that is stored in some way.

There are two fundamentally different types of problems which can occur in DFDs: syntax errors and semantics errors. Syntax errors can occur when the diagrams are not followed to the rules of the DFD. Semantics errors can occur when the meaning of the DFDs is not accurately describing the business process being modeled. Syntax errors are easier to find and fix than semantics errors, because there are clearly rules which can be used to identify them. [9]

The syntax rules of each DFD element are as follows:

A. External entity

1) Every external entity must have a unique name.

2) Every external entity must have at least one input or output data flow.

B. Data flow

1) Every data flow should have a unique name. Because the process changes the data input into a different data output in some way. Therefore output data flows usually have different names from input data flows. However, there is some process which just passes the data to another without changing it. Therefore, this rule can overlook.

2) Every data flow must connect to at least one process.

3) Data must flow only in one direction.

C. Process

1) Every process must has a unique name.

2) Every process should have at least one input data flow. A process without any inputs, but produces outputs is called a miracle error because output data miraculously appear. However, this can appear when that process issues a trigger output based on an internal time clock.

3) Every process must have at least one output data flow. A process without any outputs, but receives inputs is called a black hole error.

D. Data store

1) Every data store must have a unique name.

2) Every data store must have at least one input data flow.

3) Every data store should has at least one output data flow. If there is only an output, this data store would be an external entity.

From syntax rules of DFD, there are only 5 ways to draw a data flow. They are:

1. from an external entity to a process
2. from a process to an external entity
3. from a process to a data store
4. from a data store to a process
5. from one process to other process

[4] purposed formalized syntax rules of DFD. All possible data flows which can connect from one DFD node (external entity, process, and data store) to another node was shown in Table II.

TABLE II. CONNECTION BETWEEN DFD NODES RULES

End node / **Begin node**	**External entity**	**Process**	**Data store**
External entity	×	✓	×
Process	✓	✓*	✓
Data store	×	✓	×

* Note that, a process cannot send a data flow to itself, it must send to the other process.

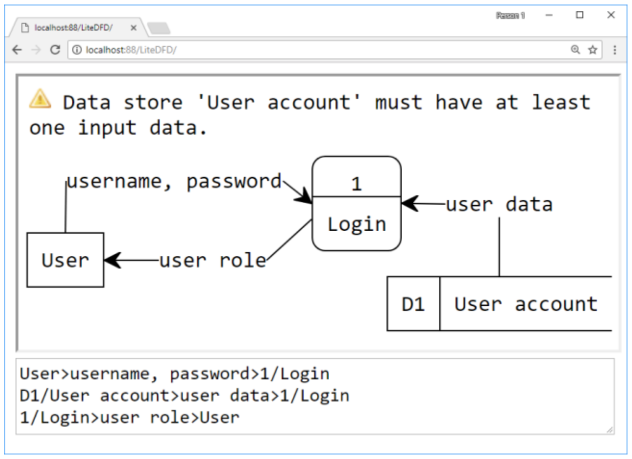
[4] also purposed the summary of input and output data for each DFD node which were shown in Table III.

TABLE III. INPUT AND OUTPUT DATA OF DFD NODES RULES

DFD node	**Input**	**Output**
External entity	must	
Process	should	must
Data store	must	should

IV. PURPOSED TOOL AND RESULT

This work develops a text-based tool for rapidly creating DFD and also automatically validate it. This tool was implemented by JavaScript and using the jsPlumb toolkit [5] to draw a data flow diagram. The example tool is shown in Fig. 1. It is composed of two main parts. The upper part shows the resulting diagram while the lower part is a text box for user entry the code of the diagram.



```
User>username, password>1/Login
D1/User account>user data>1/Login
1/Login>user role>User
```

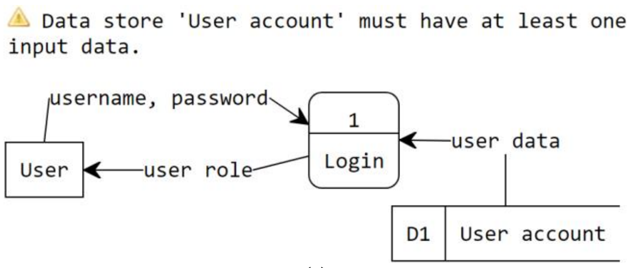
Figure 1. Lightweight web-based data flow diagraming tool

This work develops the simple code which represents DFD nodes and data flows. Then, the tool will parse these codes to DFD. The syntax code for each DFD element was shown in Table IV.

TABLE IV. SYNTAX CODE FOR DFD ELEMENT

DFD element	Syntax code
Data flow	{DFD begin node}>{Data}>{DFD end node}
External entity	{Name of external entity}
Process	{Number of process}/{Name of process}
Data store	D{Number of data store}/{Name of data store}

To diagraming the login process which similar to what is shown in Fig. 2(a), the source code of this diagram is shown in Fig. 2(b).


Data store 'User account' must have at least one input data.
username, password
1
Login
user data
User
user role
D1
User account

(a)

```
User>username, password>1/Login
D1/User account>user data>1/Login
1/Login>user role>User
```

(b)

Figure 2. Login process and its source code

From Fig. 2, each line of code is a data flow, which connects between two DFD nodes with a text data node in the middle. Each node in the data flow code separates by '>' character. The first node is the data sender and the last node is the data receiver. The '>' character also indicates the direction of the data flow, from left to right.

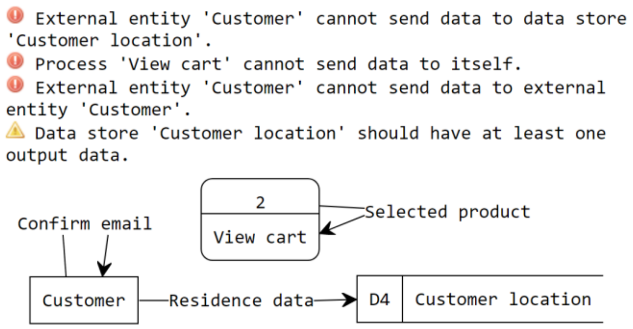
To draw an external entity node, there is not require any special markup, just type the name of that external entity node.

To draw a process, a user has to type the number of the process, following by '/' character and the process name.

To draw a data store, user is required to type 'D' character follow by the number of the data store and '/' character with the data store name.

If there is any syntax error in the diagram, the tool will show the message. In Fig. 1, there is one warning message because there is not any input data flow to the User account data store.

This tool allows user to create wrong data flows in the diagram. Considering Fig. 3, the syntax error DFD (a) and its source code (b) is shown. User can draw a data flow between an external entity and a data store, or a DFD node sends a data to itself. However, the tool will always show the error or warning messages.


External entity 'Customer' cannot send data to data store 'Customer location'.
Process 'View cart' cannot send data to itself.
External entity 'Customer' cannot send data to external entity 'Customer'.
Data store 'Customer location' should have at least one output data.
2
View cart
Selected product
Confirm email
Customer
Residence data
D4
Customer location

(a)

```
Customer>Residence data>D4/Customer location
2/View cart>Selected product>2/View cart
Customer>Confirm email>Customer
```

(b)

Figure 3. Syntax error DFD and its source code

This work selects the barbershop system to be the case study for experimentation. This shop system has two external entities: shop owner and hairdresser, 4 processes and 3 data stores. The code of level 0 DFD of the barbershop system is shown in Fig. 4. Another experimentation is a sub level DFD. The code of level 1 DFD of the maintain hairdresser process is shown in Fig. 5. Their result diagrams are shown in Fig. 6 and 7 accordingly. There is not any syntax error in these diagrams.

From all results, the codes are easier than the C# code of [4] purposed. And the tool also can be used in the variety of web-based works.

```
Shop Owner>Service data>1/Maintain Service Data
1/Maintain Service Data>Service data>D1/Service
D1/Service>Service data>1/Maintain Service Data
1/Maintain Service Data>Service data>Shop Owner

Shop Owner>Hairdresser data>2/Maintain
Hairdresser Data
2/Maintain Hairdresser Data>Hairdresser
data>D2/Hairdresser
D2/Hairdresser>Hairdresser data>2/Maintain
Hairdresser Data
2/Maintain Hairdresser Data>Hairdresser
data>Shop Owner

Hairdresser>Take care data>3/Create a Receipt
D1/Service>Service data>3/Create a Receipt
D2/Hairdresser>Hairdresser data>3/Create a
Receipt
3/Create a Receipt>Receipt>Hairdresser
3/Create a Receipt>Take care data>D3/Receipt

Shop Owner>Take care dates>4/View Receipts
D3/Receipt>Take care data>4/View Receipts
4/View Receipts>Receipts history report>Shop
Owner
```

Figure 4. Code of level 0 DFD of the barbershop system

```
2.1/View All Hairdresser Data>Hairdresser
data>Shop Owner
D2/Hairdresser>Hairdresser data>2.1/View All
Hairdresser Data

Shop Owner>New hairdresser data>2.2/Add
Hairdresser Data
2.2/Add Hairdresser Data>New hairdresser
data>D2/Hairdresser

2.3/Edit Hairdresser Data>Selected hairdresser
data>Shop Owner
D2/Hairdresser>Selected Hairdresser
data>2.3/Edit Hairdresser Data
Shop Owner>Updated hairdresser data>2.3/Edit
Hairdresser Data
2.3/Edit Hairdresser Data>Updated hairdresser
data>D2/Hairdresser
```

Figure 5. Code of level 1 DFD of the maintain hairdresser process

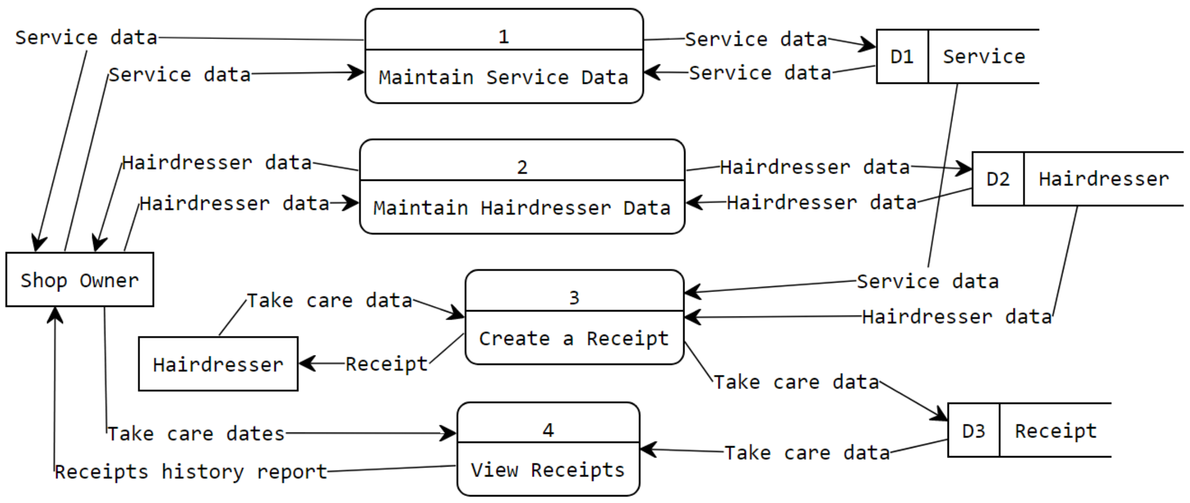


Figure 6. Level 0 DFD of the barbershop system

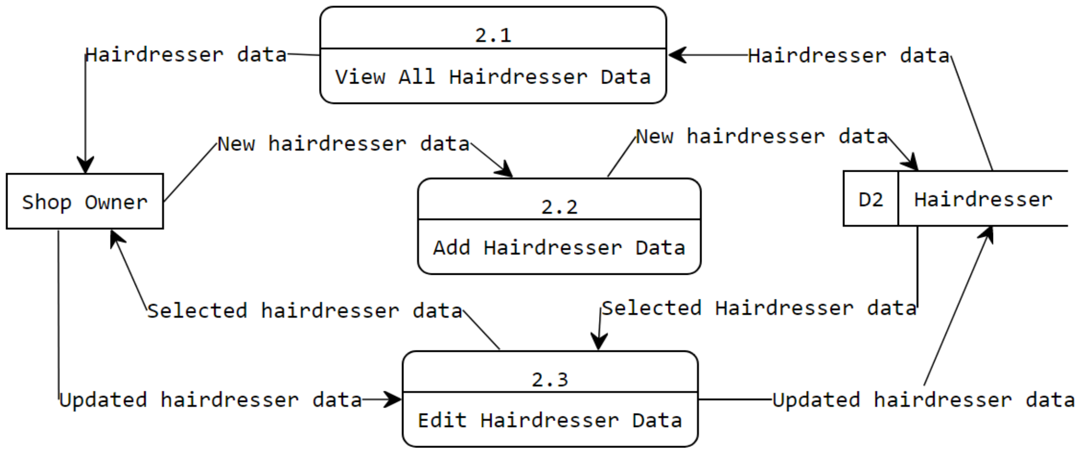


Figure 7. Level 1 DFD of the maintain hairdresser process

V. CONCLUSION AND FUTURE WORK

This work purposes the lightweight web-based data flow diagraming tool which quickly creates a DFD by the specific developed simple code. The purposed tool also automatically validates the diagram based on formalized syntax rules of DFD. The examples show how fast and easy way to diagram by the developed code and also to reduce syntax errors of DFD, which can use in the other web-based works.

However, this work focuses on modeling and validating syntax error on a separate single DFD. To check consistency through all levels of DFD and context diagram also will be addressed in further developments. Moreover, to check semantics error in DFD also may be developed. These will lead to the complete data flow diagraming tool.

ACKNOWLEDGMENT

The author would like to thank the Faculty of Science and Technology, Rajamangala University of Technology Suvarnabhumi, for supporting this work.

REFERENCES

[1] R. Ibrahim and S. Y. Yen, “Formalization of the Data Flow Diagram Rules for Consistency Check,” International Journal of Software Engineering & Applications (IJSEA), Vol. 1, No. 4, Oct. 2010, pp. 95-111, doi: 10.5121/ijsea.2010.1406.

[2] R. Ibrahim and S. Y. Yen, “An Automatic Tool for Checking Consistency between Data Flow Diagrams (DFDs),” International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol. 4, No. 9, 2010, pp. 1441-1445.

[3] R. Ibrahim and S. Y. Yen, “A Formal Model for Data Flow Diagram Rules,” ARPN Journal of Systems and Software, Vol. 1, No. 2, May 2011, pp. 60-69.

[4] N. Pattanotai, “Pattern of Data Flow Diagram Class,” Joint Conference on ACTIS & NCOBA, Jan, 2017, Thailand, pp. 27-31.

[5] jsPlumb Pty Ltd., jsPlumb Toolkit, https://jsplumbtoolkit.com/, 2017.

[6] H. J. Rosenblatt, Systems Analysis and Design, 10th ed., United States of America, 2014.

[7] T. DeMarco, Structured Analysis and System Specifi cation, Yourdon Press, New York, 1978.

[8] C. Gane, T. Sarsen, Structured Systems Analysis: Tools and Techniques, Prentice Hall, Englewood Cliffs, NJ, 1979.

[9] A. Dennis, B. H. Wixom, R. M. Roth, Systems analysis and design, 5th ed., United States of America, 2012.