Implementation of Structured Object-Oriented Formal Language for Warehouse Management System

Irfin Afifudin[1] and Inge Martina[2*]
[1−2]Faculty of Informatics Engineering, Institut Teknologi Harapan Bangsa
Bandung 40132, Indonesia
Email: [1]irfin@ithb.ac.id, [2]inge@ithb.ac.id

***Abstract*—Designing process is inseparable from software development. Like other software development processes, designing process faces many problems, such as improper and ambiguous specifications. These problems may be overcome by applying formal engineering methods. One of which is Structured Object-Oriented Formal Language (SOFL). The analysis and formation of the design and implementation of SOFL are carried out as a solution to the problem. The application of SOFL is divided into three parts according to SOFL rules, namely informal specification, semi-formal specification, and formal specification. The design and implementation are measured and tested using rigorous review and maintainability index. This research uses a warehouse management system, a safety-critical system, as a case study. Rigorous analysis shows that SOFL in warehouse management system increases the maintainability index of 56.94%. It means that it is easier to develop.**

***Index Terms*—Software Engineering, Safety-critical, Structured Object-Oriented Formal Language, Object-Based Programming, Software Quality**

I. Introduction

IN software development, a process model provides a specific roadmap for software engineering work. It defines the flow of all activities, actions, and tasks, the degree of iteration, the work products, and the organization of the work that must be done [1]. This process model is also known as Software Development Life Cycle (SDLC). The design process is an integral part of SDLC. Before starting the development, the technique of making a representation or model of software is called the design process. The problem that can occur in the design process is the incorrect design. Thus, it has a negative impact on the implementation process. In the development process, correct specifications and careful verification will significantly help prevent the unexpected result caused by errors in software components [2].

Mathematical modeling is used for the design process to provide and define a precise idea like consistency and completeness. Formal methods provide a framework for systematic specification, development, and verification system. An effective way to implement formal methods is to use Formal Engineering Method (FEM). It includes integrated specifications and verification, and all types of techniques that support the construction of specifications, transformations, and system verification and validation. Adopting FEM can reduce complexity and increase understanding, especially for large-scale and complex software [3].

Structured Object-Oriented Formal Language (SOFL) is one of the FEMs. It provides a formal but comprehensive language for design requirements and specifications and practical methods for software development [4]. SOFL provides the approach of the three-step specification, which consists of informal, semi-formal, and formal specifications. The informal specification takes the form of documentation written in a natural language. It contains the communication between software engineers and clients. The semi-formal specification is the transformation from the informal specification into SOFL notation. Then, the formal specification determines all functions by formalizing conditions before and after and identifying the system architecture using a Condition Data Flow Diagram (CDFD).

The research uses a warehouse management system. Warehouse operations play a vital role as every goods movement must be carried out and recorded correctly. It must be traceable, and any loss of goods or inaccuracy in recording results in an economic loss for the organization. Therefore, software focusing on supporting warehouse operation management is important for the organization to reduce the risks. Previous

Received: Aug. 29, 2019; received in revised form: Feb. 26, 2020; accepted: Feb. 26, 2020; available online: Mar. 27, 2020.
*Corresponding Author

researchers design and implement this software by using the traditional approach. They also employ conventional object-oriented methods. Using SOFL will apply a stricter approach. Although SOFL is more directed towards safety-critical systems [5], it can also save time and improve the accuracy of discussion and communication [6].

II. Research Method

A. Structured Object-Oriented Formal Language (SOFL)

SOFL is a formal and comprehensive language for design requirements and specifications, as well as practical methods for software development. The diagram notation used by SOFL is Condition Data Flow Diagram (CDFD), which is a formal form of Data Flow Diagram (DFD). CDFD is a graph to determine how the processes work together to provide the desired functional behavior. The CDFD is organized in a hierarchy to reduce complexity and obtain modularity for specifications [4].

B. Module

The specifications of SOFL consist of modules that are related to each other. The writing structure of the module is divided into three parts, namely the module name, CDFD, and component specifications. The components specification is the most important part of the module because it contains data and processes. Data in modules are the variables that are defined using certain types of data according to the needs. The procedure carries out tasks or instructions using the input to produce the output [4]. In the module, comments are given to explain more about everything. Module declaration using SOFL notation is depicted in Fig. 1.

module *ModuleName / ParentModuleName*;
const *ConstantDeclaration*;
type *TypeDeclaration*;
var *VariableDeclaration*;
inv *TypeandStateInvariants*;
behav CDFD_*no*;
InitializationProcess;
Process_1;
Process_2;
...
Process_n;
Function_1;
Function_2;
...
Function_m;
end_module

Fig. 1. Structured Object-Oriented Formal Language (SOFL) module declaration notation [4].

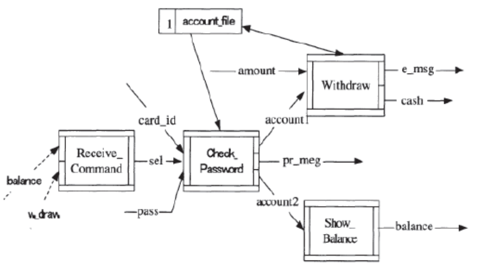


Fig. 2. Condition Data Flow Diagram (CDFD) for Automatic Teller Machine (ATM) system [4].

C. Condition Data Flow Diagram (CDFD)

CDFD is a graphical notation that shows how the processes work together to produce or provide the desired function [4]. Figure 2 is an example of CDFD of the Automatic Teller Machine (ATM) system.

Each box in the diagram represents a process, such as Receive_Command and Check_Password, that shows an operation or instruction. It also uses input and produces output. The Receive_Command process uses balance and w_draw as input, while the output of this process is sel.

There are two types of data flow for input and output. The first one is the active data flow that sends the actual data to be used by other processes and drawn in a solid line. For example, it is sel. The second data flow is control data flow sending special data that will not be used by other processes but its existence to activate the process. Control data flow is drawn in dotted lines. For example, those are balance and w_draw.

Furthermore, there is a box labeled with the number 1, account_file. It is the data store that can be a database or a file. However, in the SOFL specification, stored data are treated as ordinary variables that hold values used by processes [4].

D. Class and Object

Class is a type defined by the users. The class represents a collection of objects that have the same features. Features are attributes, descriptions of data resources, and operations that manipulate data and provide the function for other objects. Objects are instances of classes with unique identities. Figure 3 shows an example of class and objects using commonly used notations.

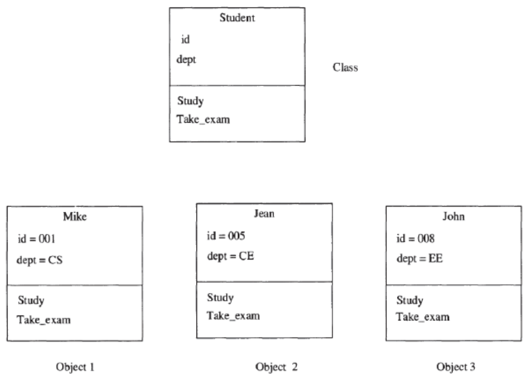


Fig. 3. Class and objects [4].

Using the example in Fig. 3, SOFL notation for objects is s: Student; and s: = new Student. Writing a class is similar to writing a module. The basic difference between classes and modules is the use of method instead of a process for object-orientation consistency. It becomes optional, and each method can be defined as an implicit or explicit specification [4].

E. Informal Specification

The informal specification is the initial step. The specification is divided into three parts. Those are functions to be implemented, the necessary constraints on both functions and resources, and the resources to be used.

In the informal specification, the relationships between functions, resources, and constraints are not given much attention. However, the most important thing is that the specification must cover function, resource, and boundary requirements as much as possible [4].

F. Semi-Formal Specification

Semi-formal specification comes from informal specifications. It aims to clarify and define all functions, resources, and constraints, and to determine the relationships of the three. Semi-formal specification acts as a vehicle that helps communication between developer and user. Semi-formal specification combines natural language with notation or formal language. The goal of this specification is to get the same understanding between user and developer. The reserved word for the semi-formal specification is listed in Table I.

G. Formal Specification

The formal specification is a transformation of the semi-formal specification. It represents the entire architecture of the software to be built. The formal specification is made using several criteria. First, all the modules are integrated into a hierarchy of CDFD. Second, all the given types are defined precisely. In other words, no given types are allowed in the formal specification because their values are not defined correctly. Third, the pre- and post-conditions of every process and function in modules are written in the SOFL language, not in any informal language.

The transformation from informal specifications to formal specifications is through a process called the evolutionary process. The evolution of this specification can be one of three possible activities, namely refinement, extension, and modification. Refinement is an activity that develops specifications by removing everything that cannot be understood. Meanwhile, the extension is the addition of new components to specifications. This new component can be a module, CDFD, process, or new data type. Finally, modification is by modifying the specifications that have been made. Modifications can be in the form of syntactic or semantic modifications without changing the suitability of the formalization standard [4]. The reserved word for the formal specification is listed in Table II.

TABLE I
THE RESERVED WORDS FOR SEMI-FORMAL SPECIFICATION [4].

No.	Reserved word	Function
1	Pre	Required condition before process execution
2	Post	Condition occurred after process execution
3	Ext	External variable, maybe from parent module or database
4	ext rd	Readable external variable
5	ext wr	Writable external variable

III. RESULTS AND DISCUSSION

A. System Implementation

The research on the application of SOFL in Warehouse Management System (WMS) will be carried out in several steps such as analysis, design, implementation, and testing of the software. Figure 4 shows a flowchart of the overall SOFL process.

For user requirements, the concept of WMS covers all activities in the warehouse. The main requirement in the warehouse is maintaining data integrity. All transaction records, whether the goods are out of the warehouse and goes into the warehouse, should be recorded as precisely as possible to avoid errors. WMS must have a clear functional, neat, and simple user interface.

A critical process in WMS can cause damage and loss of property, such as goods in the warehouse. So, the critical process in WMS affects the amount of

TABLE II
THE RESERVED WORDS FOR FORMAL SPECIFICATION [4].

No.	Reserved word	Function
1	forall	Universal quantifier. It means all elements
2	exist	Existentially quantified. It implies the variable holds many elements
3	union	Operation for unified many different sets
4	diff	Operation for getting the differences between two sets
5	modify	Operation to change field in composite data type
6	bound	Checking if a variable is defined
7	inset	Variable that is in a set
8	notin	Variable that is not in a set
9	card	Cardinality of a set, the count of elements in a set
10	subset	Subset (s1,s2) means that all elements of s1 are in s2
11	inter	Intersection of two sets
12	inds	Position of an element in a sequence
13	conc	Concatenation of two sequence
14	dunion	Unity of distributed sets
15	dinters	Intersection of distributed sets
16	power	Combination of all elements of the sets, including null elements
17	Len	Length of sequence, the number of elements in sequence
18	Dconc	Concatenation of distributed sets

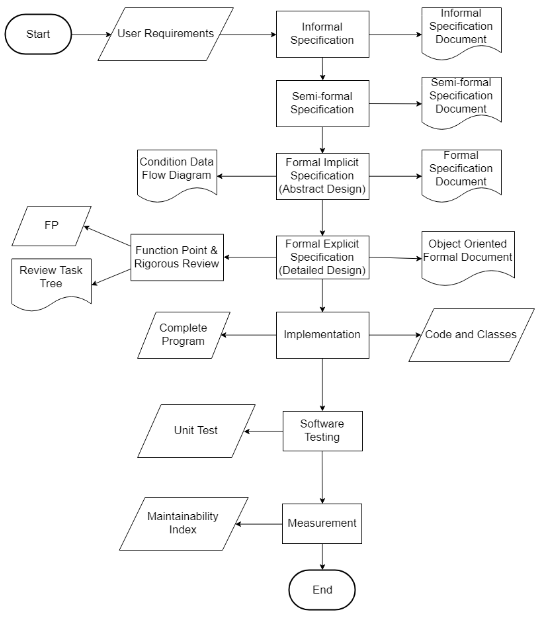

Start
User Requirements
Informal Specification
Informal Specification Document
Semi-formal Specification
Semi-formal Specification Document
Condition Data Flow Diagram
Formal Implicit Specification (Abstract Design)
Formal Specification Document
FP
Function Point & Rigorous Review
Formal Explicit Specification (Detailed Design)
Object Oriented Formal Document
Review Task Tree
Complete Program
Implementation
Code and Classes
Unit Test
Software Testing
Maintainability Index
Measurement
End

Fig. 4. Global flowchart of Structured Object-Oriented Formal Language (SOFL) process.

goods contained in the warehouse. The process that is directly related to the stock of goods in the warehouse or critical based on the design stage is as follows:

1) Three processes for the receiving module
2) Four processes for the movement goods module
3) Three processes for the issuing module
4) Zero processes for the stock report module

The top-level modules of the informal specification

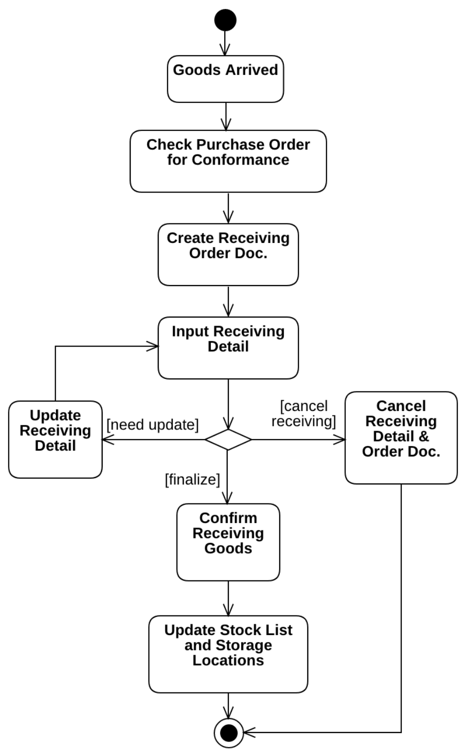

Goods Arrived
Check Purchase Order for Conformance
Create Receiving Order Doc.
Input Receiving Detail
Update Receiving Detail
[need update]
[cancel receiving]
Cancel Receiving Detail & Order Doc.
[finalize]
Confirm Receiving Goods
Update Stock List and Storage Locations

Fig. 5. Activity diagram of receiving goods at the warehouse.

are:

1) Receiving goods at a warehouse
2) Moving goods from one location to another location in the warehouse
3) Issuing goods
4) Stock records

Receiving goods is conducted when the purchased goods arrive at the warehouse. The main activities involved in this process are shown using Unified Modeling Language (UML) activity diagram in Fig. 5.

Then, moving goods are initiated when goods need to be relocated to another location. The user inputs the details of goods picking and packing. Next, the user waits for confirmation (approval). This activity is shown in Fig. 6. Moreover, goods issuing is conducted when goods ordered by the customer is ready to be delivered. The user performs outbound processes, as shown in Fig. 7.

Stock records are used by the user to get actual goods inventory stock. The purpose of this activity is to show inventory stock for goods and inventory stock grouped by storage location. Figure 8 describes this activity.

Next, the semi-formal specification must include data type, CDFD, and processes in SOFL notation. It uses a top-down approach. CDFD produced for WMS

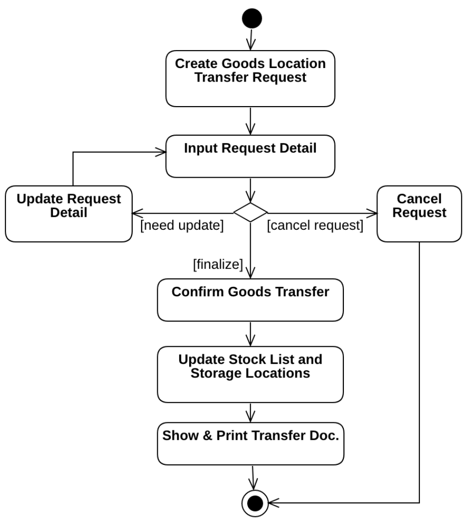


Fig. 6. Activity diagram of goods movement.

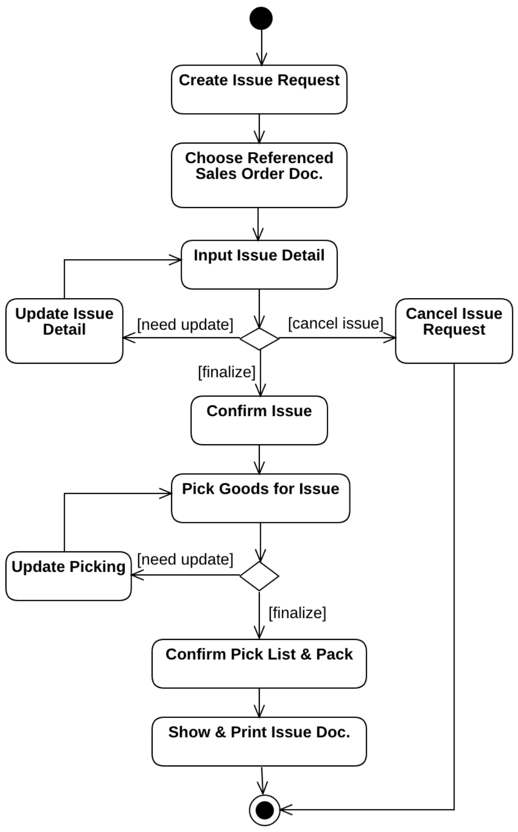
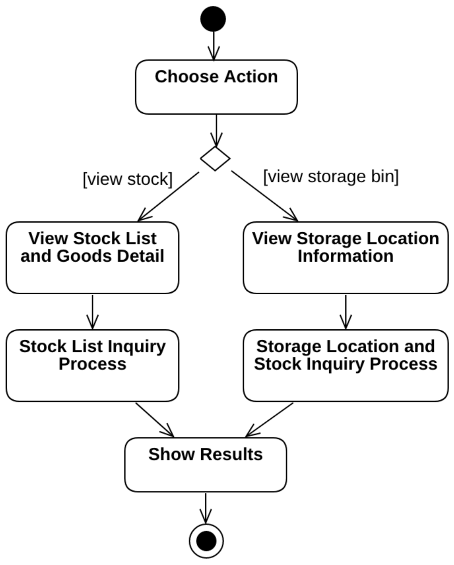


Fig. 7. Activity diagram of goods issue.

is shown in Fig. 9.

Moreover, there are two formal specifications. The first formal specification is defined using the analysis of informal specification and semi-formal specification. This first formal specification is called abstract design.



Choose Action
[view stock]
[view storage bin]
View Stock List and Goods Detail
View Storage Location Information
Stock List Inquiry Process
Storage Location and Stock Inquiry Process
Show Results

Fig. 8. Activity diagram for stock records.

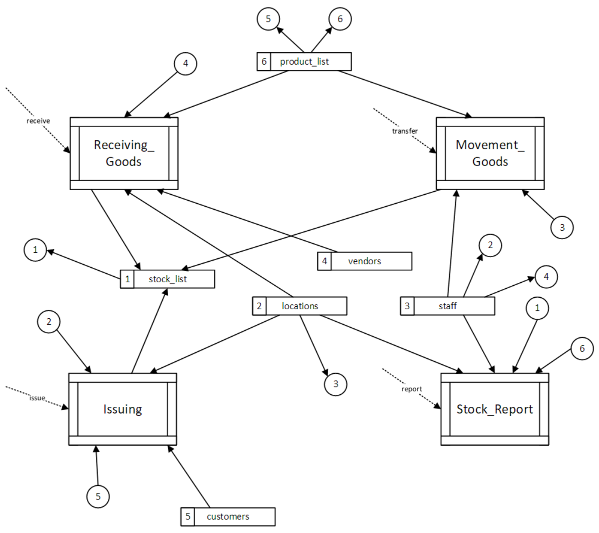


Fig. 9. Condition Data Flow Diagram (CDFD) for Warehouse Management System (WMS).

Formal specifications on the main module are broken down into a number of decomposition modules. The purpose of this decomposition is to clarify the parts of the module that have high complexity so that there are no wrong or missed specifications.

After the abstract design specification, the second formal specification may be defined, namely, detailed design. It is object-oriented. There are two important things to consider when changing an abstract design to detailed design. The first consideration is changing all-composite, product, and union data types into class. The second one is changing the implicit specification

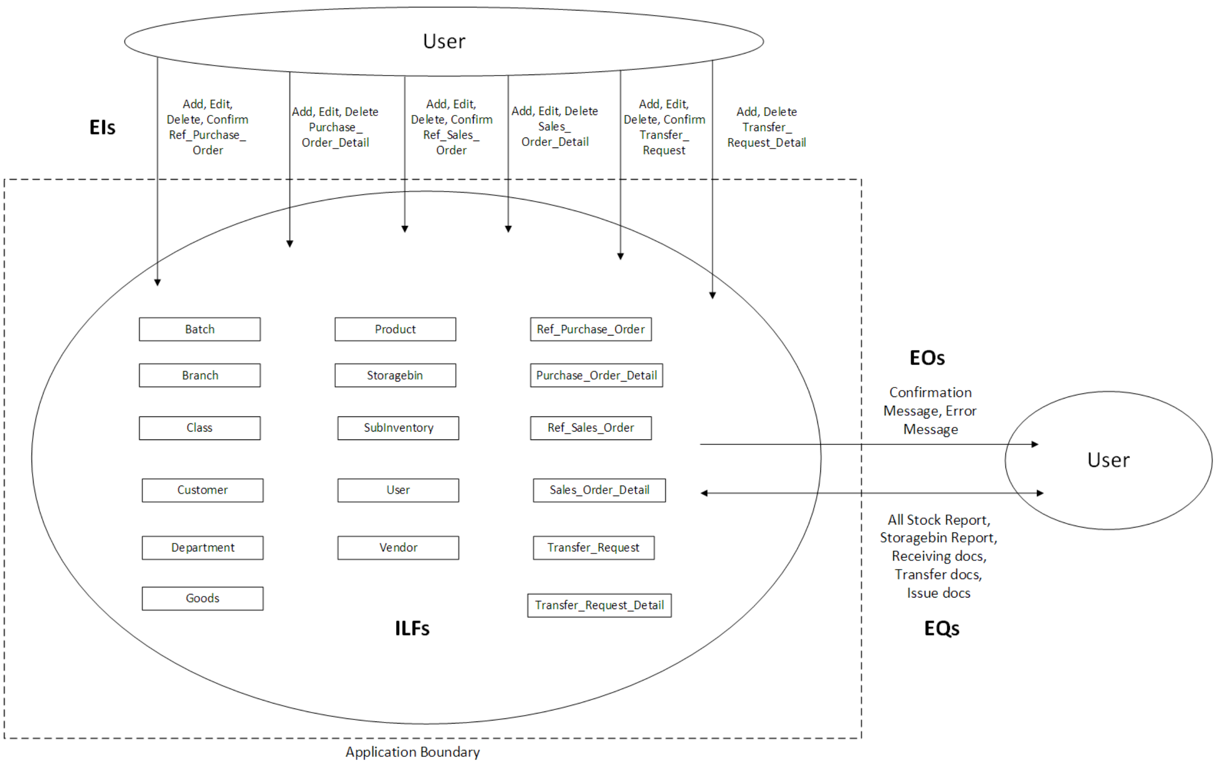
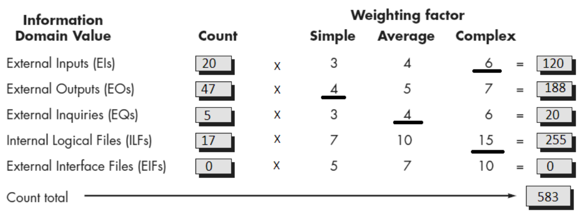

User
EIs
Add, Edit, Delete, Confirm Ref_Purchase_Order
Add, Edit, Delete Purchase_Order_Detail
Add, Edit, Delete, Confirm Ref_Sales_Order
Add, Edit, Delete Sales_Order_Detail
Add, Edit, Delete, Confirm Transfer_Request
Add, Delete Transfer_Request_Detail
Batch
Product
Ref_Purchase_Order
Branch
Storagebin
Purchase_Order_Detail
Class
SubInventory
Ref_Sales_Order
Customer
User
Sales_Order_Detail
Department
Vendor
Transfer_Request
Goods
Transfer_Request_Detail
ILFs
EOs
Confirmation Message, Error Message
User
All Stock Report, Storagebin Report, Receiving docs, Transfer docs, Issue docs
EQs
Application Boundary

Fig. 10. Function point diagram for Warehouse Management System (WMS).



Information Domain Value	Count		Weighting factor Simple	Average	Complex		
External Inputs (EIs)	20	x	3	4	6	=	120
External Outputs (EOs)	47	x	4	5	7	=	188
External Inquiries (EQs)	5	x	3	4	6	=	20
Internal Logical Files (ILFs)	17	x	7	10	15	=	255
External Interface Files (EIFs)	0	x	5	7	10	=	0
Count total							583

Fig. 11. Information domain value for Warehouse Management System (WMS).

of each process into an explicit specification.

B. Testing and Result

At this phase, the results of several measurements and tests are explained.

Function Point (FP) measurement aims to measure or estimate the cost or effort needed to design, implement, and test software. This measurement can also be used to predict the number of errors that can be encountered in testing and predicting the number of components or the number of lines of code at the implementation time.

The FP model consists of several components [7]. First, External Input (EI) moves data into the application without presenting data manipulation. Second, External Output (EO) moves data to the user and shows some data manipulation. Third, External Inquiries (EQ) moves data to the user without presenting data manipulation. Fourth, it is Internal Logical Files (ILF). The logic is in the form of fixed data managed by the application through the use of EI. Fifth, there are External Interface Files (EIF). The logic is in the form of fixed data used by the application but does not run in it.

Based on the diagram in Fig. 10, there are 17 ILF and 0 EIF because this application is not related to other applications. Then, there are 20 EI, 50 EO based on the total output of all modules, and 5 EQ.

As information domain value is obtained (Fig. 11), these values are summed from each part as count total and used by the function point formula. Value Adjustment Factors (VAF) is obtained using Eq. (1) as follows:

$$\begin{aligned} \text{FP} &= \text{count total} \times \left[0.65 + 0.01 \times \sum (F_i)\right] \\ &= 583 \times [0.65 + 0.01 \times 42] = 623.81. \end{aligned} \tag{1}$$

The value of FP as 1 is correlated to 5 lines of code. The function point of 523.81 is 3119 lines of code.

The rigorous review test focuses on three aspects, namely the consistency between process and invariant, satisfaction, and consistency of CDFD [8]. First,

TABLE III
CYCLOMATIC COMPLEXITY (CC) AND MAINTAINABILITY INDEX (MI) TEST RESULTS ON WAREHOUSE MANAGEMENT SYSTEM (WMS).

Testing	Average CC	Average MI
WMS with SOFL	3.60	96.41
WMS without SOFL	42.32	61.43

review of consistency between process and invariant aims to measure and determine the consistency between invariant and precondition in each existing process. Each process is required to meet the conditions formulated in the design.

Second, review of process satisfiability is to prove that the process can meet expectations or needs. This step tests if the precondition is correct, there will be correct post-condition. Third, review of internal consistency of CDFD is needed to review the correctness of CDFD. Internal consistency in CDFD means ensuring that output data flows from CDFD can be generated based on input data flows according to the rules of pre- and post-condition of all the processes that exist in CDFD.

Next, the researchers explain the effect of applying SOFL on the maintainability aspect. In this test, Cyclomatic Complexity (CC) and Maintainability Index (MI) of the source code produced in this research are compared to the source code produced by previous researchers that do not use SOFL. Maintainability Index is calculated using the Eqs. (2) and (3) [9]:

$$\begin{aligned} \mathrm{MI} &= 171 - 5.2\ln(\mathrm{HV}) - 0.23\cdot\mathrm{CC} \\ &\quad - 16.2\ln(\mathrm{LOC}) + 50\sin\sqrt{2.46\cdot\mathrm{COM}} \quad (2) \\ \mathrm{HV} &= N\cdot\log(2n), \quad (3) \end{aligned}$$

where HV is Halstead's volume, CC is cyclomatic complexity, LOC is line of code, COM is percentage of comments in source code, N is Program length (number of operators and operands), and n is Number of vocabulary (distinct operators and operands) in source code. The results are depicted in Table III.

Software with SOFL has a higher value compared to the software without SOFL, with an increase in MI of 56.94%. The result shows that SOFL makes software easier to maintain by prioritizing maintainability. SOFL also makes the code less complex, with an increase of 91.5% in CC. This means the level of error or bug appearances is smaller compared to the software without SOFL.

Next, the researchers explain the unit tests and their effects on applying SOFL to WMS software. This test aims to verify the smallest units of software design. The units are components or modules of the software [10]. With the unit test, all units of the WMS software have successfully passed. WMS software is successfully implemented in SOFL design.

Based on the design, implementation, and testing conducted, the use of SOFL on WMS software has a positive impact in terms of completeness without ambiguity and increasing maintainability. However, the drawback of SOFL is layered documentation requirement that spends more time.

IV. CONCLUSION

There are several conclusions of the application of SOFL for WMS software through measurements and tests. First, the measurements of function point based on Figs. 5 and 6 are better compared to previous data. However, the function points, in this case, are difficult to calculate. It is because WMS software without SOFL is an old system that has not used object-oriented architecture. Second, through rigorous review testing, it is proven that accuracy at SOFL reaches the algorithm logic level. It is not only at the design level. The application of rigorous review helps prevent errors in the design and implementation that is a great concern of the safety-critical aspects. Third, through MI test based on Table I, it is shown that SOFL gives greater MI from 61.43 without SOFL to 96.41 using SOFL. It shows an increase of 56.94%. The complexity (CC) is also improved, from without SOFL is 42.32 to with SOFL (3.60). It increases to 91.5%. Both measurements indicate that the application of SOFL has an impact on software maintainability. Last, SOFL implementation on software takes a long time, because SOFL makes layered documentation that is informal, semi-formal specification, the formal specification, and rigorous review.

In the future research, the researchers need to apply three-step specication approach in different domain safety-critical system developments. It is to confirm whether SOFL improves code complexity and increases maintainability.

REFERENCES

[1] R. S. Pressman, *Software engineering: A practitioner's approach.* New York: McGraw-Hill Education, 2014.

[2] X. Luo, S. Liu, and H. Wu, "A framework for transforming SOFL formal specifications to programs," in *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS).* Beijing, China: IEEE, Sept. 23–25, 2015, pp. 15–18.

[3] W. Miao and S. Liu, "Service-oriented modeling using the SOFL formal engineering method,"

in *2009 IEEE Asia-Pacific Services Computing Conference (APSCC)*. Singapore: IEEE, Dec. 7–11, 2009, pp. 187–192.

[4] S. Liu, *Formal engineering for industrial software development: Using the SOFL method*. New York: Springer Science & Business Media, 2013.

[5] L. E. G. Martins and T. Gorschek, “Requirements engineering for safety-critical systems: Overview and challenges,” *IEEE Software*, vol. 34, no. 4, pp. 49–57, 2017.

[6] F. Nagoya, S. Liu, and K. Hamada, “Developing a web dictionary system using the SOFL three-step specification approach,” in *2015 5th International Conference on IT Convergence and Security (IC-ITCS)*. Kuala Lumpur, Malaysia: IEEE, Aug. 24–27, 2015, pp. 1–5.

[7] H. Rohayani, F. L. Gaol, B. Soewito, and H. L. Hendric, “Estimated measurement quality software on structural model academic system with function point analysis,” in *2017 International Conference on Applied Computer and Communication Technologies (ComCom)*. Jakarta, Indonesia: IEEE, May 17–18, 2017, pp. 1–5.

[8] M. Li and S. Liu, “Tool support for rigorous formal specification inspection,” in *2014 IEEE 17th International Conference on Computational Science and Engineering*. Chengdu, China: IEEE, Dec. 19–21, 2014, pp. 729–734.

[9] I. Heitlager, T. Kuipers, and J. Visser, “A practical model for measuring maintainability,” in *6th International Conference on the Quality of Information and Communications Technology (QUATIC 2007)*. Lisbon, Portugal: IEEE, Sept. 12–14, 2007, pp. 30–39.

[10] H. K. Brar and P. J. Kaur, “Differentiating integration testing and unit testing,” in *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*. New Delhi, India: IEEE, Mar. 11–13, 2015, pp. 796–798.