Use-case and Sequence Diagram Models for Developing Transient Stability Software

Hadi Suyono, Khalid Mohamed Nor, *Senior Member, IEEE*,
Sallehhudin Yusof, *Member, IEEE*, Abdul Halim Abdul Rashid

***Abstract*—Transient stability (TS) software is a tool that is used to analyze the stability of power system after subjected a large disturbance. The transient stability software tool can be built by any approaches such as procedural programming, object-oriented programming (OOP), or component-oriented programming (COP). The component-oriented based on the object-oriented design (OOD) is proposed for developing the TS software in this paper. In the component development, use cases and sequence diagram models are required to identify the sub-system and process involved in the TS problem. The use-cases are made according to the components and objects required in the TS tool such as load flow solver, transient stability solver, linear solver, and object-oriented power system model (OO-PSM). A sequence diagram model represents the detailed process of a use case. Both use-case and sequence diagram models required for TS software tool are focused in this paper. The unified modeling language (UML) which is a standard object modeling for developing component- and object- oriented design is used to develop the use cases and sequence diagram models.**

Index Terms*— *Use-cases, sequence diagram models, object-oriented power system model, component-based development (CBD), transient stability software

I. Introduction

THE component-based development (CBD) based on the object-oriented design (OOD) is used for developing the transient stability application. The transient stability component-based software can be built by integrating any software component. Therefore, CBD application focuses on the development of the software components. A software component is designed to do a specific job, thus the component is built according to the algorithm solutions and interfaces. There are three dassifications for the software components; in-house components, wrapped components and commercial-off the shelf (COTS) components.

In-house software components are developed by own development. The wrapped components are all components that are developed from other resources and encapsulated in software components. Any software can be wrapped as a component either based on the structural programming or based on the object-oriented programming (OOP). Therefore, the wrapped components are independent of any language and programming approach. The commercial-off the shelf (COTS) components are the software components that are acquired from other third party vendors.

In object-oriented design, there are four mainly designs required such as classes, use-cases, actors, and sequence diagrams designs. Classes design comprises many involved classes that should be constructed to represent the real world problems and their relationship among them. An actor is an abstraction for entities outside the system, sub-system, or class that interacts directly with the system [3]. In other word, an actor is a user of the system that includes human user or other computer system. An actor participates in a use-case or coherent set of use-cases to accomplish an overall purpose.

A use case is a specification of sequence of actions, including variant sequences, in which a system, subsystem, or class can perform by interacting with outside actors. A use case represents an interaction between an actor and the system. Each use-case may include or extend another use case. A sequence diagram represents object interaction arranged in time sequence β]. In addition, a sequence diagram also shows the object participating in an interaction and the sequence of messages exchange. Typically, this diagram shows interactions between an actor and the object within the execution of a use-case. One sequence diagram usually represents a single use-case or flow events [6].

The components are designed according to the use-cases required in the transient stability software. There are five main software components that are needed for developing component-based transient stability software. They are load-flow, transient stability, linear solver, read ASCII data, and interface components. The load-flow, transient stability, and read ASCII data components are developed as in-house components. The linear solver is wrapped from the public domain that was developed by structural programming, and then later it is packaged and encapsulated as software component. The interface components are done by TeeChart components as COTS software for charting 2D and 3D.

II. Unified Modeling Language (UML) and Component Objects

A. Component Objects

In general, the thing that can be touch and seen is called

This work was supported in part by the University of Malaya, Kuala Lumpur Malaysia, under IRPA grant project.

Hadi Suyono is with the PT. APS Indonesia (APSI), (e-mail: hadis@aps-my.com).

Khalid Mohamed Nor is with the Faculty of Electrical Engineering, Universiti Teknologi Malaysia (UTM) (e-mail: khalidmn@fke.utm.my).

Sallehhudin Yusof is with Advance Power Solution Sdn. Bhd., (e-mail: salleh@pc.jaring.my).

Abdul Halim Abdul Rashid is with the Institute of Mathematic, Faculty of Science, University of Malaya, (e-mail: ahar@um.edu.my).

1-4244-0273-5/06/$20.00 ©2006 IEEE

Authorized licensed use limited to: UNIVERSITY TEKNOLOGI MALAYSIA. Downloaded on January 8, 2009 at 03:32 from IEEE Xplore. Restrictions apply.

as an object. An object, in the object oriented paradigm, is one of many things that together constitute the problem domain [2]. An object may stand alone or it may belong to a class of similar objects. An object is a discrete entity with a well-defined boundary and identifies that encapsulated state (information or data) and behavior (function or operation) [3] which remember the effect of this operation [1] for a specific set of purposes [4]. In addition, an object is a distinct instance of a given class that encapsulates its implementation details and is structurally identical to all other instances of that class [5]. The object that represents the real-world problem can be characterized based on what reasons to create the object. The object can represent a physical product or a functionality of the sub system. Therefore, the created object should characterize the adequate representation.

Jacobson *et al.* classified the objects into three types; interface objects, entity objects, and control objects [1]. The interface objects model behavior and information related to the presentation of the system to the outside world. By using the interface objects, a user can communicate with the system. In other word, the main duty of the interface objects is to exchange data between the user and system. Entity objects model information that exists in the system for a longer time. Typically, the problem domain objects are presented as entity objects. Control objects model functionality that is naturally tied to other object types. Some entity objects are passed to a control object, perform a computation, and then return the result though an interface object that will latter be displayed to the user. The three objects based on the UML diagram are depicted in Fig. 1.

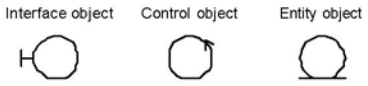

Fig. 1. Objects classification by Jacobson et al. [1]

B. *Unified Modeling Language (UML)*

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems [7]. The UML provides significant benefits by helping to build rigorous, traceable and maintainable models, which support the full software development lifecycle. Designing and developing object oriented software with its involved process can be presented by the UML that uses mostly graphical notations to express the design of software projects.

By using the UML, classes design, use-case models, actors, and sequence diagrams for developing software can be provided.

III. Use-case Models For Transient Stability Software

In the transient stability software, there are four mainly use cases. They are load-flow, transient stability, liner solver, and object-oriented power system model. Each use case is given as the following:

A. *Load-flow Use-case*

The load flow analysis is the main analysis that is required by other analyses such as short-circuit analysis, transient stability analysis, and harmonic load-flow analysis. The load flow use case is given in Fig. 2 The actor may be any power system analyses that request the load-flow solution. Besides, the actor may also be engineers or operators who perform the load-flow solution as a single application. The actor also includes other use-case i.e. transient stability use-case, since the transient stability reuses the load-flow use-case.

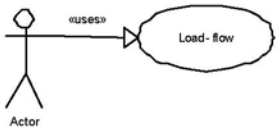

Fig. 2. Load-flow use-case

There are two algorithms that can be used for solving the load-flow problem. They are Newton Raphson and Fast-decoupled methods. Therefore, use-cases for load-flow solution can be based on the two methods as depicted in Fig 3. The actor for these use-cases is the load-flow use-case as given in Fig. 2.

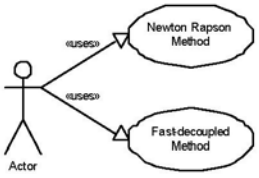

Fig. 3. Load-flow solution algorithm use-cases

B. *Transient Stability Use-cases*

Use-case for transient stability is given in Fig. 4. Unlike load-flow use-case that can be reused for other analyses; the transient stability use-case is only used for the transient stability analysis application. Therefore, the involved actor in the transient stability use-case is operators or engineers

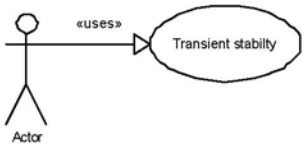

Fig. 4. Transient stability use-case

who perform the application.

Since the Modified Euler method and Trapezoidal method are chosen for solution algorithms, the transient

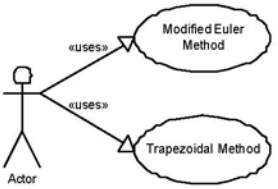

Fig. 5. Transient stability algorithm use-cases

Authorized licensed use limited to: UNIVERSITY TEKNOLOGI MALAYSIA. Downloaded on January 8, 2009 at 03:32 from IEEE Xplore. Restrictions apply.

stability algorithms have then two use-cases. The transient stability use-cases model is given in Fig. 5. The actor that employs these use-cases is the use-case that is shown in Fig. 4.

C. Sparse Linear Solver Use-case

Many power system analyses are presented as linear equations. Therefore, the sparse linear solver use-case can be used for solving the linear equation problem. In the load flow analysis, the sparse linear solver is used for solving the power mismatch equations in terms of Jacobian matrix. In the transient stability, the sparse linear solver is used for solving current/voltage equations in terms of the admittance matrix. Besides, the linear solver can also be used for other analyses for solving any linear equations. The sparse linear solver use case is given in Fig. 6. The involved actors for the sparse linear solver use-case may include many analyses that can be considered for solving linear equations such as short-circuit analysis. In this paper, use-cases as depicted in Fig. 3 and Fig. 5 are actors for the linear solver use-case.

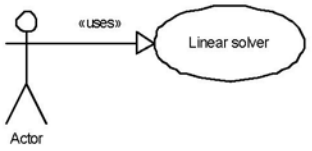

Fig. 6. Sparse linear solver use-case

D. Use Case for Object-oriented Power System Model

The object-oriented power system model (OO-PSM) use-case is used to represent a sequence of transaction performed by the system in the transient stability analysis. The use-case model for OO-PSM is given in Fig. 7. There are two use cases in the OO-PSM, read ASCII data for load-flow analysis use-case and read ASCII data for transient stability use-case. The actor in this use-case model is OO-PSM that invokes both read ASCII data for load flow and transient stability use-cases. The OO-PSM can be developed after both use-cases are performed. The first use-case is employed to develop the OO-PSM that is used only in the load-flow algorithm, whereas the second use-case is utilized to build the OO-PSM that is used in the transient stability algorithm.

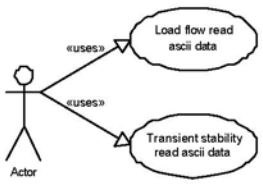

Fig. 7. Input data use-cases

IV. Sequence Diagram Models For Transient Stability Software

A sequence diagram model is needed to represent interactions in sequence between some objects in a use case as discussed in the previous section. The interactions are modeled as sequence of messages exchange. This diagram shows interactions between an actor and the object within the execution of a use-case.

A. Load-flow Sequence Diagram

In the load-flow use case, there are some requirements that are listed in the sequence diagram model as given in Fig. 8. In this sequence diagram, the actor may be the transient stability or load-flow use-cases. The requirements in this sequence diagrams are:

a) The actor requests the load-flow results that are usually given by voltage magnitude and angle. The load-flow results will be presented by using interfaces.
b) Interfaces display the result based on the data or variables requested. Since the load-flow is developed according to the object-oriented design, hence the data are presented in the OO-PSM.
c) The updated OO-PSM data can be done after the load-flow algorithm is performed. The load-flow algorithms can apply either Newton Raphson or Fast-decoupled methods.
d) The load-flow algorithms need to calculate and evaluate iteratively the power mismatch equations. Only after the convergence of the power mismatch the load-flow calculation can be stopped.
e) The power mismatch for each bus can be calculated by using the linear solution in terms of the Jacobian matrix for both Newton Raphson and Fast-decoupled methods. Therefore, in this stage, the linear solver is required. As a result, the new voltage magnitude and angle for each bus will be obtained.

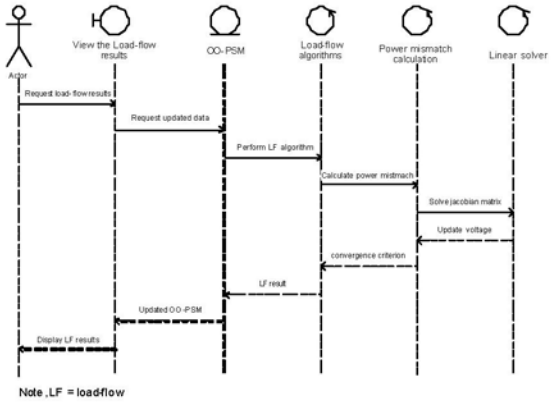

Fig. 8. Load-flow sequence diagram

B. Transient Stability Sequence Diagram

The sequence diagram model for the transient stability use-case is given in Fig. 9 Requirements for transient stability sequence diagrams are:

a) The actor requests the transient stability solution. Some procedures should be done such as running the load flow application and capturing the transient stability variables before performing the transient stability simulation. Usually, the rotor angles of synchronous

Authorized licensed use limited to: UNIVERSITY TEKNOLOGI MALAYSIA. Downloaded on January 8, 2009 at 03:32 from IEEE Xplore. Restrictions apply.

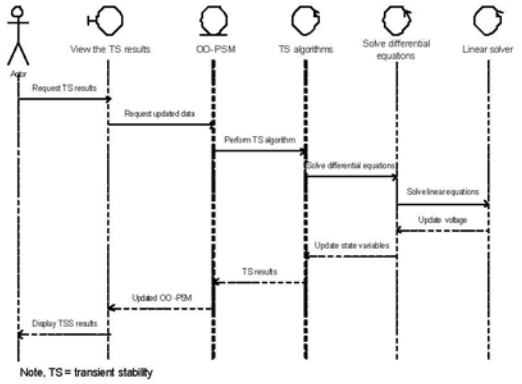

Fig. 9. Transient stability sequence diagram

machines are main variables captured. The transient stability results will be provided by using interfaces.

b) Like the load flow sequence diagram, the interfaces display the result based on the data or variables requested, that are provided in the OO-PSM.

c) The updated OO-PSM data can be provided after performing the transient stability algorithm. Modified Euler or Trapezoidal algorithm can be performed for solving the transient stability problem.

d) In the transient stability algorithms, the involved differential equations are transformed into algebraic equations for both Modified Euler and Trapezoidal methods.

e) The linear solver is needed in both Modified Euler and Trapezoidal methods for solving the voltage equation in terms of the admittance matrix.

C. *Sparse Linear Solver Sequence Diagram*

Sparse linear solver sequence diagram for transient stability use-case is given in Fig. 10. Requirements for transient stability sequence diagrams are:

a) The actors that request the linear solver are load-flow and transient stability use-cases. In the transient stability analysis, the linear solver is needed for solving YV = I.

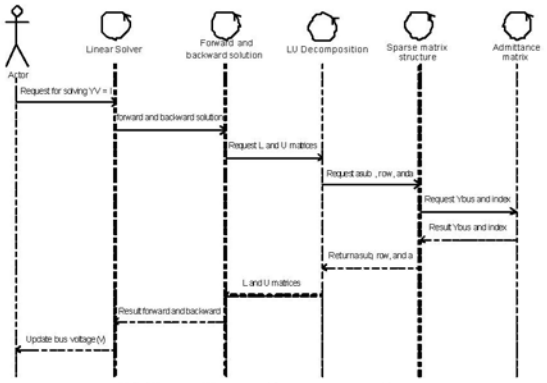

Fig. 10. Sparse linear solver sequence diagram

b) For solving the linear equation, forward and backward solutions are needed, since LU factorization is applied for network admittance matrix Y. Forward solution is used for solving the voltage in terms of L matrix, whereas backward solution is used for solving voltage in terms of U matrix.

c) The previous sequence can only be done, after L and U matrices are provided. L and U matrices is decomposed from the admittance Y matrix. This procedure can be completed in sequence LU decomposition.

d) Since LU decomposition is presented in sparse technique that only non-zero elements are stored, therefore the matrix structure should be constructed as a compressed column or compressed row format. There are three structure arrays that should be prepared such as *a*[], *asub*[], and *xa*[] that correspond to nonzero elements of network admittance matrix, row or column indexes, and indices indicating the beginning of each column in coefficient and row index arrays.

D. *Sequence Diagram for Object-Oriented Power System*

Since, there are two use cases to realize the OO-PSM, two sequence diagram models are then given in Fig. 11 and 12. In Fig. 11, OO-PSM that is used in the load-flow solution is developed. AC-Generator, transmission line, transformer, and load objects are created. These objects are developed by supplying the load-flow data (text file) through the interface object component for load-flow read ASCII data.

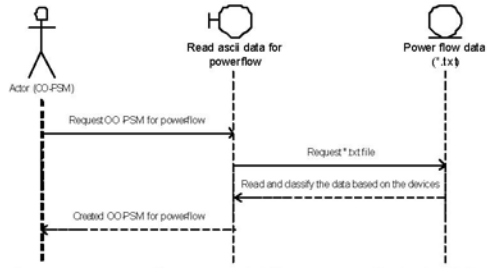

Fig. 11. Sequence diagram model for OO-PSM in the load-flow

The sequence diagram model for developing the OO-PSM for transient stability solution is given in Fig. 12. The OO-PSM includes the synchronous machine, excitation system, turbine governor, and SVC objects. These objects can be realized by requesting the dynamic data (*.dyr) though read machine and controllers interface that are encapsulated in software component. The OO-PSM that has been developed in the load flow solution is also reused in the transient stability solution.

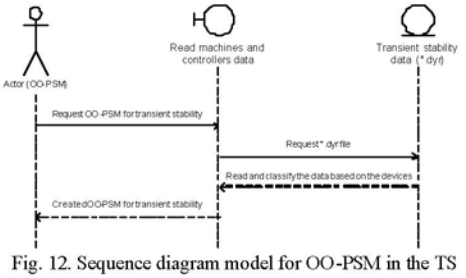

Fig. 12. Sequence diagram model for OO-PSM in the TS

V. Test Systems And Numerical Results

All components required such as load-flow, transient stability, linear solver, and interfaces components as discusses in the previous section are integrated for

Authorized licensed use limited to: UNIVERSITY TEKNOLOGI MALAYSIA. Downloaded on January 8, 2009 at 03:32 from IEEE Xplore. Restrictions apply.

developing the transient stability component-based application software. The transient stability component-based application software was tested by the IEEE data test, such as the 9-bus, 39-bus, and 162-bus test systems. The test was carried out using a computer with Intel Pentium IV/2.2 GHz processor and 512 MB main memory. The comparison between component-based software application and structural programming application is made. The execution time as a main comparison is accomplished through the capture of dynamical performances of the transient stability software application when the fault occurs at bus 7, and the clearing time is at 0.7 s, while the maximum time of study is at 2.5 s.

Table I shows the comparison results. The differences of component-based and structural programming transient stability software are insignificant. The percentage differences of execution time for both Modified Euler method and Trapezoidal method are about 10% for 17 machines, 2% for 10 machines, and 5% for 3 machines. The component-based application consumes more time as compared with structural programming, because of the fact that component-based application needs to call library files which are separated from the application files. On the other hand, for structural programming application, function calling takes place in the same memory block of the application, which makes the execution time less than that of component-based application.

TABLE I
EXECUTION TIME COMPARISON (IN SECOND)

Data Test	Methods	Component-based	Structural-programming	Diff (%)
9-bus	M. Euler	0.0200	0.0190	5.2632
	Trapezoidal	0.0200	0.0190	5.2632
39-bus	M. Euler	0.0640	0.0630	1.5873
	Trapezoidal	0.0650	0.0640	1.5625
162-bus	M. Euler	0.2330	0.2120	9.9057
	Trapezoidal	0.2240	0.2050	9.2683

The mathematical solution algorithms are frequently subjected to be improved or changed with better capabilities algorithms. Consequently, these algorithms were modeled as independent software components with minimum coupling. Therefore, any of the components can be reused or replaced without affecting the other components in the software. The proposed design involved more than one component that can perform the same power system analysis. Therefore, the proposed design was generalized such that the software components that perform the same analysis should have the same interface. Therefore, all involved solver for transient stability software and object-oriented power system model are designed such that they have the same interface.

VI. CONCLUSION

The paper has been presented the design and development of component-based transient stability software. Since the transient stability component-based software is based on the object-oriented design (OOD), thus the use-cases and sequence diagram models required for component-based software are needed. The uses-cases and sequence diagram models are developed according to the software components required in the component-based software. The CBD of transient stability solution requires some software components to be integrated. Software components that have been built include in-house, wrapped, and COTS components. Load-flow, transient stability, and data input components have been developed as in-house components. The wrapped components have been realized for developing the linear solver components. The COTS components have been acquired from other vendors. The COTS components are required for graphical user interfaces such as charting, and windows application. In addition, the components have also been developed as independent components. Therefore, updating and maintaining the components can be done without affecting other components or software application that uses these components.

REFERENCES

[1] Jacobson, I.; Christerson, M.; Jonsson, P.; Overgaard, G; Object-Oriented Software Engineering, A Use Case Driven Approach, Addison-Wesley , 1992

[2] Behforoos, A. and Hudson, F.J.; Software Engineering Fundamental, New York: Oxford University Press, 1996

[3] Rounbough, J.; Jacobson, I., Booch, G.; The Unified Modeling Language Reference Manual, Addison-Wesley, USA, 1999

[4] Zamir, S.; Handbook of Object Technology, CRC Press LLC, 1999

[5] Kafura, D.; Object Oriented Programming, Available in: http://people.cs.vt.edu/ ~kafura/cs2704/oop.swe.html

[6] Spark, G; An Introduction to UML – The Use Case Model, available at: http://www.sparxsystems.com.au

[7] OMG, Unified Modeling Language: Superstructure Ver 2.0, July, 2004

BIOGRAPHIES

Hadi Suyono *was born in East Java, Indonesia on May 20, 1973. He graduated from Department of Electrical Engineering at University of Brawijaya, Malang-Indonesia in 1996. He obtained his M.Eng and Ph.D from University of Gadjah Mada Indonesia in 2000 and University of Malaya in 2006 respectively. Currently he is a Senior Consultant in PT. APS-Indonesia (APSI). His major interests are application of computer in power system and software engineering.*

Khalid Mohamed Nor *was born in Sungai Pelong in Selangor, Malaysia. He graduated with First Class Honors in Bachelor of Engineering from the University of Liverpool, England. He later obtained his MSc in 1978 and PhD in 1981 from the University of Manchester Institute of Science and Technology, England. He joined the University of Malaya, Malaysia as a lecturer in 1981 and currently is a professor in the department of electrical engineering in the said university. He is a Senior Member of IEEE. His current research interests are in the field of electrical power system simulation and power quality.*

Sallehhudin Yusof *was born in Perak, Malaysia in 1954. After graduating from Southampton University in 1978, he worked in TNB, Malaysia for 17 years in various areas of business and engineering. He obtained MEE degree from UTM Malaysia 1989 and PhD degree from McMaster University, Canada in 1993. Salleh worked for PTI-Asia between 1995 and 2000. In November 2000, Salleh and colleagues formed Advanced Power Solutions (APS), a Strategic Global Partner of Shaw PTI. Since with APS, Salleh has been heavily involved in development and worldwide supports of Shaw PTI software products. In addition, he continues to provide consulting and educational services in the region. He is a member of CIGRE and the IEEE.*

Authorized licensed use limited to: UNIVERSITY TEKNOLOGI MALAYSIA. Downloaded on January 8, 2009 at 03:32 from IEEE Xplore. Restrictions apply.