Transient Stability Program Using Component-Based Software Engineering

Hadi Suyono
Department of Electrical Engineering,
University of Malaya
Kuala Lumpur, Malaysia
hadis@um.edu.my

Khalid Mohamed Nor
Department of Electrical Power Engineering
University of Technology Malaysia (UTM)
Johor, Malaysia
khalidmn@utm.my

Sallehhudin Yusof
Advance Power Solution Sdn. Bhd
Kuala Lumpur, Malaysia
salleh@pc.jaring.my

Abstract—**This paper presents a development of transient stability analysis (TSA) software by applying component-based software engineering (CBSE). The TSA application needs software components to be integrated such as linear solver components, load flow analysis (LFA) components, and TSA components. The TSA components are built independently from the LFA and other components. Therefore, the TSA components can be integrated with any load flow package. The power system devices are represented as entity objects and then encapsulated in an independent class hierarchy. In this development, the same object-oriented power system model (OO-PSM) that has been used in the LFA components is reused for developing the TSA components, but it needs to be extended to model new devices such as synchronous machines, exciters, speed-governors, turbines, PSS, and SVC system. The performance of the proposed TSA software was tested with large systems and compared with structural programming. The results exhibit that there is no much difference in the execution time regarding to the quality of the component-based TSA application such as saving in the development resources.**

I. INTRODUCTION

Component-based Software Engineering is the methodology that builds a software application by assembling many software components. A software component is built and tested before integration into an application. A component has a specific functionality since it is independent and it can be replaced with other component has the same role and interface. On the other hand, the component sometimes needs to be modified or updated because of maintenance reason. Modifying and updating a component can be done without affecting other components, since it is supposed that both the role and the interface are not changed. The software component can be implemented either structural design or object-oriented design (OOD). The OOD offers some a great reusability that can be found in its features such as inheritance, association, polymorphism, and encapsulation. In the OOD, a class is defined as user-defined data abstraction and methods for an object. By using objects and classes, an application can be divided into small, more manageable pieces that are more closely matched to the real structures and behaviors of existing systems.

The OOD and object-oriented programming (OOP) have been widely used in many power system applications, including simulation for LFA and fault analysis [2]-[3], dynamic stability analysis [4], power system education purpose [5], and for graphical user interface [6]. Nevertheless, most of the previous designs are only based on the inheritance approach. The design has also coupled the solver algorithm inside the class hierarchy. Thus, the solver has class dependencies or class deep dependencies that make it difficult to be updated, extended, maintained, or replaced. This drawback can be eliminated by separating the solver algorithm from the object oriented power system model (OO-PSM) that presents the devices of an electrical network. The paper proposes a new design of the OO-PSM that will be developed by utilizing inheritance, association, and a combination of the inheritance and association approaches, whereas the solution algorithm is built as software components. The application is developed by integrating both the load flow and transient stability software components. Numerical results are provided in the end of the paper to test the performance of the proposed component-based TSA application.

II. TRANSIENT STABILITY PROBLEM

A. *Transient Stability Equation*

The transient stability analysis is used to evaluate the ability of an electric power system to regain the state of the operating equilibrium after being subjected to a physical disturbance [7]. The stability performance of the power system depends on the type of disturbance and the initial operational condition. When a large disturbance subject to power system, the voltages will drop, and if this situation occurs for a long time, the synchronization will be lost. It may even lead to the power system blackout. Examples of large disturbances are short circuit fault, loss of loads, and loss of generations.

Generally, the transient stability of power system is highly non-linear. It can be mathematically expressed as:

$$\dot{x} = f(x,V) \tag{1}$$

where, x is the state variable, f is the non-linear vector function, and V is the bus voltage vector. Because of the state variable depends on the voltage, it will be updated for along study. On the other hand, the new voltage vector can be obtained by solving the network equation that is formulated as follows:

$$YV = I(x,V) \tag{2}$$

where I is the vector of injected current, and Y is the complex admittance matrix.

B. *Power System Model*

Since many devices are connected to the power system, they should be mathematically modeled to run a power system analysis. A synchronous machine is represented with a model that depends on the damper in two *d*- and *q*-axis considered. The damper is indicated by the number of winding. There are six machine models, ranging from classical to complete models [8] [9].

A load in the power system analysis is commonly modeled as a static power load that is represented by a MVA rating at a particular bus. The static load can be represented as a function of voltage magnitude and frequency [10].

An excitation system is used to control the generator terminal voltage via the adjustment of generator field current. There are three types of excitation system, depending on the basis of the excitation power source; DC, AC, and ST excitation system [11].

Power system stabilizer is an element that provides an additional input to the voltage regulator to improve the dynamic performance of the power system. A speed governor and turbine models are used to control the mechanical torque and mechanical power variables during the dynamic simulations [12].

A Static VAR compensator (SVC) is equipped to maintain voltage levels, and to improve the power with the injection of a controlled capacitive or inductive current with a specific variable [13]. The SVC is widely used in the transmission system.

Usually all machines and their controllers are involved in the TSA. They are represented from a composition of block diagrams that are given in the frequency domain. Since step-by-step solution is applied, the equation in the frequency domain is converted to the differential equation in the time domain. Furthermore, the numerical integration method can be implemented to solve the differential equation.

III. Numerical Integration Method

The numerical integration method has two techniques; explicit and implicit. Modified Euler method is applied as an explicit technique, whereas the Trapezoidal method is used as an implicit technique. The Modified Euler method needs two equations to solve equation (1), namely initial and final estimate equations. The initial estimate equation is given by:

$$x^{(n)} = x^{(o)} + \Delta t \, \mathrm{f}(x^{(o)}, V^{(o)}) \tag{3}$$

where (*n*) denotes the current estimate, (*o*) denotes the previous value, and Δ*t* indicates the integration step. Since the voltage *V* also depends on the current *I*, which in turn is the function of the state vector, the new estimate for voltage is:

$$YV^{(n)} = I(x^{(n)}, V^{(o)}) \tag{4}$$

Then, the final value of x is solved by:

$$x^{(f)} = x^{(o)} + \frac{1}{2}\Delta t \left[\mathrm{f}(x^{(o)}, V^{(o)}) + \mathrm{f}(x^{(n)}, V^{(n)}) \right] \tag{5}$$

and the final voltage can be updated by using:

$$YV^{(f)} = I(x^{(f)}, V^{(n)}) \tag{6}$$

where (*f*) denotes the final value.

The Trapezoidal method is the most popular choice for TSA study. The implementation of this method in equation (1) produces an algebraic equation which is expressed as:

$$x_n = x_{n-1} + \frac{1}{2}\Delta t \left[f(x_{n-1}, V_{n-1}) + f(x_n, V_n) \right] \tag{7}$$

and the network equation is derived from equation (2) is given as follows:

$$YV_n = I(x_n, V_n) \tag{8}$$

where *n* is the current value and *n-1* indicates the previous value.

IV. Object-oriented And Components Design In The Transient Stability Analysis

The objects in the TSA design are divided into three major groups i.e. entity objects, control objects, and interface objects. The power system devices are modeled as entity objects. They are represented as the OO-PSM, whereas the solution algorithm or analysis objects are modeled as control objects. They are created as software components. Finally, the interface objects include the objects that handle the communication between the analysis objects and their clients.

A. *Software Architecture of the TSA*

The software architecture is defined as the structure of the system which comprises software components, the externally visible properties of these components, and the relationship among them [14]. The architecture design of the TSA is shown in Fig. 1. There are two groups of components that compose the architecture: analysis components and interface components. The analysis components correspond to the algorithms such as computational or functional algorithms that can be employed to manipulate the data.

The analysis components include the LFA, TSA, and linear solver components. The interface components are required to transfer the data to the analysis components from raw sources, and also to display the input or output of the analysis results. The ReadAsciidata, ReadMachinedata, TeeChart Pro, and DBCAD are categorized as interface components.

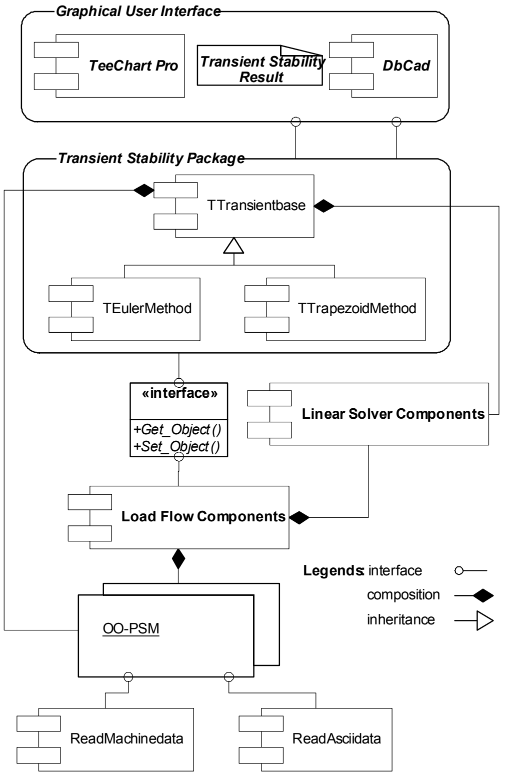


Figure 1. The architecture design of the TSA software

The ReadAsciidata component is employed to prepare the LFA data which contains bus and branch network data. The ReadMachinedata component is used to read the machine data and its controllers. The OO-PSM will be created and then associated in the LFA as well as TSA components. If the LFA components are firstly executed, the OO-PSM has existed. Then the OO-PSM can be transferred to the TSA components by using the interfaces Get_Object and Set_Object.

The LFA and TSA analyses need a sparse linear solver for their solution process. The solver is developed in independent component. The linear solver is the modules which are acquired from the public domain. The modules are implemented through structural programming and wrapped as well as encapsulated into components. The solver component uses the standard sparse data storage formats such compressed column or row formats. Therefore the solver components can be replaced at any time with a component has better performance without altering the TSA software. In the LFA, the linear solver is used to solve the power mismatch in terms of Jacobian matrix, whereas in the TSA, the linear solver is utilized to solve equation (2).

The TSA components consist of three classes; TTransientbase, TMEulerMethod, and TTrapezoidalMethod classes. The TTranseintbase is a base class that includes general attributes and functions which can be accessed by descendent classes. The TMEulerMethod and TTrapezoidalMethod are derived from TTransientbase which are linked by inheritance concept. These classes correspond to Modified Euler and Trapezoidal methods respectively. Since the software application tools need the solver to run the computational purposes, only the solver classes are then encapsulated into the software components.

The TSA results are visualized by using graphical user interface (GUI) that is achieved by associating DBCAD and TeeChart Pro components. DBCAD is an advantageous component in computer aided drawing (CAD). DBCAD is used to draw the one line diagram of the power system. TeeChart Pro is a charting component, which has several chart types available in 2D and 3D versions. This component allows the creation of general purpose windows application which is very easy to use, flexible and effective. In the TSA simulation, TeeChart component is reused for plotting the dynamic performances of the TSA results.

B. *Object-Oriented Power System Model (OO-PSM)*

The OO-PSM class hierarchy of the TSA is shown in Fig. 2. The OO-PSM classes that are used in the LFA are similar to the TSA. The detailed explanation of the LFA classes is reported in the references [16]. Thus, the OO-PSM classes that are developed in the LFA are reused as base classes in the TSA. From the base classes, any other classes required in the TSA analysis can be extended. The extended classes are synchronous machines, exciters, governors, turbines, PSS, and SVC system.

The synchronous machines are modeled from classical to complete models. The classical model is encapsulated into cMachine00 class and is designed as the base class of all machine classes. The machine model that represents the field circuit with one damper in the q-axis is cMachine10 class. The machine model that considers two damper circuits in d- and q-axes is named as cMacine11 class. The cMacine11 class is inherited from cMachine10 class. As for cMachine20 class, it represents the field circuit incorporated with two dampers in the q-axis, whereas the machine models that include one and two dampers in the d-axis are encapsulated into cMachine21 and cMachine22 classes respectively. The cMachine22 class has two ancestors, namely cMachine11 and cMachine21 classes which are derived from the same class i.e. cMachine10 class. This design can be realized by using a virtual class implementation. The virtual class is a base class that is passed to more than one derived class through a concept called multiple inheritances. The code fragment implementations of these classes are provided as follows:

```
class cMachine11:virtual public cMachine10 {}
class cMachine20:virtual public cMachine10 {}
class cMachine22: public cMachine21,
                  public cMachine11{}
```

The multiple inheritances are employed to enhance the reusability in the OOD perspective. The cPss class is constructed to represent the power system stabilizer system. The static VAR compensation system is represented as cSVC

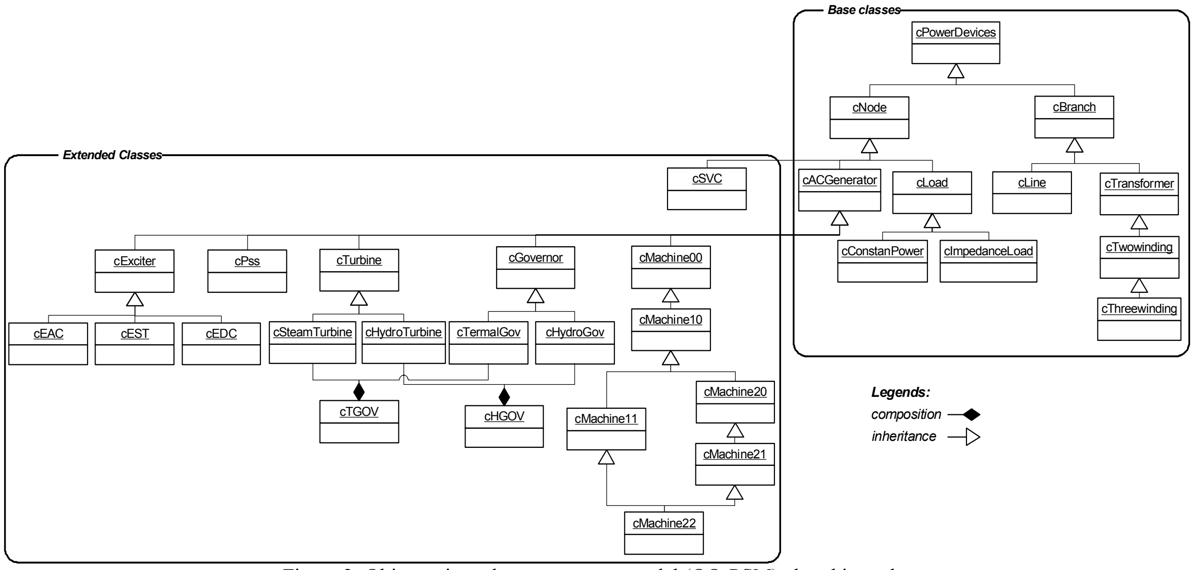


Figure 2. Object oriented power system model (OO-PSM) class hierarchy

class. The excitation system can be derived as three classes: cEAC class for AC excitation system, cEST class for ST excitation system, and cEDC class for DC excitation system. The governor system classes are developed through a composition of classes. The cHGOV class of hydro governor system is composed of two classes, namely cHydroGov class that corresponds to the speed governor of hydro system and cHydroTurbine class corresponding to the hydro turbine. The cSteamTurbine class represents the steam turbine, whereas the cThermalGov class is for the speed governor and speed control of thermal system. Both classes, cSteamTurbine and cThermalGov are built in cTGOV class that represents the thermal governor system.

Since the object devices have been defined in the OOD class hierarchy, it can be extended without changing the existing classes. To enhance reusability, not only inheritance mechanism is applied in the design, but also compositional approach and multiple inheritances have been utilized to increase the reusability technique in the OOD.

C. *Interfaces Design*

The software component has three main characteristics. They are interface, implementation, and deployment characteristics. The interface is used to allow the user or other components to access and interact with it. The interface is a set of services which is naturally grouped into meaningful clusters [15]. The interfaces of the TSA components are depicted in Fig. 3. The power system devices data required are transferred by Add_devices interface. All devices data and properties can be passed using this interface such as Add_generator used to supply the properties of generator, Add_exciter used to pass the exciter data and etc. The devices data can also be accessed by Read_device interfaces.

Since LFA uses instance objects of OO-PSM and this analysis must be performed as a pre-requisite for TSA, the OO-PSM that already exists in LFA components can be passed to the TSA component by Get_Object and Set_Object interfaces. The Get_Object interface is used to evoke the object from the LFA components, whereas the Set_Object interface is to put the object in the TSA components. To set fault parameters such as bus fault, type of fault and amount of the impedance fault, Set_parameter and Set_fault interfaces are employed. The time of study and capturing of the channel considered can be employed by using the Set_time_study and Capture_channel interfaces. After the parameters and OO-PSM are realized, Calculate interface is then performed to execute the solver components. The result can be saved by Save_output interface.

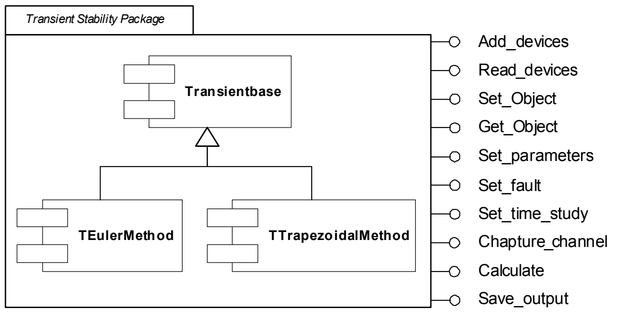


Figure 3. Interfaces for transient stability components

Beside the availability of interfaces, the software application also needs implementations. The implementation is the code that makes the components work. A component may be built with more than one implementation. The deployment of the component is the physical executable files (exe and dll) and the package files (bpl, bpi, and lib).

D. *Function Design*

Usually the controllers of a synchronous machine consist of composition of base block diagram such as voltage regulator, lead-lag block, integrator with non-windup or with windup, and etc. The block diagram corresponds to a physical model has particular functions. Hence, software design and implementation based on base block diagrams is used to enhance the reusability of the code so that it can be reused for other controllers. Fig. 4 shows an example to create an exciter ST1 IEEE model code. There are two states: state 0 and state 1 that correspond to the initial and the final estimates based on equations (3) and (5) respectively, for Modified Euler method. Both initial and final estimates are encapsulated into a method that is easy to debug and maintain. To execute either initial or final states, the input must be passed to the method. That can be shown in Fig. 5 for regulator voltage non-windup codes. In the initial state, the voltage regulator quantity (*Vrn*) and its differential (*dVr/dt*) are calculated. Both variables are stored to obtain the final value (*Vrn1*) in the final state. Then the regulator voltage is updated with a new value that is evaluated the limit. This approach is also implemented in the Trapezoidal method. Since all controllers of machine are composed of the base block diagram, the method can be reused for other controllers. These methods are included in the software components.

```
void TEulerMethod ::ExcEstimateST 1(int state ,...)
{
for (i=1; i<=EST1->NDevice ; i++) {
    j = Exc [i].MacHandle ;
    if (state == 0){
      vcn [j] = TermVoltTransduser (…);
      Vsum =  Vref -EST1[i].Vc+EST1[i].vso-EST1[i].Vf;
      Vrn [j] = RegVoltNonWindup (…);
      Vfn [j] = ExcitationSystemStabilizer (…);
    }
    else if (state == 1){
      EST 1[i].Vc = TerminalVoltageTransduser (…);
      Vsum = EST1[i].Vref -vcn[j]+sm[j].vson -Vfn [j];
      EST 1[i].VR  = RegVoltNonWindup (…);
      EST 1[i].efd = EST1[i].VR;
      EST 1[i].efd = NonWindup (efd,Efdmin ,Efdmax );
      EST 1[i].Vf  = ExcitationSystemStabilizer (…);}}
}
```

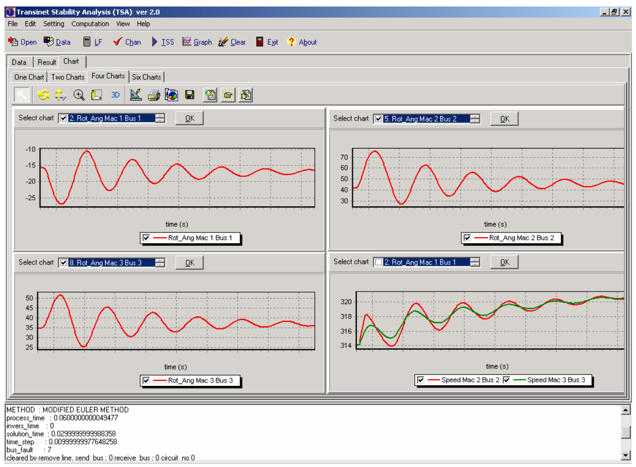
Figure 4. Exciter ST1 IEEE model representation

```
float cExciter ::RegVoltNonWindup (int state ,…)
{
  dvrdt = *dVrdt;
  if (state == 0){                                 ⎫
      dvrdt  = (input * Ka - Vr)/Ta;               ⎪
      Vrn 1 = Vr + dvrdt * dt;                     ⎬Initial state
      Evaluate _limit ();                          ⎪
      *dVrdt  = dvrdt ;                            ⎭
   }
   else if (state == 1){                           ⎫
      dvrdtn  = (input * Ka - Vrn)/Ta;             ⎪
      Vr = Vr + 0.5 * (dvrdt + dvrdtn ) * dt;      ⎬Final state
      Evaluate _limit ();                          ⎪
      Vrn 1 = Vr;                                  ⎪
   }                                               ⎪
return Vrn 1;                                      ⎭
}
```

Figure 5. Voltage regulator non-windup representation

V. COMPONENT INTEGRATION

The LFA and TSA software components are integrated to develop a component-based application. The components used in the TSA application are built via in-house development, only DBCad and TeeChart Pro components selected as the graphical user interface are acquired from the third party. The components are integrated through some well-defined interfaces. These interfaces provide the bond that forms a system from the disparate components. All the components that are built through in-house development are created by Borland C++ Builder which strongly supports the Integrated Development Environment (IDE) tools. C++ Builder is chosen as the IDE tool because it offers a wide range of component platforms, such as Visual Component Library (VCL), and Component Object Model (COM). The components are gathered to form the TSA application.



Transinet Stability Analysis (TSA) ver 2.0
File Edit Setting Computation View Help
Open Data LF Chan TSS Graph Clear Exit About
Data Result Chart
One Chart Two Charts Four Charts Six Charts
Select chart 2. Rot_Ang Mac 1 Bus 1 OK
Select chart 5. Rot_Ang Mac 2 Bus 2 OK
Select chart 8. Rot_Ang Mac 3 Bus 3 OK
Select chart 2. Rot_Ang Mac 1 Bus 1 OK
time (s)
Rot_Ang Mac 1 Bus 1
Rot_Ang Mac 2 Bus 2
Rot_Ang Mac 3 Bus 3
Speed Mac 2 Bus 2
Speed Mac 3 Bus 3

```
METHOD  : MODIFIED EULER METHOD
process_time   : 0.0600000000049477
invers_time    : 0
solution_time  : 0.0299999999988358
time_step      : 0.00999999977648258
bus_fault      : 7
cleared by remove line. send  bus : 0 receive  bus : 0 circuit  no 0
```

Figure 6. User friendly TSA component based application

The LFA components are reused with the TSA components, the exchange data is transferred between components based on the interface SetObject and GetObject. Firstly, the power system data is transferred to the LFA components by the interface SetObject and after the LFA solution, the data is brought by the interface GetObject. As an example, the LFA components and TSA components have an object called Bus of the type TNode. The interface Set_Bus passes a copy of the object Bus defined in TSA and LFA components. After the solution, Get_Bus method delivers a copy of the object Bus from the LFA component to the object Bus in the TSA components. For other OO-PSM objects, similar approach can be applied. This means that TSA components take the solution from LFA components encapsulated in OO-PSM. The TSA analysis components are also integrated with both the third party components that include DBCAD and TeeChart components. The user of the application can execute the transient stability analysis with many facilities for results visualization such as charts and tables. Fig. 6 shows a window for the CBSE transient stability application.

VI. COMPONENTS REUSABILITY

Reusability is an important issue in software engineering. In developing the TSA application, reusability can be categorized into two types: OO-PSM reused and components reused. The OO-PSM object reused is established by using inheritance, composition and a combination both of inheritance and composition. The percentage of reusability of the OO-PSM is exhibited in Table I. The percentage of reusability by composition is equal to 40%. For new requirement of the TSA, new classes are extended from base classes in the LFA. These can be developed by inheritance approach that indicates 50% in percentage, whereas the last object reused is built from both inheritance and composition mechanism that is equal to 10% in percentage. These are realized for developing governor system objects that are composed of speed governor and turbine system.

TABLE I. PERCENTAGE OF THE OO-PSM REUSABILITY

Reusability	**OO-PSM**	**Percentage (%)**
Inheritance	10	50
Composition	8	40
Inheritance and composition	2	10
Total	20	100

TABLE II. COMPARISON OF THE REUSABILITY OF COMPONENTs

Software Components	**Reused**	**Scratch**
Load Flow	2	0
Transient Stability	0	2
Input Data	1	1
Linear Solver	1	0
Output Interface	0	1
GUI	2	0
Total	6	4
Percentage (%)	60	40

Statistics for the component reusability in the TSA simulation using CBSE is given in Table II. The table shows that the reused components consist of a total of six components, whereas the components developing from scratch are four in total. Obviously, the result shows that reusability is one of the most important advantages when the CBSE and OOD methodology are implemented in TSA software. The reusability percentage is 60% for components that have been developed and used in LFA and still reused in the TSA.

VII. TEST SYSTEMS AND NUMERICAL RESULTS

The TSA component-based software was tested by the IEEE data test, such as the 9 buses-3 machines, 39 buses-10 machines, and 162 buses-17 machines 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 CBSE software application and conventional structural programming application is made. The execution time as a main comparison is accomplished through the capture of dynamical performances of TSA 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 III shows the comparison results. The differences of OOD and CBSE implementation and structural programming 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 CBSE application consumes more time as compared with non CBSE, because of the fact that CBSE application needs to call library files which are separated from the application files. On the other hand, for non-component application, function calling takes place in the same memory block of the application, which makes the execution time less than that of CBSE application. The comparison is also made to compare the response of machine when different time step is carried out in the study. The field voltage responses are captured with time step equal to 0.001, 0.01 and 0.05 s. The response is accurate when a small time step such as 0.01 and 0.001 s are applied, while inaccurate response will occur when a larger time step such as 0.05 s in the Modified Euler method. The Modified Euler is an explicit numerical integration that allows the size of time step to be compatible with the bandwidth of the equipment model in order to avoid numerical instability. Typically, the time step is one-half of the smallest of time constant in equipment model, usually the sub-transient rotor time constant [17].

TABLE III. EXECUTION TIME COMPARISON (IN SECOND)

Data Test	Method	OOD & CBSE	Structural	Diff (%)
9 bus, 3 machines	M. Euler	0.0200	0.0190	5.2632
	Trapezoid	0.0200	0.0190	5.2632
39 bus, 10 machines	M. Euler	0.0640	0.0630	1.5873
	Trapezoid	0.0650	0.0640	1.5625
162 bus, 17 machines	M. Euler	0.2330	0.2120	9.9057
	Trapezoid	0.2240	0.2050	9.2683

In the Trapezoidal method, the responses are stable with all time step of study, although the time step is large. The Trapezoidal method as the implicit integration method eliminates the small time constant numerical instability problem. This property can be used to speed up simulation by using larger time step. However, there is a limit to the size of the time step beyond which the fidelity of high frequency response is lost. Besides, with a large time step, the number of iterations increases and this may offset the advantage that a large time step offers.

VIII. CONCLUSION

The paper has presented the application of CBSE methodology for developing a TSA program. The architecture and development design in the CBSE have been proposed. The power system devices that are involved in the TSA are modeled as entity objects (OO-PSM) such that the OO-PSM is independent from the solution algorithms. The solvers are built separately to enhance reusability and maintainability of software. The solvers are encapsulated into independent software components. The components are reused to develop user friendly transient stability application. The reusability results show that a 60% of the software is reused from a component based load flow software. Although the execution time of CBSE is slightly more than non CBSE, it will not be a major problem since the difference is small. Different time steps are also employed to investigate the response characteristics of both methods; Trapezoidal and Modified Euler. The responses are stable with all time step of study, although the time step is large in Trapezoidal method, whereas for Modified Euler method, unstable responses are observed when a large time step such as 0.05 second is implemented.

REFERENCES

[1] I. Crnkovic, M. Larsson, "Challenges of component-based development", Journal of Systems and Software, Vol. 61, No. 3, April 2002, pp. 201-212.

[2] E.Z. Zhou, "Object-oriented Programming, C++ and Power System Simulation", IEEE Transactions on Power Systems, Vol. 12, No. 1, Feb. 1996, pp. 206-215

[3] S. Pandit, S.A. Soman, S.A. Khaparde, "Object oriented design for power system applications", ISSN 0895-0156 ©2000 IEEE, Oct. 2000, pp. 43-47

[4] A. Manzoni, S. E. Silva, I.C. Decker, "Power systems dynamics simulation using object-oriented programming", IEEE Transactions on Power Systems, Vol. 14, No. 1, Feb.1999, pp. 249-255

[5] D.L. Lubkeman, A. Ghosh, J. Zhu, "Object-oriented programming for power engineering education", 0094-2898 IEEE, 1993, pp. 69-73

[6] S.N. Ironmonger, M.J. Bushnell, R. Patel, M.E. Bradley, B.W. Vaughan, "An object-oriented power system model and graphical information display system for control engineers", Fourth International Conference on Power System Control and Management, (Conf. Publ. No. 421), 16-18 April 1996, pp. 120 - 124

[7] Kundur, P.; et al. "Definition and classification of power system stability", IEEE Transactions on Power Systems, Vol. 19, No. 3, Aug. 2004, pp.1387-1401

[8] J Arrilaga, C.P. Arnord, *Computer Analysis of Power System*, New York: John Wiley & Sons, 1990

[9] Kundur P., Dandeno P.L., "Implementation of advanced generator models into power system stability programs", IEEE Transaction on Power Apparatus and System, Vol. PAS-102, No. 7, July 1983, pp. 2047-2063.

[10] IEEE Task Force, "Standard load models for power flow and dynamic performance simulation", IEEE Transactions on Power Systems, Vol. 10, No. 3, Aug. 1995, pp. 1302 – 1313

[11] IEEE Standard Board, "IEEE recommended practice for excitation system models for power system studies", IEEE Std 421.5, 1992

[12] IEEE Committee Report, "Dynamic models for steam and hydro turbines in power system studies", Vol. PAS-92(6), 1973, pp. 1904-1915

[13] IEEE Special Stability Controls Working Groups, "Static VAR compensator models for power flow and dynamic performance simulation", IEEE Transactions on Power Systems, Vol. 9, No. 1, Feb. 1994, pp. 229 – 240

[14] L. Bass, P. Clements, R. Kazman, *Software Architecture In Practice*, New York: Addison Wesley, 1998

[15] A.W. Brown, K. Short, "On components and objects: the foundations of component-based development", Proceedings Fifth International Symposium on Assessment of Software Tools and Technologies, June 1997, pp. 112 -121

[16] K.M. Nor, H. Mokhlis, T.A. Gani, "Reusability techniques in load-flow analysis computer program", IEEE Transactions on Power Systems, Vol. 19, No. 4 , Nov. 2004, pp. 1754-1762

[17] F.P. de Mello, F.P. Feltes, J.W. Laskowski, T.F Oppel, "Simulating fast and slow dynamic effects in power systems", IEEE Computer Applications in Power System, Vol. 5, No. 3, July 1992, pp. 33-38