Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2017, 4(12): 932-940

Research Article ISSN: 2394 - 658X

Elena – A Novel Component Based Life Cycle Model

Sandeep Chopra[1], MK Sharma[2] and Lata Nautiyal[3]

[1]Department of Master of Computer Application, SGRR University, Dehradun, India
[2] Department of Master of Computer Application, Amarapali Institute, Haldwani, India
[3] Department of Master of Computer Application, Graphic Era University, Dehradun, India
tosandeepchopra2016@gmail.com

ABSTRACT

As day by day the digital world is spreading all over the world, user is dependent more and more on information technology. From household activities to the big scientific activities, everything is impossible without the use of information technology. Although the enormous growth shown by the Information Technology world also faces large number of challenges and issues every day. To solve the challenges of software development, which start from developing the software from the scratch, the component based software engineering (CBSE) has become the solution in the form of ready-made package. One of the inspiration to adopting CBSE as software development paradigm is the easily available software within less time frame and the lesser cost. Here the objective of this paper is to present an innovative software life cycle model for CBSE and to fulfill the need of IT community by providing the ready to use software system to end user within the limited time frame and in limited budget.

Keywords: Component-Based Development (CBD), Software Development Life Cycle, Elena Life Cycle Model, COTS, Static requirement, Dynamic requirement

INTRODUCTION

From last two decades Component-Based Development (CBD) has established itself as a strong pillar in the IT industry. It is a compact artifact with features to fulfill the IT needs of various stakeholders. The Component-based applications are distinct from the assembling components [1-3]. Although it came in light in the starting of 1990's but in the early 2000 it has become the necessity of large number of business organization like IBM [4-5]. IT community was looking for some inbuilt things and solution was in front of them in the form of CBSE. Component based technology (CBT) has many advantages like good quality product in less time, minimum cost and to develop a system which is easy to install. With the change in time the component based engineering has also attained certain standards and its growth has also been enhanced in embedded system [6-7]. CBD has come into force by using the already existing products that ensures reusability and provides good software at low cost which is ever lasting. This component based software system can even be used in constructing a new model for development *i.e.* in large and complicated systems [8-9].

STATE OF ART

There are large numbers of CBSE models available in IT industry. The most popular one are presented in the following literature review. To express the activities of software developments, which start from requirement analysis till the maintenance, we need a diagrammatic form of sequence pattern and that sequence of events is regarded as SDLC. With the increase in time, new inventions are going on in the development of software industry and with the help of this one can easily select, modify or customize the software components [10-11]. The first state of art presented is Twin Peaks model [12] which expresses the continuous development of requirements and architecture throughout the development. It shows the partial and easy way to develop the software product.

In X Model, one of the oldest models start from requirement analysis. The important attributes of this software process is reusability. With the help of reusability and testable modules one can develop a gigantic software system in a very short span of time [13]. In Y Software Life Cycle Model(SLCM) the main focus is on reusing the software module during component based software development. This model examines the step by step process of the software development process which is overlapped during other steps of the development process. The important phase is the designing which includes: domain engineering, frame working, assembly, archiving, system analysis, design, implementation, testing, deployment and maintenance [14]. Umbrella model depends on three stages i) Design stage ii) Integration stage iii) Run-time stage. For designing stage components are picked from the pool of library and then the module is constructed. In integration stage components are joined together with common interfaces. In run time stages the component binary is initialized in the running system. Testing and verification are implemented at every stage of umbrella model [15].

In COTS model, component can be developed by many software engineer using different programming languages and platforms which is suitable for the project. Then this module is assembled in a properly defined software architecture. The prime important factor in COTS based approach is that we can change the traditional software development life cycle model [16]. The V model consists of V shape sequence of steps [17]. It consists of many phases and provides the details information at the design phases. The main emphasis of V-Development is component development lifecycle. Component development lifecycle was considered as different process. The selection phase gets input from the separate system that usually finds and evaluates the suitable components to be composed into the system. The V Model is an adaptation of the rigid traditional waterfall model for modular system development with little flexibility. The combination of two V's creates W model [18]. This model comprises two important stages, *i.e.* component design and component deployment. Modules are picked and connected according to the domain specification and then it is put into the component pool of library. It supports verification and validation at all stage i.e component level and system level but it does not show risk analysis.

In knot model, every step focuses on reusability which is based on the two factors i.e risk analysis and feedback. This model contains three stages and is mainly useful for complex systems. This model identifies risk factor at early stages. This will immensely reduce the cost and time of making a software. Feedback is helpful to improve the performance of a component [19]. The Elite Life Cycle Model (ELCM) is an innovative software lifecycle model which presents the risk analysis [20]. This state of art provides a growth in the field of component-based life-cycle models.

'ELENA' – A NOVEL CBSE MODEL

Component-based software engineering is based on the concept to generate a large system by picking suitable components or modules and then integrate all the components by providing a common interface. The proposed Elena dynamic Model for Component-Based Development is a growing life cycle model for enhancement of the computer programs by just adding the concept of dynamic requirements according to the new trend. Reusability is the key and a very important attribute of component based software engineering which means developing the component once that can be used in many different domains. To develop a module of a computer system a systematic approach will be required. This Elena model supports the programmer to make a choice of the most suitable software development process. This model focuses on the requirement gathered from different resources and then changes according to the new market trend which will satisfy the end user or customer. Following are the steps of this model which is shown below.

Requirement Analysis

Collecting the requirement and understanding the expectations of the customer where the consumer must be the first step of the development of any system because it's a vital and difficult phase of any life cycle model. If the real life problems and the requirements are not understood in advance, then all work done will go in vain. In fact, requirements specification tells the developer what is to be built so that the end user can use it in a proper way. In order to collect information, there are certain strategies which are discussed below.

a) Interview: This approach is based on the interaction between the two persons which is used for collecting the data. There are two persons involved in this method and the interview is being conducted. In this process one person asks the questions 'interviewer' and the other person gives the response 'interviewed'. It may be further divided into two parts *i.e.* formal and informal and the problem asked may be pre decided questions or it may be undecided questions. While organizing an interview, one should keep the following things in mind.

- **i)** Whom to interview?
- **ii)** What to ask?
- **iii)** How to start the interview?
- **iv)** How to conclude the interview?

b) Surveys, Questionnaire and Data Mining: In this method, various questions regarding the development process are given to different end users and their responses are collected. It's a beneficial and trust worthy method because with the help of this technique one can get the real facts and data. The best thing about this technique is that one can get many different responses from different end users. By using the concept of data mining, one can draw useful charts, diagrams which will help to know the exact requirement of the client.

c) Observation and Past experience: Observation is the process of recognizing and noticing people's objects and occurrences and to obtain the required information. In an organization working in the similar or same domain might help to provide past experiences or even documented case studies. This helps to give a clear picture of the requirement, which may otherwise be left hidden. The system developer uses all the permutation and combination to examine all the methods, as one method is not enough for gathering the requirement of the system.

Requirement Updating as Per the New Market Trend

After taking all the necessary inputs from the customer, the software designer will go for market analysis and update the requirement as per the market trend using different new technologies. Market trend reveals how the product may change over the coming years. Analysis of similar rival products, their features, current and future market trends give indication about the potential of the product.

Customer Review

After updating the customer's requirement from the current market trend, the review from the user is taken. If the customer is not satisfied with the above changes, the further requirement from the customer is taken and this process continues till the customer is satisfied, If the customer is satisfied we can further proceed by categorizing the requirement into two parts.

Static Requirement: It does not change over the period of time; it will remain fix and contains all the necessary features and functions that are required to build a software product. for example

a) Suppose a software company is involved in making a game or the project is of making the yearly calendar etc, this type of software requires a static requirement because once the software is developed either of the game or a calendar, it will be considered as permanent and no changes can be done in future.
b) If the written assurance from the customer side is given that he will not update his/her requirement in future, then it will be considered as static requirement.

Dynamic Requirement: This type of requirement changes over the period of time. It is more flexible and can be rectified or modified by the experts of the industry. It contains all the required features to build a software product which is based on the latest technology and can be updated with the market trends. Once analysed that it's a dynamic requirement, process of developing a software product can start and it can change in every step or phase while moving ahead. For example - Suppose the university is running certain courses but over the period of time it might add or delete few courses as per the industry's demand. So this type of requirement is considered as dynamic where changes can be done at any time in the near future.

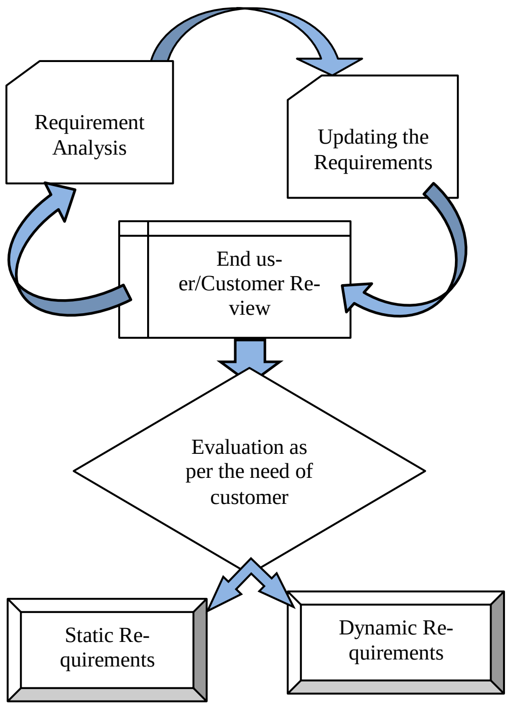

Fig. 1 Steps involved in requirement analysis

SRS: The requirements are systematically documented for future reference using certain specification formats or it can be said that it's just a written agreement between a client and a software company. After the confirmation of the software project from the client, the next step is to design the schema, the software architecture. It has been recently indulging in the software industry after understanding its importance. Software architecture is just a out-line structure *i.e.* how a software product is made. The prime concern of designing software architecture is to identify the components and their communication interfaces, combining with other components and to make a gigantic software system. Once the designing of software architecture is over, the design should be verified whether it will meet requirements of the customer correctly or not. There are some characteristics of a good SRS which is listed below
i) Correct ii) Consistent iii) Complete iv) Unambiguous v) Verifiable vi) Traceable vii) Modifiable

Selection of the Component
Component can be defined as a collection of interfaces and functional properties. It can be mathematically expressed as
$C = [\ I\ , Fp\ ]$ where C denotes component, I denotes interfaces i.e $I = \{ I_1, I_2, I_3, I_4, \ldots\ldots I_m\}$ and Fp denotes functional properties *i.e.* $Fp = \{ Fp_1, Fp_2, Fp_3, Fp_4 \ldots\ldots\ldots\ldots Fp_m\}$
A component is a collection of more than one sub-system. The main objective is to develop a software system according to the customer requirement. To achieve this a designer can use these pre-existing components. Some of the components are common in most of the software products and they are called 'patterns'. These patterns are very helpful in searching the components in component library. Selection of components can be done by number of ways.
1) Repository 2) Update or Modify 3) New 4) Outsource

Repository: Repository is a collection of reusable components. For selecting a component from the repository *i.e.* already existing component, a developer need to know the functional requirement of the software product. Component is chosen from the repository that fulfills the need to build the software product.

Modify: If the component from the repository does not meet the specific requirement and if the developer thinks that certain changes in a particular component can help the desired objective than modification should be made.

New: If the component does not meet the specific requirement and if the developer thinks that certain changes in a particular component cannot help to fulfill the desired objective then a new component can be designed in such a manner which can be reused in future.

Outsource: If the component does not meet the specific requirement and if the developer thinks that it can be developed outside the organization so that it becomes beneficial for the software product, then the component is built outside the organization and list into the component directory for the future project.

COMPONENT INTEGRATION

After selecting all the software components either a pre-existed component or a newly developed component, the components are gathered in a common pool. Now this is the time to combine or integrate all these component to build a customer oriented software system. To combine these different components, interfaces are required. Interfaces are used to check the functioning of these integrated components *i.e.* whether output is produced according to the desired data on the basis of input. But it is not compulsory that after combining all the tested components the desired result is achieved as there are many different types of errors which occur. for example, data type mismatch, number of arguments mismatch etc. So there is a requirement to combine these components at many level of integration. Now there is a need to test these integrated components as a complete single unit and to generate the corresponding test report which can be used during system validation and maintenance activities.

COMPONENT TESTING

Testing the component is the vital part of this step. The main objective of testing is to find an error. The component which is required to build the project may not be necessary available in the domain of library. So at that moment the developer needs to write a code for the unmatched components. Sometime few changes in the component will make the component ready and at the other time only a little change in the interface will make the component ready for the project. So the tester needs to take a decision whether changes in the component are required or to develop a new component from the scratch. These new component should be developed according to the new standard coding guidelines. These new component should be tested thoroughly because it is used number of times in different-different systems. The newly developed component should be included in the library of component for the future use. If the result is not up to the mark, it indicates that the selection of component is wrong and the designer has to go back again to3.5 and if the result of the testing is correct then move to the next step i.e validating the product. Before moving to the next step, the verification is to be carried out.

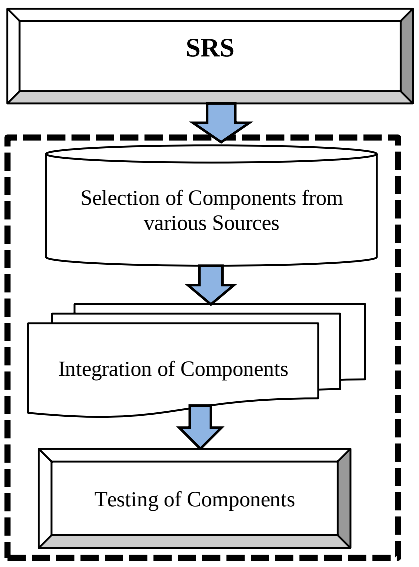
SRS
Selection of Components from various Sources
Integration of Components
Testing of Components

Fig. 2 Traditional CBSE methodology of Selection- Integration and Testing

CUSTOMER VALIDATION

Software validation is used to ensure that the software meets the expectations of the customer. It ensures that the developer has created an appropriate product, as per the expectations of the consumer and customer. It must be noted that the verification is based on the specifications whereas the validation is purely based on the customer's expectations and satisfactions. The validation may over rule the mistakes of requirement specifications and may fail even if the product is verified and meets the requirement specifications themselves may have been specified wrongly. Component selection, integration and testing phases are common for both static software product and dynamic software product, but the process of validation is different in both the cases. It is based on the category which we have discussed above.

Static Software Product Validation

In a static product, requirements are fixed and should be known to the customer before development program starts. It cannot be changed because the customer was satisfied during the review process. In this phase customer will check the performance of the product which was written or documented at initial phase. If the end result of the product is considered satisfactory by the customer, it will be accepted and delivered. Static software product is specially designed for a specific user or customer so the changes or upgrading the product is generally not required.

Dynamic Software Product Validation

The basic logic hidden behind a dynamic product is that requirement may get change in the near future. So before launching the product or handling it over to the customer, validation of the product within the organization by different programmers or system designers is required. When the product gets validated by different programmers or system designers, now this is the right time to call the customer or actual user and to validate the product in a company site as well as client site. If the user or customer is not satisfied, then developer has to go back to section (dynamic requirement) and update the requirement as per the market trend. If the customer is satisfied, then create a smaller version of that software product and release it into the market and collect the feedback. Considering the feedback of the product company has to decide whether changes are required or not. If the feedback is up to the mark or expectations, then the product should be launched into the market. If the feedback is not up to the mark, then change the dynamic requirement document by taking the consent from the customer. This model suggests that this process should be done on a regular basis if the customer wants changes in its requirement.

A COMPARATIVE ANALYSIS OF AVAILABLE COMPONENT-BASED MODEL WITH THE PROPOSED MODEL

The main objective of component based model is to develop a quality software product with in stipulated time and cost. The proposed model provides an approach to identify or to categorize the user's requirement into two parts *i.e.* static or dynamic at an initial phase of the process model after taking the customer review. As we know static requirement does not change over the period of time but dynamic requirement can change. So when the developer knows about the type of requirement whether static or dynamic in the starting phase of the model, the developer sets his mind accordingly as there is less effort, skill and cost required in a static software product development. Whereas in a dynamic software product an extra effort, mental calibre or skill and cost are required as the requirement of the customer can be changed after launching the product in the future so the developer have to design the software product in such a way that it meets the customer requirement.

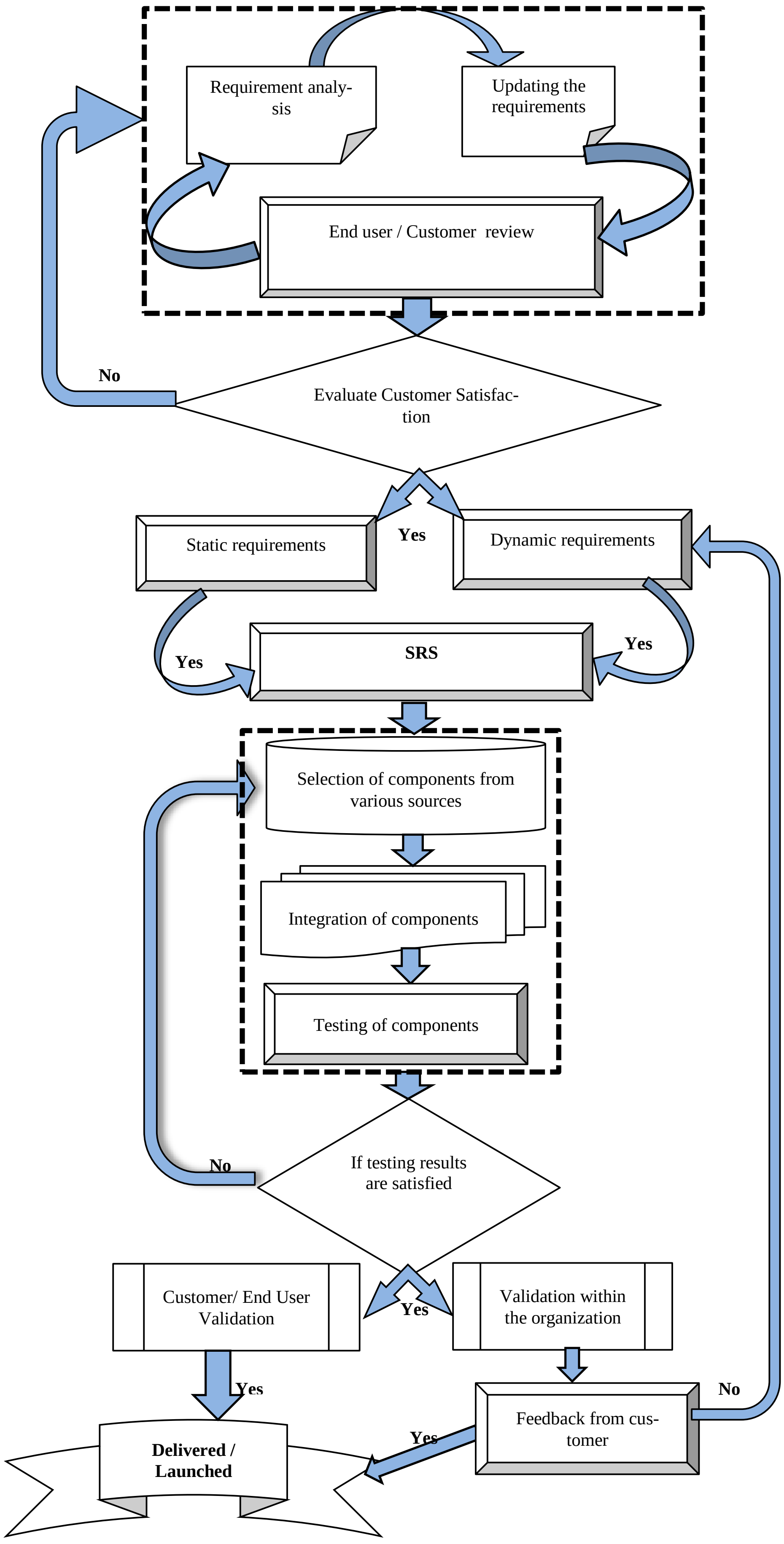
Requirement analysis
Updating the requirements
End user / Customer review
No
Evaluate Customer Satisfaction
Yes
Static requirements
Dynamic requirements
Yes
SRS
Yes
Selection of components from various sources
Integration of components
Testing of components
No
If testing results are satisfied
Customer/ End User Validation
Yes
Validation within the organization
Yes
No
Feedback from customer
Yes
Delivered / Launched

Fig. 3 Proposed Elena Model for CBSE

The proposed Elena model is built on the concept of reusability *i.e.* selection of pre-existing components and adds an understandable integration and test the entire process. This model proposes two step validations, one for static product and the other for dynamic product. In a dynamic product, Validation takes place in two steps. Firstly, it is validated by different programmers or designers, within the organization and secondly it is validated by actual user in the company site and at the client site. Then a small version of the product is launched in the market. After that the feedback of the product is taken and considering the feedback if changes are required in the product it might change but if it is considered all right or up to the mark, the product is launched in the market. As it is a dynamic product and if there is any requirement of the customer in the near future, the product will change according to the customer's need and this process continues.

This Elena model is an appropriate choice for development of various software products like small, medium and large size. A brief comparison of proposed model with the existing models is depicted through Table -1.

Table 1 – Comparative Analysis of Existing CBSE Models

Events ↓ Model →	V	W	X	Y	Umbrella	Knot	COTS	Elite	Elena Dynamic
Analysis of domain	T	T	T	T	T	T	T	T	T
Updating Domain as new market trend	F	F	F	F	F	F	F	F	T
Requirement categorization	F	F	F	F	F	F	F	F	T
Component Searching	F	T	T	F	F	F	T	F	T
Evaluation of Component	T	T	T	T	T	T	T	F	T
Component selection	T	T	T	T	T	T	T	T	T
Analysis of risk factors	F	F	F	F	F	T	F	F	T
Risk analysis of integrated components	F	F	F	F	F	F	F	F	F
Component certification	T	F	F	F	F	F	F	F	F
Component testing	T	T	T	T	T	T	T	F	T
Integrated CBSE software testing	F	F	F	F	F	F	F	T	F
Reliability of components	T	T	T	T	T	T	T	T	T
User customer involvement	F	F	T	F	F	F	F	T	T
Reusability	F	T	T	T	F	T	T	F	T
Recent or dynamic requirement changes	F	F	F	F	F	F	F	F	T

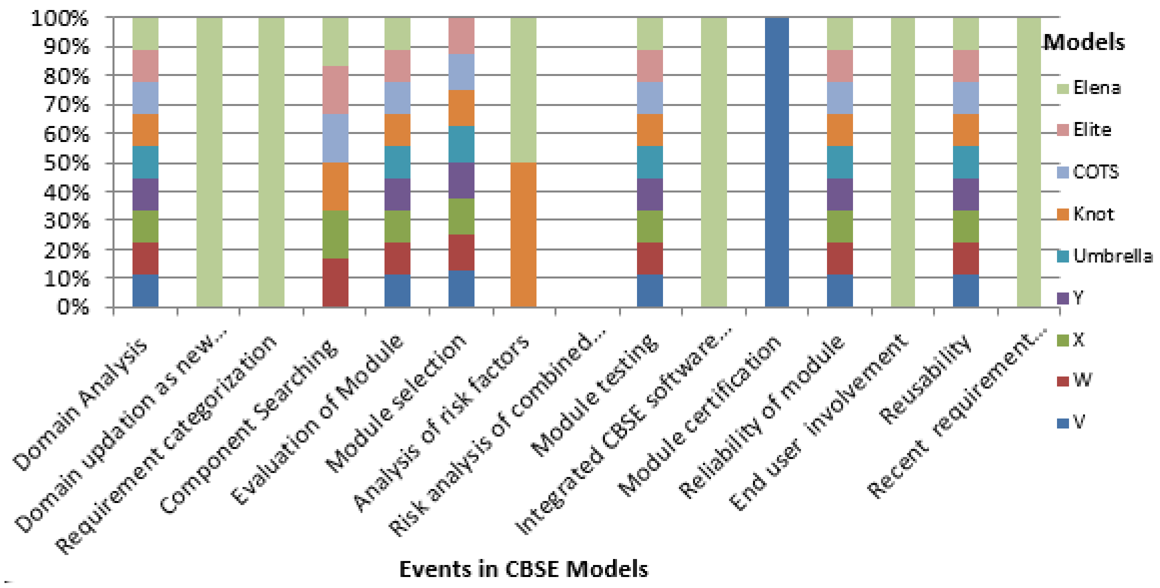

Fig. 4

Number of events found in V model = 6	Number of events found in W model = 7
Number of events found in X model = 8	Number of events found in Y model = 6
Number of events found in Umbrella model = 5	Number of events found in knot model = 7
Number of events found in COTS model = 7	Number of events found in Elite model = 5
Number of events found in Elena Dynamic model = 12	

Various events were found while studying different CBSE models and created a chart. It is concluded that these CBSE models have fifteen events. Fig. 5 shows the capability of various CBSE models. By using the formula:

Percentage of events = (Number of events in a model) X 100 / (Total number of events)

For Model V =	40%	For Model W =	46.66%
For Model X =	53.33%	For Model Y =	40%
For Model Umbrella =	33.33%	For Model knot =	46.66%
For Model COTS =	46.66%	For Model Elite =	33.33%
For Model Elena =	80%		

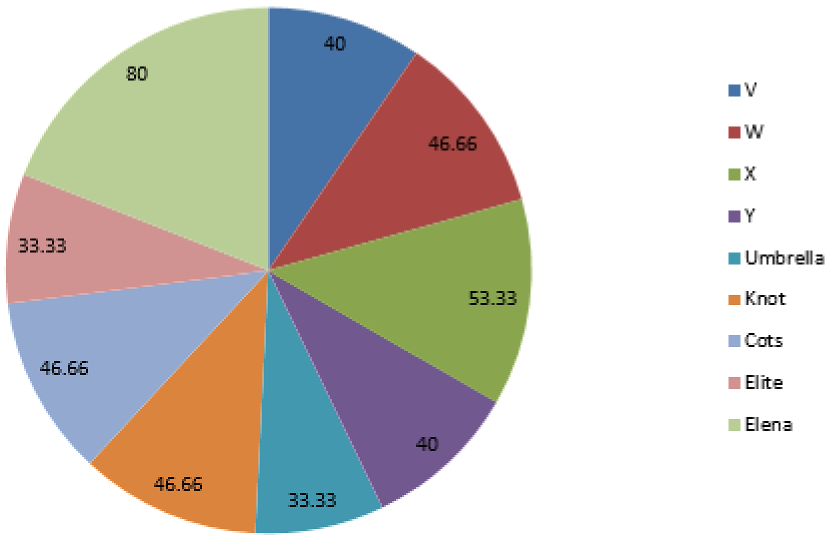

Fig. 5 Percentage of events in different CBSE Models using Pie Chart

CONCLUSION

The prime objective of component based software development is to provide good software product with in the restricted time and budget. Selecting the right logical development process is one of the key factors for generating the desirable software. This paper proposes a new approach that after the customer review, one can categorize the requirement into two parts i) Static requirement ii) Dynamic requirement. After that we can made SRS. Based on some events we have also compared this new model with other CBSE models and make different chart and diagram, which tells an efficiency of a particular CBSE model.

REFERENCES

[1] C Szyperski, *Component Software, Beyond Object-Oriented Programming*, ACM Press, Addison-Wesley, NJ, **1998.**

[2] GT Heineman and WT Councill, *Component-Based Software Engineering: Putting the Pieces Together*, Addison-Wesley, **2001.**

[3] Bertrand Meyer, The Grand Challenge of Trusted Components, *Proceedings of 25th International Conference on Software Engineering, IEEE Computer Society Press*, **2003**, 660–667.

[4] Malcolm Douglas McIlroy, Mass Produced Software Components, In Peter Naur and Brian Randell (Eds), NATO Science Committee, NATO, Brussels, Belgium, **1969**, 88–98.

[5] Grady Booch, *Software Components with ADA: Structure, Tools and Subsystems*, 3rd Edition, Addison-Wesley, **1993**.

[6] G Myers, *The Art of Software Testing*, Wiley, **1979**.

[7] M Sitaraman and BW Weide, Special Feature Component-Based Software Using RESOLVE, *ACM SIGSOFT Software Engineering Notes*, **1994**, 19 (4), 21-67.

[8] Ivica Crnkovic, Component-Based Software Engineering for Embedded Systems, *Proceedings of the 27th International Conference on Software Engineering*, ACM New York, NY, USA, **2005**.

[9] Ivica Crnkovic, Stig Larsson and Michel Chaudron, Component-Based Development Process and Component Lifecycle, Online Available: http://www.mrtc.mdh.se/publications/0953.pdf.

[10] G Pour, M Griss and J Favaro, Making the Transition to Component-Based Enterprise Software Development: Overcoming the Obstacles – Patterns for Success, *Proceedings of Technology of Object-Oriented Languages and Systems*, **1999**, 419 – 419.
[11] Lata Nautiyal, Umesh Tiwari, Sushil Dimri and Shashidhar G Koolagudi, Component based Software Development- New Era with new Innovation in Software Development, *International Journal of Computer Applications*, **2012**, 51(19), 5-9.
[12] M Sitaraman and BW Weide, Special Feature Component-Based Software Using Resolve, *ACM SIGSOFT Software Engineering Notes*, **1994**, 21-67.
[13] NS Gill and P Tomar, X Model: A New Component- Based Model, *MR International Journal of Engineering and Technology*, **2008**,1(1-2), 1-9.
[14] Luiz Fernando Capretz, Y: A new Component-Based Software Life Cycle Model, *Journals of Computer Science*, **2005**, 1 (1), 76-82.
[15] Anurag Dixit and PC Saxena, Umbrella: A New Component-Based Software Development Model, *International Conference on Computer Engineering and Applications*, IACSIT Press, Singapore, **2011**.
[16] Christine L Braun, A Lifecycle Process for the Effective Reuse of Commercial Off-the-Shelf (COTS) Software, *Proceedings of the Symposium on Software Reusability*, ACM New York, NY, USA,**1999**, 29-36.
[17] Ravi Shanker Yadav, Improvement in the V-Model, *International Journal of Scientific & Engineering Research*, **2012,** 3(2), 1-8.
[18] The W Model for Component-based Software Development, Online Available: http://www.cs.man.ac.uk/~kung-kiu/pub/seaa11b.pdf, **2011**, 47-50
[19] Rajender Singh Chhillar, Parveen Kajla, A New Knot Model for Component Based Software Development, *International Journal of Computer Science*, **2011,** 8 (3), 480-484.
[20] Lata Nautiyal, Umesh Kumar Tiwari, Sushil Chandra Dimri, Shivani Bahuguna, Elite - A New Component based Software Development Model, *International Journal of Computer Technology and Applications,* **2012**, 3(1), 119 – 124.