International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-3, February 2020

Assessment Cost Estimation for Component Based Software: A Tool Based Technique

Rajul Jain

***Abstract:** Assessment cost of the software component contributes a major part in software cost estimation and it is one of the major cost of the software out of- integration cost i.e. the cost of glue codes, assessment costs and tailoring cost. Many researchers have proposed formulas for evaluating assessment and tailoring costs theoretically. Assessment cost is very often considered to be theoretical cost which involves cost of component selection and composition. According to Moguel Goulao et. al assessment cost for overall component can be measured qualitatively and quantitatively both. He has suggested that qualitative measurement is mostly based on views provided by the experts whereas the quantitative measurement is more subjective and repetitive in nature. Various metrics has been suggested by different authors to quantitatively measure the assessment cost for software components. In this work we applying the metrics on case study of UCRS and developed a tool for evaluating the assessment cost which can be used in calculating the overall cost of the software.*

***Keywords:** Software Metrics, Assessment Metric, Component-based software systems, Assessment Cost, Integration Cost, Tailoring Costs.*

I. INTRODUCTION

Component based software system allows selection of appropriate components at a proper time is highly desirable to achieve objectives of better quality product within specified time and budget constraints. In component selection process component evaluation is a most critical activity. Components ought to be evaluated in both aspects-technically (quality and functionality) as well as non-technically (cost and vendor support) (Brereton and Budgen, 2000). One out of many strategies of component evaluation is to judge its design for numerous concepts such as cohesion, coupling and complexity. Software metrics are conducive in many ways to produce quality software products within specified time and budget constraints.

For the overall quality of the system the interaction among components in an assembly is important. This means that the “best” component of a particular assembly may not be the overall “best” component available for that kind of functionality. Instead, those components should be chosen that maximizes the overall system quality. In this assembly centric view, individual component assessment may be performed as a fraction of the component assembly evaluation, but the main focus should be on the overall best solution.

Revised Manuscript Received on February 05, 2020.

* Correspondence Author
Dr. Rajul Jain, Assistant professor, Dept. of Computer science, Mata Gujri Mahila Mahavidyalaya (Auto), Jabalpur (M.P)

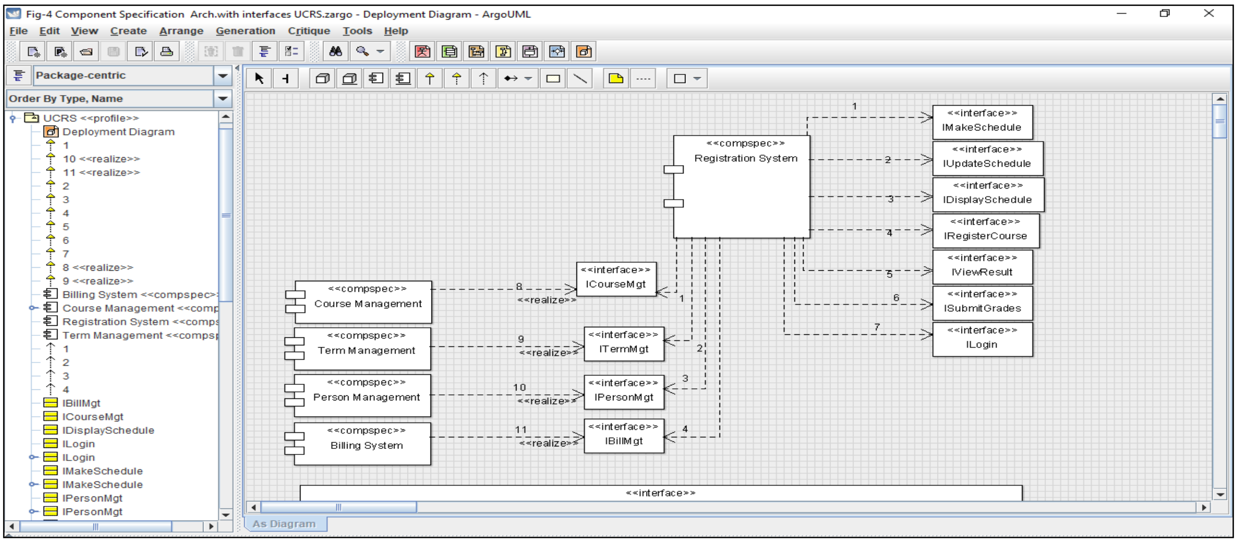
Component assembly evaluation is of two types: qualitative or quantitative. The qualitative evaluation can be performed by an expert subjected to availableness but it can be biased. Evaluation done by different experts may be conflicting and it is difficult to do comparison between evaluations performed on different assemblies. In contrast quantitative evaluation can be objective and repeatable, by using well-defined metrics thus enabling to compare the results of evaluations performed on various assemblies. The domain of software metrics includes proposals for process and product assessment. In this study, we are mainly concern with product metrics with prime concern on metrics that address reusability.

II. LITERATURE REVIEW

Miguel Goulao et al. presented a metric formalization technique on the basis of use of ontology with a formal specification language. Jianguo Chen et al. suggested a formal direct and indirect component coupling metric for both individual component and assembly between components. P. K. Suri et al. presented the metrics for evaluating the independency of component for reusability. The use of chi-square test has been made for evaluation. V. Lakshmi Narasimhan et al. described a systematic comparisons of three suite of metrics allowing a user to choose the best applicable as per the need. P. Edith Linda et al. made comparison among various algorithms on the basis of their performance and memory usage. Sidhu Pravneet described an objective way to calculate the quality of software component by using component quality metrics like presence, Ivalues and ratios. These quality metrics have been used to define the exact quality of an artificial intelligence component which is the AI back propagation algorithm. Hesham Abandah et al. presented the effectiveness and power of call graph based metrics by evaluating the many categories of bugs. Taranjeet Kaur et al. made comparison of various lack of cohesion metrics to increase the fault prediction power and to decrease the complexity.

III. ABOUT UCRS

UCRS (Jawwad W. Shareef et al.,2012) is a automation system for the universities and provides various facilities to the students, faculties and staff. Within this system, a student registers for classes. Once given access, the students may select a term and build a class schedule from the offered classes. The system passes information about a student’s schedule to the billing system. A student can also register, add, or drop a course.



Snapshot 1. Component Diagram of UCRS Registraton System (Jawwad W.Shareef et al.,2012)

An instructor may use the registration system to print a student class list and to submit grades for her/his class. The administrator may maintain student and teacher

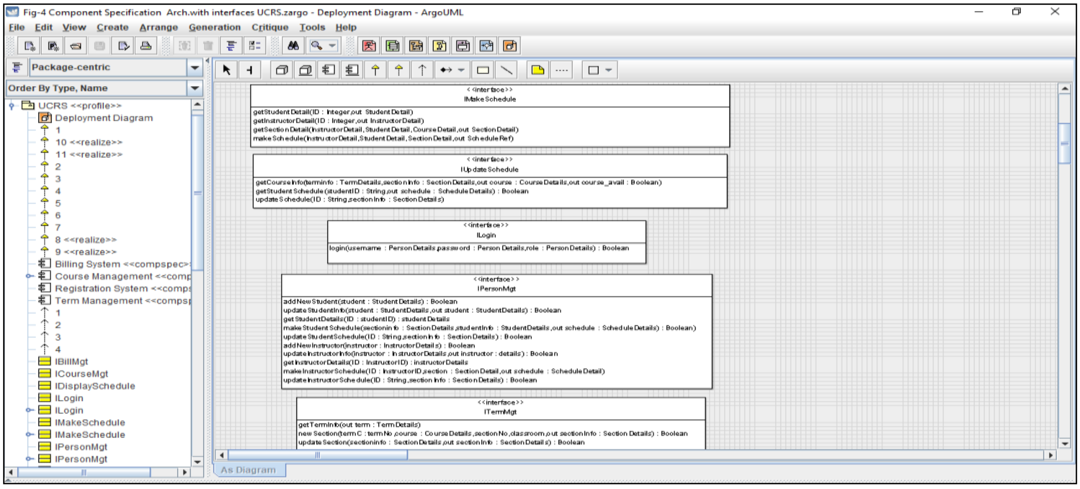
information. This model provides an overall view of the system and helps to demonstrate the extraction of existing component assembly complexity metrics. The component, RegistrationSystem, has seven provided interfaces namely, IMakeSchedule, IUpdateSchedule, IDisplaySchedule, IRegisterCourse, IViewResult, ISubmitGrades and ILogin which in ArgoUML tool are linked by an arrow known as (Abstraction). Similarly there are four required interface s of component 'Registration System', linked by an arrow known as (Dependency). These required interfaces serve as provided interfaces for the following.

- ICourseMgt by component 'Course Management'
- ITermMgt by component 'Term Management'
- IPersonMgt by component 'Person Management'
- IBillMgt by component 'Billing System'

After the modeling of UCRS is completed, the metrics are derived using ArgoUML tool, the XMI 1.2 file is generated with the help of Export XMI option (ArgoUML using Netbeans XMI Writer version 1.0). Using this XMI file, the metrics are derived by parsing the XMI 1.2 file. The UCRS model in XMI is identified by a unique id (UML:Model xmi.id). The XMI file contains information of all components by assigning a unique (UML:Component xmi.id) to each component.The component provided and required interfaces are shown as a link pointed to a stereotype <<interface>>, here in XMI file the component which provides an interface to other components is identified by (UML:Dependency.client) by assigning a unique (UML:Component xmi.idref) to each component, the link which carries this dependency to the stereotype <<interface>> is identified by (UML:Abstraction) assigning a unique (xmi.idref), similarly for a required interface of a component the link which carries this dependency to the stereotype <<interface>> is identified by (UML: Dependency. supplier) in the system.

The XMI files stores all necessary information regarding UCRS model. This file is parsed Java Parser tool developed with the help of Argo UML parser; to

derive different metrics related to component assembly, using a Java API tool.



```
<<interface>>
IMakeSchedule
getStudentDetail(ID : Integer,out StudentDetail)
getInstructorDetail(ID : Integer,out InstructorDetail)
getSectionDetail(InstructorDetail,StudentDetail,CourseDetail,out SectionDetail)
makeSchedule(InstructorDetail,StudentDetail,SectionDetail,out ScheduleRef)

<<interface>>
IUpdateSchedule
getCourseInfo(terminfo : TermDetails,sectionInfo : SectionDetails,out course : CourseDetails,out course_avail : Boolean)
getStudentSchedule(studentID : String,out schedule : ScheduleDetails) : Boolean
updateSchedule(ID : String,sectionInfo : SectionDetails)

<<interface>>
ILogin
login(username : PersonDetails,password : PersonDetails,role : PersonDetails) : Boolean

<<interface>>
IPersonMgt
addNewStudent(student : StudentDetails) : Boolean
updateStudentInfo(student : StudentDetails,out student : StudentDetails) : Boolean
getStudentDetails(ID : studentID) : studentDetails
makeStudentSchedule(sectioninfo : SectionDetails,studentInfo : StudentDetails,out schedule : ScheduleDetails) : Boolean)
updateStudentSchedule(ID : String,sectionInfo : SectionDetails) : Boolean
addNewInstructor(instructor : InstructorDetails) : Boolean
updateInstructorInfo(instructor : InstructorDetails,out instructor : details) : Boolean
getInstructorDetails(ID : InstructorID) : instructorDetails
makeInstructorSchedule(ID : InstructorID,section : SectionDetail,out schedule : ScheduleDetail)
updateInstructorSchedule(ID : String,section Info : SectionDetails) : Boolean

<<interface>>
ITermMgt
getTermInfo(out term : TermDetails)
newSection(termC : termNo,course : CourseDetails,sectionNo,classroom,out sectionInfo : SectionDetails) : Boolean
updateSection(sectioninfo : SectionDetails,out sectionInfo : SectionDetails) : Boolean
```

Snapshot 2. Various Interfaces with Operations and Parameters of UCRS Registration System

IV. ASSESSMENT METRICS:

Hoek et al. & Narasimhan and Hendradjaya, have assessed the fitness of component's aim using a specific architecture. Depending on the assembly a component will be integrated on the basis of context specifications. It is done by sensing that the same component may have different metric values.

4.1 Component service utilization metrics-

(i) The Provided Services Utilization (PSU) is represented by the ratio of services provided by the component actually used to the total number of services. Similarly Required Services Utilization (RSU) is evaluated:

$$\text{PSUx} = \frac{Pactual}{Ptotal}$$

Where $Pactual$ = actual number of services provided by component which are used by other components

$Ptotal$ = total number of services provided by component and

$$\text{RSUx} = \frac{Ractual}{Rtotal}$$

Where $Ractual$ = actual number of services required by component which are provided by other components.

$Rtotal$ = total number of services required by component.

(ii) PSU depicts the extent to which assembly uses the services provided by component. Although a low value of PSU may occur if a component was built for reuse, it also means that the component provides large amount of extra functionalities which are not required by assembly. RSU specifies the extent to which services are required by a component and are available in component assembly. Ideally, RSU's value should be 1.

(iii) As presented by Hoek et al. the notion of service is fuzzy, it means it covers any kind of publicly accessible resource of the component, including operations, data structures and so on. It further means to say that the granularity of notion of a service also varies, as per the need and suitability for Architecture Description Language (ADL) which is used to express the component assemblies under analysis.

In our formalization interfaces are assumed as services which any component provides, or requires, and is in line with the granularity level established for component dependency.

4.2 Compound Service Utilization metrics-

(i) The Compound Provided Service Utilization (CPSU) is ratio of actual number of provided services used from the services provided by all components in the assembly to the total number of services being provided by all components in the assembly. Similarly the Compound Required Service Utilization (CRSU) is evaluated.

$$\text{CPSUx} = \frac{\sum_{\mathrm{i-1}}^{\mathrm{n}} Pactual^{\mathrm{i}}}{\sum_{\mathrm{i-1}}^{\mathrm{n}} Ptotal^{\mathrm{i}}}$$

Where $Pactual^{\mathrm{i}}$ = actual number of services provided by ith component of the assembly that are actually used by other components

$Ptotal^{\mathrm{i}}$ = total number of services provided by ith component and

$$\text{CRSUx} = \frac{\sum_{\mathrm{i-1}}^{\mathrm{n}} Ractual^{\mathrm{i}}}{\sum_{\mathrm{i-1}}^{\mathrm{n}} Rtotal^{\mathrm{i}}}$$

Where $Ractual^{\mathrm{i}}$ = number of services required by ith component that are provided by the assembly.

$Rtotal^{\mathrm{i}}$ = number of services required by ith component in the assembly.

(ii) Small values of CPSU indicate architecture unbalance; this implies that assembly is larger than requirement. A small value of CRSU also suggests that assembly is not sufficient, means more components will be required to fulfill the desired functionalities.

(iii) The assumptions in this formalization follow that the CPSU and CRSU are defined in scope of assembly.

4.3 Interaction density of a component-

(i) It is defined as a ratio of actual interactions to the potential interactions. The Incoming Interaction Density of a Component (IIDC) and Outgoing Interaction Density of a Component (OIDC), are also similar and are calculated as:

$$\text{IDC} = \frac{\# I}{\# I_{max}}$$

Where $\# I$ = Actual Interactions and $\# I_{max}$ = Maximum available interactions and

$$\text{IIDC} = \frac{\# I_{IN}}{\# I_{maxin}}$$

Where $\# I_{IN}$ = Actual incoming interactions and I_{maxin} = Maximum available incoming interactions

$$\text{OIDC} = \frac{\# I_{out}}{\# I_{maxout}}$$

Where $\# I_{out}$ = Actual outgoing interactions and I_{maxout} = Maximum available outgoing interactions.

(ii) A higher interaction density causes a higher complexity in the interaction. Narashiman and Hendrajaya have taken this complexity as a source of risk during assigning professionals to design components. This means only experienced developers must be assigned on tasks involving high density interactions.

4.4 Average Interaction Density of Software Components –

(i) This represents the sum of IDC for each component divided by the total number of components.

$$\text{AIDC} = \frac{IDC_1 + IDC_2 + IDC_3 .... + IDC_n}{\# \, Components}$$

Where IDC_i = IDC of component I $\# \, Components$ = Number of components in the system

(ii) Narashiman and Hendrajaya suggest that AIDC can be used as an explanatory variable for assembly complexity, so that a lower value may indicate that less effort can be committed to software risk analysis.

4.5 Overall Assessment Cost for the System:

From the metric values defined above we can evaluate the overall assessment cost of the system. The assessment cost is the cost of the evaluation of the component for applicability in the new system. This cost is affected by the different metric values as follows:

Component Service utilization metrics i.e. PSU and RSU indicated that how much involved is the component i.e. how much dependencies it has on other components and how much other components are dependent on the chosen component. Higher the provided service utilization means the system will be dependent on more number of output components and will provide more services, whereas the required service utilization indicates that how many components are required by the chosen component to function fully. Hence the value of PSU should be more and that of RSU should be less. Hence in this work we have taken higher ratio of PSU (70%) and lower ratio of RSU (30%). 70% and 30% are arbitrary values which can be adjusted depending on the quality of service.

(1) Compound Component Service utilization metrics i.e. CPSU and CRSU indicated that all components are involved how much i.e. how much dependency they have on other components and how much other components are dependent on the other chosen component. Higher the compound provided service utilization means the system will be dependent on more number of output components and will provide more services, whereas the compound required service utilization indicates that

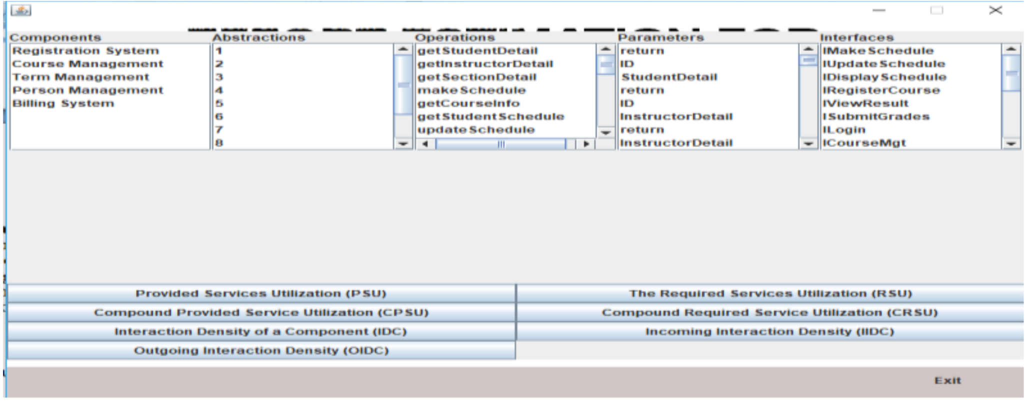
how many components are required by the other chosen component to function fully. Hence the value of CPSU should be more and that of CRSU should be less. Hence in this work we have taken higher ratio of CPSU (70%) and lower ratio of CRSU (30%). 70% and 30% are arbitrary values which can be adjusted depending on the quality of service.

Interaction Density of a component metrics i.e. IIDC and OIDC indicated that how much interactions any component has with other components whether they are in use or not. Such densities should be lower on input end whereas higher on the output end. Since they may or may not be used hence the low contribution value chosen for IIDC (40%) and high contribution value chosen for OIDC (60%). 40% and 60% are arbitrary values which can be adjusted depending on the quality of service.

4) Overall Assessment Cost is combination of all above costs again contribution of each one of them may be different as per the requirement of the quality of the service and any other factors such as monetary cost, human efforts, code efforts etc. In this work we have taken ECPSU to be 40%, EPSU to be 40% and EIDC to be 20%. The selection ratios are based on the intensity of the metrics in the system. Overall assessment cost calculated in this work should be having higher value to indicate the component to be suitable for the system. Ideally should be very high. Lower values indicate the component selection may lead to extra burden on the system and organization.

V. IMPLEMENTATION & RESULTS:

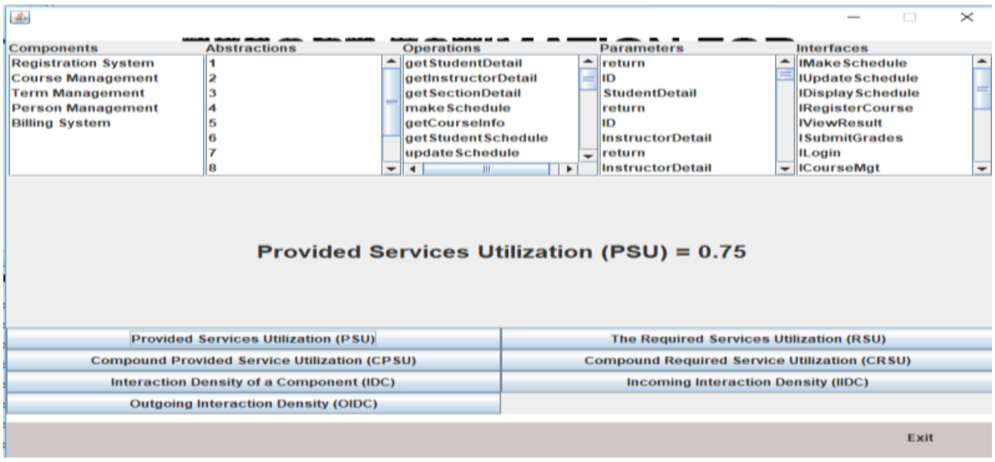
A JAVA based application has been developed to evaluate the assessment cost for the case study of UCRS to estimate the software efforts. The implementation has been done to list all the components, interfaces, operations and parameters in the system by parsing the XMI file. As proposed in section above for evaluation of assessment cost, various statistics regarding the provided and required interfaces has been calculated as stated.



Components	Abstractions	Operations	Parameters	Interfaces
Registration System	1	getStudentDetail	return	IMakeSchedule
Course Management	2	getInstructorDetail	ID	IUpdateSchedule
Term Management	3	getSectionDetail	StudentDetail	IDisplaySchedule
Person Management	4	makeSchedule	return	IRegisterCourse
Billing System	5	getCourseInfo	ID	IViewResult
	6	getStudentSchedule	InstructorDetail	ISubmitGrades
	7	updateSchedule	return	ILogin
	8		InstructorDetail	ICourseMgt

Provided Services Utilization (PSU)
The Required Services Utilization (RSU)
Compound Provided Service Utilization (CPSU)
Compound Required Service Utilization (CRSU)
Interaction Density of a Component (IDC)
Incoming Interaction Density (IIDC)
Outgoing Interaction Density (OIDC)
Exit

Snapshot 3. Control Panel view for evaluation of Assessment Cost of the Component Based Model of UCRS

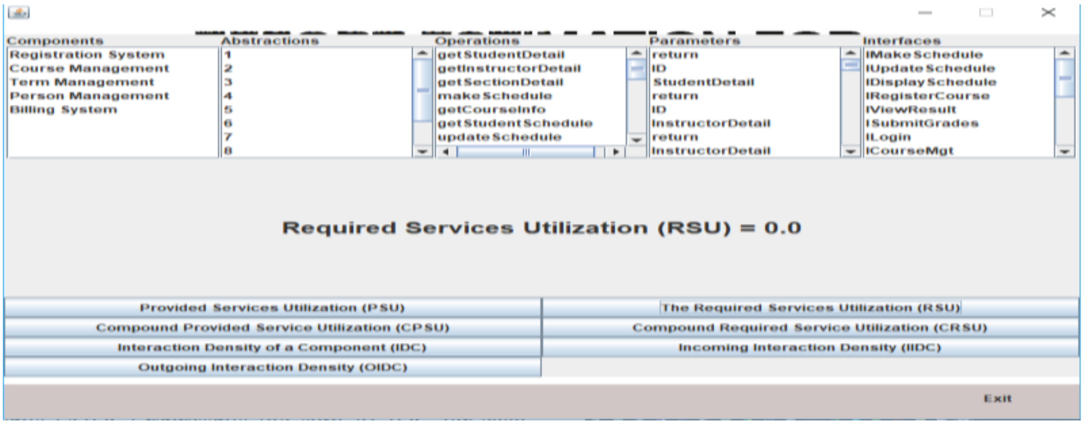


Components	Abstractions	Operations	Parameters	Interfaces
Registration System	1	getStudentDetail	return	IMakeSchedule
Course Management	2	getInstructorDetail	ID	IUpdateSchedule
Term Management	3	getSectionDetail	StudentDetail	IDisplaySchedule
Person Management	4	makeSchedule	return	IRegisterCourse
Billing System	5	getCourseInfo	ID	IViewResult
	6	getStudentSchedule	InstructorDetail	ISubmitGrades
	7	updateSchedule	return	ILogin
	8		InstructorDetail	ICourseMgt

Provided Services Utilization (PSU) = 0.75

Provided Services Utilization (PSU)
The Required Services Utilization (RSU)
Compound Provided Service Utilization (CPSU)
Compound Required Service Utilization (CRSU)
Interaction Density of a Component (IDC)
Incoming Interaction Density (IIDC)
Outgoing Interaction Density (OIDC)
Exit

Snapshot 4. Control Panel view for evaluation of Assessment Cost of the Component Based Model of UCRS

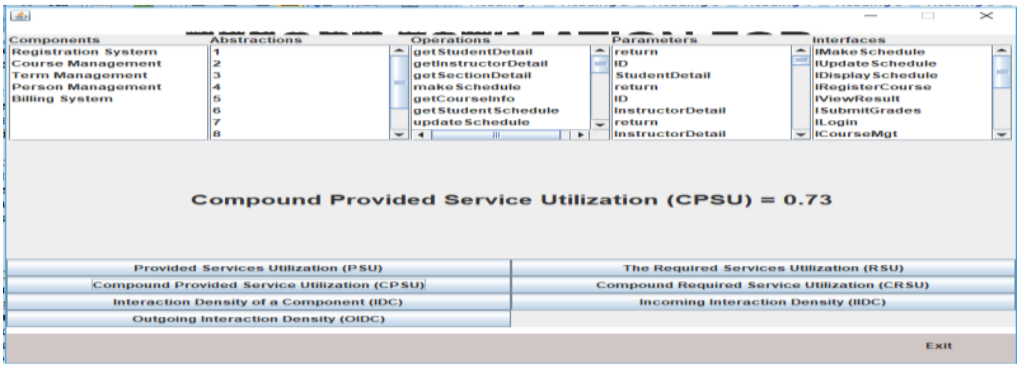


Components	Abstractions	Operations	Parameters	Interfaces
Registration System	1	getStudentDetail	return	IMakeSchedule
Course Management	2	getInstructorDetail	ID	IUpdateSchedule
Term Management	3	getSectionDetail	StudentDetail	IDisplaySchedule
Person Management	4	makeSchedule	return	IRegisterCourse
Billing System	5	getCourseInfo	ID	IViewResult
	6	getStudentSchedule	InstructorDetail	ISubmitGrades
	7	updateSchedule	return	ILogin
	8		InstructorDetail	ICourseMgt

Required Services Utilization (RSU) = 0.0

Provided Services Utilization (PSU)
The Required Services Utilization (RSU)
Compound Provided Service Utilization (CPSU)
Compound Required Service Utilization (CRSU)
Interaction Density of a Component (IDC)
Incoming Interaction Density (IIDC)
Outgoing Interaction Density (OIDC)
Exit

Snapshot 5. Control Panel view for evaluation of Assessment Cost of the Component Based Model of UCRS

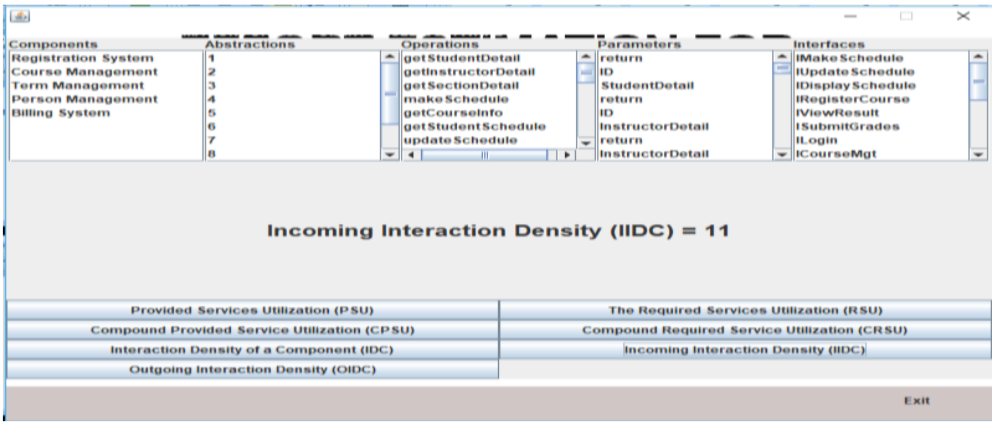


Components	Abstractions	Operations	Parameters	Interfaces
Registration System	1	getStudentDetail	return	IMakeSchedule
Course Management	2	getInstructorDetail	ID	IUpdateSchedule
Term Management	3	getSectionDetail	StudentDetail	IDisplaySchedule
Person Management	4	makeSchedule	return	IRegisterCourse
Billing System	5	getCourseInfo	ID	IViewResult
	6	getStudentSchedule	InstructorDetail	ISubmitGrades
	7	updateSchedule	return	ILogin
	8		InstructorDetail	ICourseMgt

Compound Provided Service Utilization (CPSU) = 0.73

Provided Services Utilization (PSU)
The Required Services Utilization (RSU)
Compound Provided Service Utilization (CPSU)
Compound Required Service Utilization (CRSU)
Interaction Density of a Component (IDC)
Incoming Interaction Density (IIDC)
Outgoing Interaction Density (OIDC)
Exit

Snapshot 6. Control Panel view for evaluation of Assessment Cost of the Component Based Model of UCRS

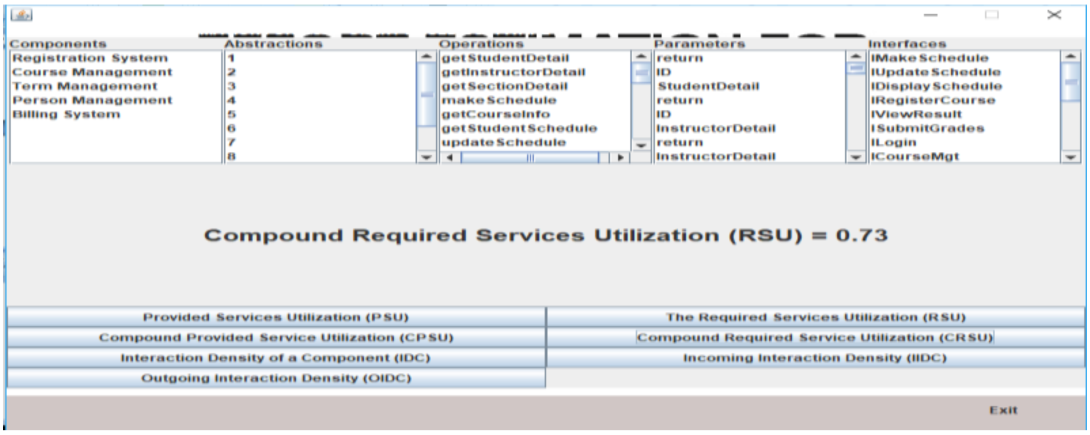


Components	Abstractions	Operations	Parameters	Interfaces
Registration System	1	getStudentDetail	return	IMakeSchedule
Course Management	2	getInstructorDetail	ID	IUpdateSchedule
Term Management	3	getSectionDetail	StudentDetail	IDisplaySchedule
Person Management	4	makeSchedule	return	IRegisterCourse
Billing System	5	getCourseInfo	ID	IViewResult
	6	getStudentSchedule	InstructorDetail	ISubmitGrades
	7	updateSchedule	return	ILogin
	8		InstructorDetail	ICourseMgt

Incoming Interaction Density (IIDC) = 11

Provided Services Utilization (PSU)
The Required Services Utilization (RSU)
Compound Provided Service Utilization (CPSU)
Compound Required Service Utilization (CRSU)
Interaction Density of a Component (IDC)
Incoming Interaction Density (IIDC)
Outgoing Interaction Density (OIDC)
Exit

Snapshot 7. Control Panel view for evaluation of Assessment Cost of the Component Based Model of UCRS

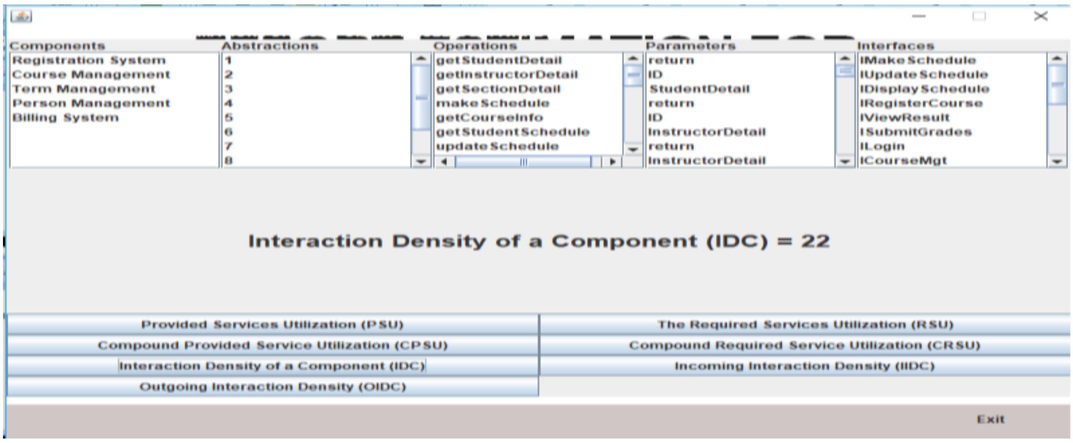


Components	Abstractions	Operations	Parameters	Interfaces
Registration System	1	getStudentDetail	return	IMakeSchedule
Course Management	2	getInstructorDetail	ID	IUpdateSchedule
Term Management	3	getSectionDetail	StudentDetail	IDisplaySchedule
Person Management	4	makeSchedule	return	IRegisterCourse
Billing System	5	getCourseInfo	ID	IViewResult
	6	getStudentSchedule	InstructorDetail	ISubmitGrades
	7	updateSchedule	return	ILogin
	8		InstructorDetail	ICourseMgt

Compound Required Services Utilization (RSU) = 0.73

Provided Services Utilization (PSU)
The Required Services Utilization (RSU)
Compound Provided Service Utilization (CPSU)
Compound Required Service Utilization (CRSU)
Interaction Density of a Component (IDC)
Incoming Interaction Density (IIDC)
Outgoing Interaction Density (OIDC)
Exit

Snapshot 8. Control Panel view for evaluation of Assessment Cost of the Component Based Model of UCRS

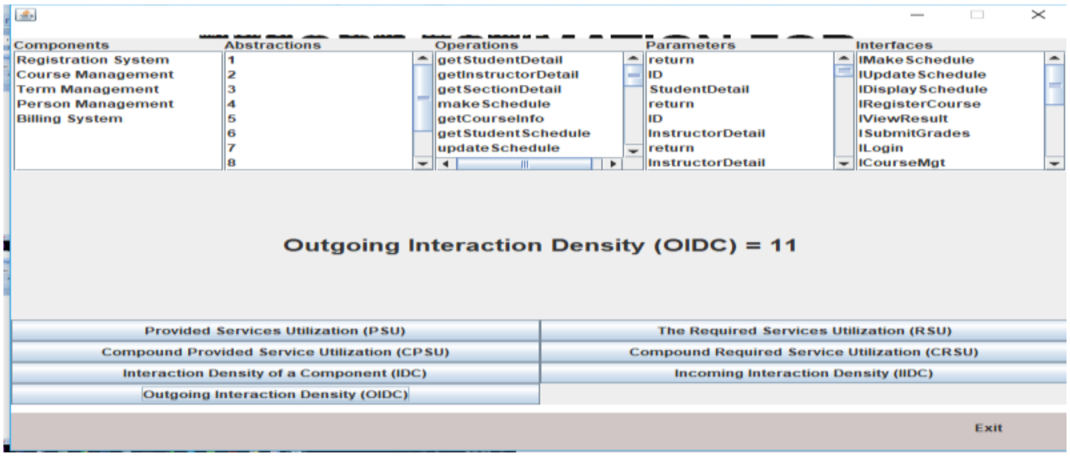


Components	Abstractions	Operations	Parameters	Interfaces
Registration System	1	getStudentDetail	return	IMakeSchedule
Course Management	2	getInstructorDetail	ID	IUpdateSchedule
Term Management	3	getSectionDetail	StudentDetail	IDisplaySchedule
Person Management	4	makeSchedule	return	IRegisterCourse
Billing System	5	getCourseInfo	ID	IViewResult
	6	getStudentSchedule	InstructorDetail	ISubmitGrades
	7	updateSchedule	return	ILogin
	8		InstructorDetail	ICourseMgt

Interaction Density of a Component (IDC) = 22

Provided Services Utilization (PSU)
The Required Services Utilization (RSU)
Compound Provided Service Utilization (CPSU)
Compound Required Service Utilization (CRSU)
Interaction Density of a Component (IDC)
Incoming Interaction Density (IIDC)
Outgoing Interaction Density (OIDC)
Exit

Snapshot 9. Control Panel view for evaluation of Assessment Cost of the Component Based Model of UCRS



Components	Abstractions	Operations	Parameters	Interfaces
Registration System	1	getStudentDetail	return	IMakeSchedule
Course Management	2	getInstructorDetail	ID	IUpdateSchedule
Term Management	3	getSectionDetail	StudentDetail	IDisplaySchedule
Person Management	4	makeSchedule	return	IRegisterCourse
Billing System	5	getCourseInfo	ID	IViewResult
	6	getStudentSchedule	InstructorDetail	ISubmitGrades
	7	updateSchedule	return	ILogin
	8		InstructorDetail	ICourseMgt

Outgoing Interaction Density (OIDC) = 11

Provided Services Utilization (PSU)
The Required Services Utilization (RSU)
Compound Provided Service Utilization (CPSU)
Compound Required Service Utilization (CRSU)
Interaction Density of a Component (IDC)
Incoming Interaction Density (IIDC)
Outgoing Interaction Density (OIDC)
Exit

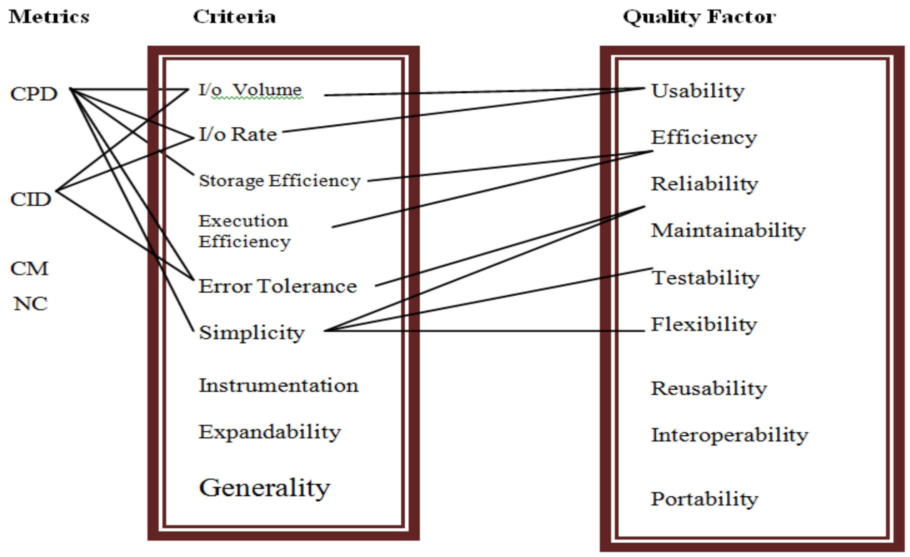
Snapshot 10. Control Panel view for evaluation of Assessment Cost of the Component Based Model of UCRS

Significance of Assessment metrics in **Assessment Effort Estimation**

Assessment metrics discussed above have the following significance-

Service utilization metrics i.e. PSU and RSU indicate that how much involved is the component i.e. how much dependencies it has on other components and how much other components are dependent on the selected component.

Higher the provided service utilization means the system will be dependent on more number of output components and will provide more services.



Fig, Assessment metrics Vs. Quality Factor

The required service utilization indicates that how many components are required by the chosen component to function fully. Hence the value of PSU should be more and that of RSU should be less.

Compound Component Service utilization metrics i.e. CPSU and CRSU indicates that all components are involved how much i.e. how much dependency they have on other components.

Higher the compound provided service utilization (CPSU) means the system will be dependent on more number of output components and will provide more services, whereas the compound required service utilization indicates that how many components are required by the other chosen component to function fully. Hence the value of CPSU should be more and that of CRSU should be less.

Interaction Density of the component metrics i.e. IIDC and OIDC indicates that how much interactions any component has with other components. Ideally it should be low. Higher values indicate that component is more complex and component selection may lead to extra burden on the system.

Suit of Assessment metrics Vs. Quality factors –

Narasimhan V. and Hendrajayaji B. (Narasimhan V. and Hendrajayaji B, 2007) related metrics suit to the quality factors and hence the assessment attributes of the components.

Component Interaction Density is useful for measuring the Usability factors, Efficiency, Reliability, Maintainability and Testability.

- Interaction among the components results in dependencies, the more the interaction among components; the system will be more complex which results in poor understanding.
- High Interaction among component results means higher complexity thus maintenance cost is also high.

High interaction results in low reliability thus Interaction complexity should have low value for high reusability.

Low value of CID increases simplicity and hence reliability is also increases.

Assessment Effort Calculation-

Assessment is the process of selecting appropriate components for use in the system being developed. It is the method of determining the feasibility or appropriateness of Components to realize required system utilities Assessment is done in two steps- initial filtering and final selection. Initial filtering is the quick and

dirty effort required to identify those components which are inferior in quality and thus are unsuitable in the current framework. Final selection is the process of carefully analyzing the remaining components after rejecting the unsuitable components in order to obtain the final set of components. Initial filtering is simply called 'assessment' and final selection as 'evaluation'..(Abts C. and Boehm B.,2001)

Total Assessment effort= Initial filtering effort + Final selection effort

Initial Filtering effort calculation-

This technique presumes some parameterized value of initial filtering effort per candidate for the given domain. Initial filtering effort is then the product of number of components being filtered and initial filtering effort per candidate. (Abts C. and Boehm B.,2001).

Table-1. Domain for initial filtering effort (Abts C. and Boehm B.,2001)

Domain	A	B	C	D
No of components filtered	150-200	100-150	50-100	0-50
Initial filtering effort per candidate	.03pm/candidate	.02pm/candidate	.01pm/candidate	.005pm/candidate

The steps to be followed to estimate initial filtering effort-

STEP-1: First decide the domain under which project falls.

Suppose project falls in Domain C, so the initial filtering effort is .01pm/candidate.

STEP-2: Now decide number of components for the given domain to be filtered.

Suppose approximate 75 components to be considered for rough filtering.

Thus initial filtering effort = 75 X .01pm/candidate =0.75 pm

Final selection effort estimation-

Step-1 Decide how many components will go through final selection assessment? A general rule of thumb is that about 20% of initial filtering effort domain is assessed in the final assessment effort estimation step. Thus 20% of 75 = 15 components will go through final assessment round.

Step-2 This step is more complex where each component is assessed in light of certain product attributes. Abts C.et al. (Abts C. and Boehm B.,2001)defines the following seventeen assessment attributes on which COTs components are assessed, these are-

Now each attribute is parameterized according to standardized rating of its relative importance on the scale from extra low to extra high. The table below summarizes the rating scale applied to each assessment attribute.

Tool developed for evaluating the values of provided service utilization(PSU), Required service Utilization (RSU), Compound Provided service utilization (CPSU), Compound Required Service Utilization (CRSU), Component Incoming Interaction density (IIDC) and Component Outgoing Interaction Density(OIDC)

Table-2 Standard Rating scale for component Assessment (Abts C. and Boehm B.,2001)

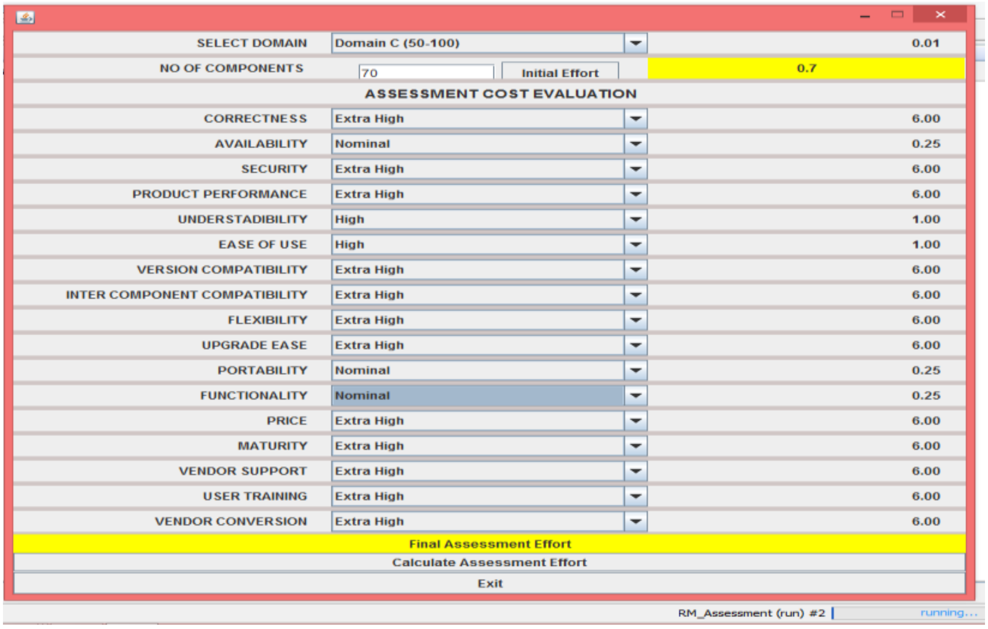
Correctness	Reliability/availability	Security	Understandability
Ease of use	Upgrade ease	Functionality	Flexibility
Portability	Product performance	Price	Vendor Support
User Training	Inter-component compatibility	Maturity	Version compatibility

assists in providing the ratings for some of the assessment attributes such as Understandability, Reliability, Ease of use, functionality and Portability.

Step 3: Now rate the remaining attributes from very low to very high. For example the components are rated such as-

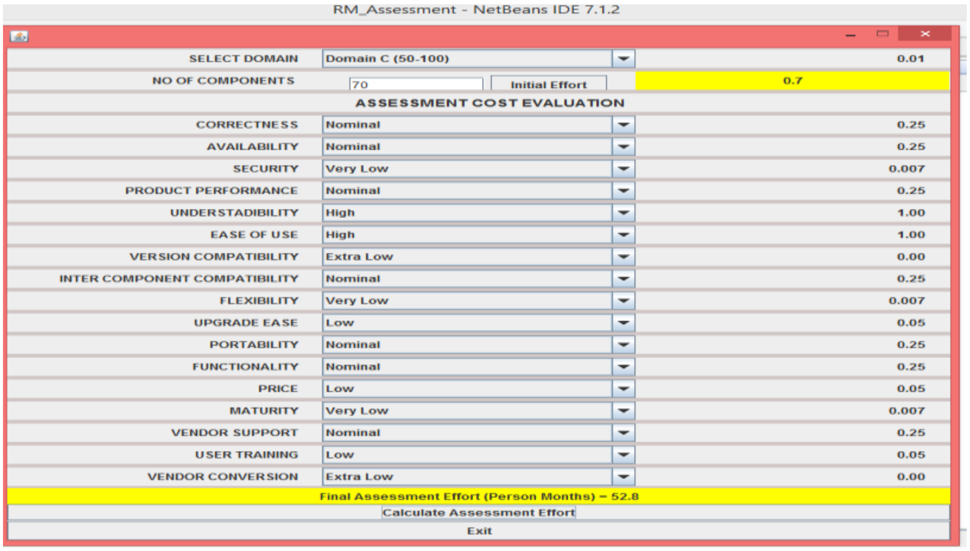
Table-3 Standard Rating scale for component Assessment (Abts C. and Boehm B.,2001)

Rating	Extra Low	Very Low	Low	Nominal	High	Very High	Extra High
Importance of the component	Irrelevant	Unnecessary	Somewhat Useful	Useful	Desirable	Important	Mondatory
Average Assessment Effort (person-months)	0	0.007	0.05	0.25	1	3	6



SELECT DOMAIN	Domain C (50-100)	0.01
NO OF COMPONENTS	70	0.7
ASSESSMENT COST EVALUATION		
CORRECTNESS	Extra High	6.00
AVAILABILITY	Nominal	0.25
SECURITY	Extra High	6.00
PRODUCT PERFORMANCE	Extra High	6.00
UNDERSTADIBILITY	High	1.00
EASE OF USE	High	1.00
VERSION COMPATIBILITY	Extra High	6.00
INTER COMPONENT COMPATIBILITY	Extra High	6.00
FLEXIBILITY	Extra High	6.00
UPGRADE EASE	Extra High	6.00
PORTABILITY	Nominal	0.25
FUNCTIONALITY	Nominal	0.25
PRICE	Extra High	6.00
MATURITY	Extra High	6.00
VENDOR SUPPORT	Extra High	6.00
USER TRAINING	Extra High	6.00
VENDOR CONVERSION	Extra High	6.00

Final Assessment Effort



RM_Assessment - NetBeans IDE 7.1.2

SELECT DOMAIN	Domain C (50-100)	0.01
NO OF COMPONENTS	70	0.7
ASSESSMENT COST EVALUATION		
CORRECTNESS	Nominal	0.25
AVAILABILITY	Nominal	0.25
SECURITY	Very Low	0.007
PRODUCT PERFORMANCE	Nominal	0.25
UNDERSTADIBILITY	High	1.00
EASE OF USE	High	1.00
VERSION COMPATIBILITY	Extra Low	0.00
INTER COMPONENT COMPATIBILITY	Nominal	0.25
FLEXIBILITY	Very Low	0.007
UPGRADE EASE	Low	0.05
PORTABILITY	Nominal	0.25
FUNCTIONALITY	Nominal	0.25
PRICE	Low	0.05
MATURITY	Very Low	0.007
VENDOR SUPPORT	Nominal	0.25
USER TRAINING	Low	0.05
VENDOR CONVERSION	Extra Low	0.00

Final Assessment Effort (Person Months) = 52.8

Retrieval Number: C5764029320/2020©BEIESP
DOI: 10.35940/ijeat.C5764.029320
Journal Website: www.ijeat.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Step 4 : Final Selection filtering effort=(14 candidates x o.25 pm/candidate) + (14 candidates x 0.25 pm/candidate) +(14 candidates x 0.007 pm/candidate) + (14 candidates x 0.25 pm/candidate) + (14 candidates x 1.00 pm/candidate) + (14 candidates x 1.00 pm/candidate) +(14 candidates x 0.00 pm/candidate) + (14 candidates x 0.25pm/candidate) + (14 candidates x 0.007 pm/candidate) + (14 candidates x 0.05 pm/candidate) + (14 candidates x 0.25 pm/candidate) + (14 candidates x 0.25 pm/candidate) + 14 candidates x 0.005 pm/candidate) + (14 candidates x 0.07 pm/candidate) + (14 candidates x 0.25 pm/candidate) + (14 candidates x 0.05 pm/candidate) + (14 candidates x 0.00 pm/candidate)=52.1 pm

CONCLUSION: Calculation of assessment cost i.e. the cost of integration has been proposed to be based on provided and

required service utilizations involved in the various components of the system. The values presented in the previous section illustrate some of the ideas that lead to the proposal of the corresponding metrics. All the provided services and emitted events are used or consumed by components within the assembly. If all provided services are not used by the components then the waste of resources occurs. All the formalized metrics are defined as ratios where the nominator corresponds to the effective usage of a given mechanism, while the denominator has the maximum possible utilization of the mechanism within the component assembly. This indicates a concern from the metrics proponents to make them dimensionless. This prevents the metrics values from being correlated to the size of the assembly, or the number of times a particular mechanism is used.

VI. FUTURE RESEARCH:

The tool can be further upgraded for estimation of tailoring cost for component-based systems.

Other metrics related to Component-based systems can be included in enhanced version of the tool proposed.

REFERENCES

1. BRERETON, P., AND BUDGEN D., November 2000, "Component-Based Systems: A Classification of Issues," IEEE Computers (33:11), pp. 54-62.
2. BROWNSWORD, L., OBERNDORF, T., AND SLEDGE, C. A., 2000, "Developing New Processes for COTS-Based Systems", IEEE Software, pp. 48-55, 2000.
3. GOULÃO, M. AND ABREU, F. B., 2004, "Software Components Evaluation: an Overview", 5ª Conferência da APSI, Lisbon.
4. GRADY BOOCH 1994, "*Object –Oriented Analysis And Design*", ISBN 0-8053-5340-2 15 1617181920 DOC 0 1 00 99 98
5. HESHAM ABANDAH AND IZZAT ALSMADI, 2013,"Call Graph based Metrics to Evaluate Software Design Quality", International Journal of Software Engineering and its Applications, Vol.7, No.1, pp.1-12.
6. HOEK,A., DINCEL E., AND MEDVIDOVIC N., "Using Service Utilization Metrics to Assess and Improve Product Line Architectures", 9th IEEE International Software Metrics Symposium (Metrics'2003), Sydney, Australia, 2003.
7. J. MASCENA, E. ALMEIDA, ET AL. 2005, A Comparative Study on Software Reuse Metrics and Economic Models from a Traceability Perspective. IEEE Information Reuse and Integration. Las Vegas, USA.
8. JAWWAD W.SHREEF, PANDEY RAJESH KUMAR, 2012,"Dependency Analysis using UML for Component Based Software System : An XMI Approach " International Journal of Computer Applications (0975-888), Vol. 47, No 18, June 2012.
9. JIANGUO CHEN, WAI K. YEAP, STEFAN D. BRUDA, 2009, "A Review of Component Coupling Metrics for Component based Development" , Proceeding WCSE '09 Proceedings of the 2009 WRI World Congress on Software Engineering - Volume 04, pp. 65-69.
10. MILI, H., MILI, A., YACOUB, S., AND ADDY, E, 2002, *Reuse-Based Software Engineering, New York*, 2002.
11. Narasimhan, V. L. and Hendradjaya, B., 2004, "A New Suite of Metrics for the Integration of Software Components", *The First International Workshop on Object Systems and Software Architectures (WOSSA'2004),* South Australia, Australia.
12. P. EDITH LINDA, V. MANJU BASHINI, S. GOMATHI, 2011, "Metrics for Component Based Measurement Tools", International Journal of Scientific & Engineering Research Volume 2, Issue 5, May-2011 1 ISSN 2229-5518
13. POULIN JEFFREY S.,1994, *Proceedings of the Third International Conference on Software Reuse, Rio de Janeiro, Brazil, 1-4 November 1994.*
14. SIDHU PRAVNEET, 2012, "Quality metrics implementation in component based software engineering using AI back propagation algorithm software component", I.J.E.M.S., vol.3(2), pp. 109 – 114, ISSN 2229-600x.

AUTHOR PROFILE

Dr. Rajul Jain, received MCA, MSC (Physics) and Ph.D degree from Rani Durgawati Vishwavidyalaya, Jabalpur . She is presently working as Assistant professor in the Dept. of Computer science, Mata Gujri Mahila Mahavidyalaya (Auto), Jabalpur (M.P) since last 25 years. She has published several research papers in reputed international Journals and conferences. Her research area includes "Effort Estimation in component based Software Development Projects.".