Програм хангамжийн хөгжүүлэлтийн хэмжээг тооцох нь

Ч.Наранмандал
МУИС, ХШУИС, magii_89@yahoo.com
С.Уянга
МУИС, ХШУИС, uyanga@seas.num.edu.mn
Н.Мөнхцэцэг
МУИС, ХШУИС, munkhtsetseg@seas.num.edu.mn
Н.Баттүшиг
МУИС, БС, tushgee@gmail.com

***Abstract*— Аливаа мэдээллийн систем, багтаамж ихтэй програм хангамж хөгжүүлэх, түүний хөгжүүлэлтийн үйл ажиллагааг оновчтой удирдах асуудал чухал байдаг. Өнөөдөр улсын хэмжээнд, тодорхой салбар, байгууллага, аж ахуйн нэгжийн хүрээнд ашиглах мэдээллийн систем, програм хангамжийг үйлдвэрлэх, удирдах үйл ажиллагаанд програм хангамжийг хөгжүүлэх бодит хугацааг Монголын нөхцөл байдалд суурилан үндэслэлтэй тогтоох арга аргачлал шаардлагатай байна. Энэхүү илтгэлд програм хангамжийн хөгжүүлэлтийн хэмжээг тооцох аргачлалын талаар авч үзсэн болно.**

Түлхүүр үг*—*сосото; програм хангамжийн зардал, функцын цэгийн арга; ажлын өртөг зардал, хугацаа

I. ЗАРДАЛ, ЗАРДЛЫГ ТООЦООЛОХ АРГУУД

Зардлын тооцоо нь үйл ажиллагаа, арга гэсэн 2 хэсгээс бүрддэг. Энэ нь бодит болон нийт зардлыг тооцоход тусалдаг. Програм хангамж үйлдвэрлэгч нь хамгийн бага зардлаар (хямдаар), төлөвлөсөн хугацаанд чанартай бүтээгдэхүүнийг захиалагчид нийлүүлэх гол зорилготой ажилладаг. Програм хангамжийн зардлын тооцоо нь үндсэндээ системийн шинжээчид төсөлд шаардлагатай чухал нөөцүүд болон төслийн хугацааг нарийвчлалтайгаар авахад хэрэглэгддэг[1]. Зардлыг тооцоолоход хэмжээ, цаг хугацаа, хичээл зүтгэл гэсэн үзүүлэлтүүдийг авч үзнэ. Зардлыг тооцох үйл ажиллагаа нь үндсэн 4 алхамтай. Хамгийн эхэнд програм хангамжийн хэмжээг тооцож, улмаар тухайн програм хангамжийг хөгжүүлэхэд шаардагдах хичээл зүтгэлийг тооцоолдог. Ингэснээр програм хангамжийг хөгжүүлэхэд шаардлагатай нийт зардал болон хөгжүүлэлтийн хугацаа гарна.

Програм хангамжийн зардлыг тооцоолоход төрөл бүрийн арга техник хэрэглэгддэг. Эдгээр аргыг гол төлөв алгоритмын болон алгоритмын бус гэж хоёр ангилдаг. Алгоритмын арга нь математикийн тэгшитгэл дээр үндэслэх ба ийм аргуудаас COCOMA (Constructive Cost Model) арга хамгийн түгээмэл хэрэглэгддэг. Алгоритмын аргууд нь олон давуу талтай боловч тэдгээрийг эзэмшиж, ашиглахад төвөгтэй, харьцангуй олон тооны өгөгдөл шаарддаг онцлогтой. Харин алгоритмын бус аргууд нь сурахад хялбар боловч тухайн програм хангамжийн төслийг түүнтэй ижил төстэй, өмнөх төслүүдтэй харьцуулж, тэдгээрийн өгөгдөл дээр тооцоолдог тул өмнөх төслүүдийн талаарх бүрэн мэдээллийг шаарддаг[2]. Зардлыг тооцоолох дараах аргууд түгээмэл хэрэглэгддэг.

A. Алгоритмын бус аргууд

Алгоритмын бус аргууд нь адилтгах (analogy), хасах (deduction) үйл ажиллагаан дээр суурилж тооцоолдог. Иймээс хэмжээг нь тооцоолох програм хангамжаас өмнө хийгдэж байсан програм хангамжуудыг, тэдгээрийн ижил төстэй талуудыг мэдэх шаардлагатай байдаг. Зардлын үнэлгээ нь өмнөх програм хангамжийн төслийн үндсэн шинжилгээ эсвэл өгөгдлийн багц дээр хийгддэг. Алгоритмын бус онолын дараах арга, техник өргөн хэрэглэгддэг. Тухайлбал:

- Адилтгалд суурилсан үнэлгээ. Энэ нь зардлыг тооцохдоо тухайн төсөлтэй хамгийн ойр төстэй түүхт програм хангамжийг харьцуулж тооцдог. Өмнөх төслүүдийн өгөгдөл, үнийн дүнг харьцуулж тухайн төслийн зардлыг тооцоолон гаргадаг. Энэ аргыг мэдээллийн систем, эсвэл модулийн түвшинд аль алинд нь ашиглаж болно[3]. Адилтгал дээр үндэслэн тооцоход явагдах тодорхой үе шатуудтай. Үүнд:

1 Одоогийн төслийн шинж чанарыг тодорхойлох

2 Мэдээллийн санд хадгалагдаж буй шинж чанарааараа ижил төстэй төслийг олох

3 Ижил төстэй төслүүдээс одоогийн төслийн зардлыг тооцоолох.

- Мэргэжилтний үнэлэмжийн арга. Энэ арга нь шинжээчийн мэдлэг, туршалагад түлхүү суурилдаг. Зардлын үнэлгээ нь тэдгээр төсөлд хэдэн шинжээч оролцоноос хамаарна[4]. Мэдээллийг тодорхойлох, илрүүлэх, цуглуулах үйл ажиллагаа хязгаарлагдмал үед энэ аргыг ашиглах нь тохиромжтой. Програм хангамжийн төслийн стратегийг тооцоолоход өргөнөөр ашиглагддаг. Мэргэжилтний үнэлэмж арга дээр суурьлагдсан зардал тооцох арга бол Wideband Delphi арга юм [5].
- Дээрээс-доош тооцооллын арга (Top-down Estimation). Энэ арга нь алгоримтын болон алгоритмын бус аль аргад хэрэглэгддэг. Нийт зардлыг системийн хүрээнд гаргаад, улмаар тухайн зардлаа тухайн системийн бүрэлдэхүүн хэсэг, модулиудад хуваагддаг. Энэ аргыг програм хангамжийн хөгжүүлэлтийн эрт үед илүү ашигтай. Энэ аргын жишээ бол Putnam загвар юм.

• Дооройос-дээш тооцооллын арга (Bottom-up Estimation). Энэ нь өмнө дурдсан дээрээс-доош тооцооллын үнэлгээний эсрэг арга юм. Энэ арга нь програм хангамжийн модуль тус бүрийн зардлыг тооцож гаргаад улмаар тэдгээрийг нэгтгэж, нийт зардлыг гаргадаг. Системийн бүрэлдэхүүн хэсэг, дэд систем, модуль тус бүрийн үнэлгээг нарийвчлан тооцдог онцлогтой.

• Price-to-Win арга. Програм хангамжийн функциональ хамаарлаас илүүтэйгээр захиалагчийн төсөв дээр түлхүү анхаардаг арга. Програм хангамжийн нийт зардлыг хөгжүүлэлтийн үндсэн санал дээр харилцан тохиролцох замаар гаргадаг. Өөрөөр хэлбэл, програм хангамжийн хөгжүүлэлтийн зардлыг төслийн төсвөөр хязгаарлагддаг[6].

B. Алгориmтын аргууд

Алгоритмын аргууд нь математик болоод тооцооллын үндсэн дээр зардлыг тооцоолж гаргадаг. Зардал тооцоолох томьёо нь судалгааны үндэслэлтэй байж, оролт нь шугаман код, функцийн цэг, зардлын эх үүсвэр болох эрсдэлийн тооцоолол, програмчлалын хэл, аргачлалын дизайн зэргээс бүрддэг. Энэ төрлийн загваруудаас COCOMO, Putman, функцын цэг загвар, SEER-SEM загвар өргөн хэрэглэгддэг [3]. Тухайлбал:

COCOMO Ажлын өртөг загвар

Barry Boehm нарын гаргасан зардлын тооцоо, төслийн хөгжүүлэлтийн хугацааг зэрэг тооцдог түгээмэл хэрэглэгддэг арга бол COCOMO (ажлын өртөг загвар) юм. Энэ аргад ашигладаг параметр, томьёо нь өмнө нь хэрэгжүүлсэн програм хангамжийн төсөлтэй уялдаатай байдаг. Програмын кодын хэмжээг KLOC буюу мянган кодын шугамаар, харин хөдөлмөр/ажлыг хүн, сараар илэрхийлдэг. Boehm COCOMO загварын 3 төрлийг санал болгосон байдаг. Үүнд:

• Энгийн COCOMO (анхны загвар).

$$Хөдөлмөр = a*(KLOC)\ b \quad (1)$$

Үүнд: kloc нь кодын хэмжээ, a болон b нь тогтмол хэмжигдэхүүн болно. a болон b утгын онцлогоос (төслийн төрөл, үндсэн эсэх, хагас тусдаа эсвэл нэгдсэн гэх мэт) хамаарна.

• Дундын COCOMO.

$$Хөдөлмөр = a*(KLOC)b*\ EAF \quad (2)$$

энд хүчин чармайлал тохируулах хүчин зүйл нь EAF болно.

• Нарийвчилсан COCOMO. Энэ нь дэд систем тус бүр дээр ажилладаг болон нэгэн төрлийн дэд системүүдээр ерөнхий системдээ дэмжлэг болдог..

COCOMO II арга

COCOMO ажлын өртөг загварууд систем болон програмын шаардлага нь тогтмол, эсвэл урьдчилан тодорхойлогдоно гэж тооцоолдог. Гэвч мэдээллийн систем, програм хангамжийн төсөл үргэлж дээрх шиг байдаггүй. Уг загварын онцлогоос үүдэн 1990 онд өгөгдлийн багцын өргөн хүрээтэй COCOMO II загвар бий болсон. Энэ аргад шугаман код, функцийн цэг, объектын цэгүүдийг оролт болгон ашиглаж, өмнөх COCOMO загварын зардлын эх үүсвэр болох хүч чармайлтын үржүүлэлтийг өөрчилсөн. Гарсан үр дүн нь хэмжээ болон хүч чармайлтын тооцоогоор илэрхийлэгддэг бөгөөд түүнд үндэслэн төслийн хуваарийг боловсруулдаг[7].

Putnam загвар

Энэ загвар нь олон програмын төсөл судалдаг бөгөөд хүн хүчийн хуваарилалтаас хамаардаг.

$$S = E * хөдөлмөрt1/3*td4/3 \quad (3)$$

Үүнд: S нь програмын хэмжээг мөр тус бүрийг кодоор илэрхийлдэг (LOC/нэг мөр код)

Тухайн орчны хүчин зүйл нь E хөгжлийн чадавхийг илэрхийлдэг.

Td нь програм бэлэн болох хугацааг илэрхийлрх ба хөдөлмөр нь хүн жилээр тооцогддог.

Үүнээс гадна Putnam загварт нэг хамаарал олсон юм.

$$Хөдөлмөр = D0*td3 \quad (4)$$

Энд хүн хүч илэрхийлдэг DO нь шинэ болон дахин боловсруулсан програмд 8-27 хооронд хэлбэлздэг. Putnam загвар дээр үндэслэдэг SLIM гэх арга хэрэгсэл нь зардлын тооцоо болон хүн хүчний хуваарь гаргахад ашиглагддаг.

Функцийн цэг дээр үндэслэсэн шинжилгээ

Albrecht функцийн цэгийн шинжилгээг програм хангамжийн төслийн үйл ажиллагааг хэмжих зорилгоор хөгжүүлжээ[9]. Энэхүү аргачлал нь програмын хэмжээг тодорхойлж, дотоод логикийн файл, гадаад интерфэйс, гадаад оролт ба гаралтыг хэмждэг. Жишээлбэл, estimacs ба SPQR/20 бол функцийн цэгийн судалгаан дээр үндэслэсэн арга болно[8].

C. АРГУУДЫН ХАРЬЦУУЛАЛТ

Бид судалгааны ажлын хүрээнд дээр дурдсан аргуудын давуу ба сул талыг дараах байдлаар тодорхойлж байна. Хүснэгт 1.

ХҮСНЭГТ 1. АРГУУДЫН ДАВУУ БА СУЛ ТАЛУУД

№	Арга	Давуу тал	Сул тал
1	Адилтгалд суурилсан үнэлгээ	Өмнөх төслүүдийн өгөгдөл болон үнийн дүнгээс хамаарна. Үнэлгээчийн чадвараас зардлын тооцооллын оновчтой байдал хамаарна.	Шинжээчийг чухал хэрэгтэй мэдээллээр хангах төслийг тодорхой шинж чанараар тодорхойлох шаардлагатай байдаг. Төсөл бүр дээр ашиглах боломжгүй.
2	Мэргэжилтний үнэлэмжийн арга	Мэргэжилтний таамаглалаас хамаарч технологи, архитектур, програмчлалын хэлийг ашиглана. .	Баримтжуулахад төвөгтэй. Үнэлгээчин, шинжээчид бүрэн найдаж болохгүй.

3	Дээрээс-доош тооцооллын арга	Төслийн талаар дэлгэрэнгүй мэдээлэл шаарддаггүй. Хэрэгжүүлэхэд хялбар, хурдан. Үйл ажиллагаанд төвлөрдөг.	Жижиг хэмжээний асуудал, системийн зардлын нэмэгдэл, нийлмэл асуудлыг харгалзан үздэггүй.
4	Доороос-дээш тооцооллын арга	Тооцооллын найдвартай байдал өндөр.	Баримт бичгийн боловсруулалттай холбоотой зардал зэрэг системийн түвшний үйл ажиллагаанд анхаардаггүй. Цаг хугацаа их шаарддаг.
5	COCOMO загвар	Тооцоолоход хялбар.	Програм хангамжийн хөгжүүлэлтийн эхэн үед хэрэглэдэг учир тооцооллын алдаа гарах магадлалтай.
6	COCOMO II	Үйлдвэрийн стандарт загвар. Үйл явц нь тодорхой, хялбар тохируулагддаг.	Жижиг хэмжээний төслийн үргэлжлэх хугацааны тооцоолол үндэслэлгүй гарах нь бий.
7	Putnam загвар	Цаг хугацаа, хэмжээ гэсэн хувьсах хоёр хүчин зүйлд үндэслэгддэг.	Програм хангамжийг хөгжүүлэх амьдралын циклын бусад асуудлыг авч үздэггүй.
8	Функцийн цэгийн шинжилгээ	Програмчлалын хэл, хэрэгсэл, хэрэгжүүлэх аргаас хамааралгүй. Хөгжүүлэлтийн зардлыг програм хангамжийн хөгжүүлэлтийн эхэн үе шатанд тооцоолж болно.	Цаг хугацаа их шаарддаг, гар ажиллагаа ихтэй. Функцийн цэгийн хэрэглээг баримжаалахад туршлага шаарддаг.

Програм хангамжийн зардлыг тооцоолох хамгийн сайн, шалгарсан нэг арга гэж байдаггүй. Иймээс нөхцөл байдалд нийцүүлэн тооцооллын аргыг оновчтой сонгон ашиглах нь чухал. Практикт тооцооллын хэд хэдэн аргыг хослуулан ашиглах нь зардлын тооцоог илүү оновчтой гаргадаг.

II. ФУНКЦИЙН ЦЭГИЙН ШИНЖИЛГЭЭ

Програм хангамжийн хөгжүүлэлтийн хугацаа тооцож гаргах нь практик туршлагаас хамаарна гэж үзэх нь түгээмэл. Гэвч ямар нэгэн зүйлийг хэмжээг тооцоход тодорхой нэгжийг гаргах хэрэгтэй ба уг нэгж дээр тулгуурлан нийт хэмжээг гаргаж болно. Програм хангамжийг хөгжүүлэх төслийн хувьд программыг хөгжүүлэхэд хэмжих нэгжийг функцийн цэг гэж нэрлэнэ. Үүнийг 1979 онд Аллен Алберт програм хангамжийг үр ашигтай хөгжүүлэхэд тооцох нэгж болгон анх гаргаж ирсэн. 2003 онд ISO стандарт болж батлагдсан[9].

Хөгжүүлэх програм хангамжийн хэрэглэгчийн функциональ шаардлага дээр тулгуурлан системийн функцуудыг үндсэн 5 хэсэгт хуваадаг. Үүнд:

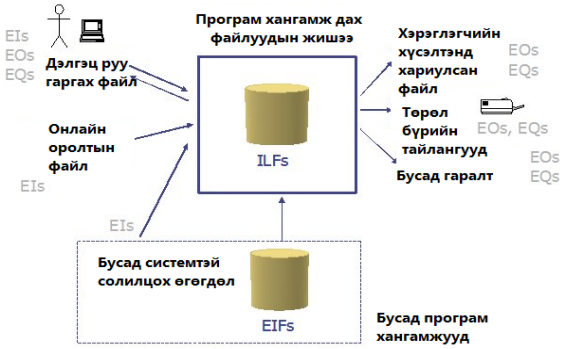
1) Програм хангамжийг хөгжүүлэхэд бичигдэж байгаа функцуудыг үндсэн 2, 5 дэд ангилалд хуваадаг.

Өгөгдлийн функцууд:

1. Дотоод логик файл (ILF, Internal logical files)
2. Гадаад интерфэйсийн файл (EIF, External interface files)

Транзак функцууд:

3. Гадаад оролт (EI, External Inputs)
4. Гадаад гаралт (EO, External Outputs)
5. Гадаад шаардлагатай шилжүүлэг (EI, External Inquiries)



Зураг 1. Програм хангамжийн файлуудыг функцийн цэгээр ангилах

Шинжилгээ хийх аргачлал

Дотоод логик файл нь хэрэглэгчээс системд дамжуулж байгаа өгөгдлийн файлууд ихэвчлэн ордог. Жишээ нь:

1. Холбоост өгөгдлийн сангийн хүснэгтүүд.
2. Гол өгөгдлийн файлууд.
3. Хэрэглэгчийн тохиргооны файл
4. LDAP-н өгөгдлийн файлууд.

Ийм төрлийн хэдэн ширхэг файл байгааг тоолон Хүснэгт 2-т өгөгдсөн өгөгдлийн дагуу харгалзах оноог тооцно.

ХҮСНЭГТ2. ӨГӨГДЛӨӨС ХАМААРУУЛАН ILF ФАЙЛЫГ АНГИЛАХ

Rets-Бүлэг өгөгдөл	**Data element types (dets)-Өгөгдлийн элементийн төрөл**		
	1-19	20-50	51+
1	Б	Б	Д
2 - 5	Б	Д	Б
6 - дээш	Д	И	И

Үүнд,

- RETS (Record Element Type) гэдэг нь ILF болон EIF файлд агуулагдаж байгаа хэрэглэгчийн тодорхойлсон бүлэг өгөгдөл болно. Жишээ нь объект.
- DETS (Data Element Type) нь хэрэглэгчийн тодорхойлсон өгөгдлийн төрөл болно. Жишээ нь int x, class person гэх мэт.
- Б-Бага, Д-Дундаж, И-Их

ХҮСНЭГТ 3. ILF-Н ИХ, БАГА, ДУНДАЖ ТҮВШИНД ОНООХ ЖИН

Түвшин	Оноо
Бага	7
Дундаж	10
Их	15

Гадаад интерфайсын файлууд-EIF(External interface files)Программаас хэрэглэгчид зориулан гаргаж байгаа интерфайсын файлууд

ХҮСНЭГТ 4. ӨГӨГДЛӨӨС ХАМААРУУЛАН EIF ФАЙЛЫГ АНГИЛАХ

Rets- Бүлэг өгөгдөл	Data element types (dets)- Өгөгдлийн элементийн төрөл		
	1-19	20-50	51+
1	Б	Б	Д
2 to 5	Б	Д	И
6 -их	Д	И	И

ХҮСНЭГТ 4. EIF –Н ИХ, БАГА, ДУНДАЖ ТҮВШИНД ОНООХ ЖИН

Түвшин	*Оноо*
Бага	5
Дундаж	7
Их	10

гэх мэтчилэн ангилал тус бүрийн түвшинг тодорхойлж Function point шинжилгээг Unadjusted FP Count-н тоог гаргана.

Function point шинжилгээний 4-р алхам нь Value Adjustment Factor-г гаргах ба 14 хүчин зүйлийг тодорхойлж өгсөн.

ХҮСНЭГТ 5. НӨЛӨӨЛЛИЙН ЗЭРЭГТ АВЧ ҮЗЭХ 14 ХҮЧИН ЗҮЙЛ

1	Найдвартай нөөцлөх болон сэргээх боломжтой байх
2	Мэдээлэл солилцох боломжтой
3	Тархсан функцуудтэй
4	Гүйцэтгэл өндөртэй
5	Тохируулга ихтэй
6	Олон транзакцитай
7	Онлайн өгөгдөл дамжуулах
8	Нийлмэл үйл ажиллагаатай
9	Дахин ашиглагдах боломжтой
10	Хялбар суурилуулалт хийгддэг
11	Ашиглагдахад хялбар
12	Олон интерфэйстэй
13	Өөрчлөх боломж хялбар
14	Олон суурилуулалтай

Эдгээр 14 хүчин зүйлийг өөрийн хөгжүүлж байгаа системд байгаа эсэхээс хамаарч 0..5 оноо өгч жинлэнэ.Нийт нөлөөллийн зэргийг гаргана.

$$VAF = (НН3*0.01) + 0.65 \quad (5)$$

Үүнд, НН3-Нийт нөлөөллийн зэрэг

Adjusted FP Count буюу Тохируулгын функцын тоог Function Point-функцын цэгийн тохируулгын тоог томьёо (6) ашиглан олно.

Тохируулгын функцыг цэгийн тоо =
тохируулагдаагүй функцын цэгийн тоог * VAF (6)

VAF (Value Adjustment Factor)-нөлөөллийн хүчин зүйлийн оноо

Олон улсын туршлагаас харахад VAF 1 оноо нь ажлын 2 өдрөөр тооцогдож, програм хангамжийг хөгжүүлэх хугацааг гаргаж ирнэ.

ДҮГНЭЛТ

Судалгааны хүрээнд авч үзсэн арга, техникийн давуу болон сул талаас харахад шилдэг шалгарсан арга байхгүй нь харагдаж байна. Дээрх аргуудын давуу болон сул талууд хоорондоо уялдаа холбоотой учраас тэдгээрийг нэгтгэж ашигласнаар аль нэг сул талаас сэргийлэх боломжтой. Ингэснээр аль нэг арга, техникийн сөрөг нөлөөг бууруулж, тус бүрийн давуу талыг бий болгоно. Тооцоолох аргуудыг хооронд нь харьцуулах боломжтой. Тооцоолох аргыг сонгохдоо тодорхой төслийг алгоритмын бус аргаар баримжаалан тооцох нь илүү үр дүнтэй. Харин том болон тодорхой бус төслийг алгоритмын аргачлалаар тооцоолох нь зүйтэй. Алгоритмын загваруудаас COCOMO II нь COCOMO загвараас илүү байдаг нь зөвхөн шугаман кодод хязгаарлагдах бус, функцийн цэг, объектын цэг ашиглаж програмын төслийн хэмжээг баримжаалах боломжтой холбоотой.

Програм хангамж хөгжүүлэхэд гарах зардлыг урьдчилан таамаглах нь хүндрэлтэй ажиллагаа юм. Програм хангамжийн төслийн төлөвлөгөө, санхүүгийн тооцоо, зардлын тооцоотой уялдаатай тул програмын хэмжээг тооцоолох нэн чухал. Төсөл эхлэхээс өмнө зардлын тооцоолол гаргасан тохиолдолд төслийн хязгаарлагдмал нөөцөөр хэрэгжүүлж болох үзүүлэлтүүдийг илрүүлж, эрсдэлийг бууруулах давуу талтай. Ингэснээр ерөнхий зардлын тооцоо нь програмын төслийн хөгжүүлэх хугацаа, төслийн хэрэгжилтэд эерэг нөлөө үзүүлдэг. Хямд өртгөөр сайн чанарын програм хангамжийг хөгжүүлэх зорилгод хүрэхийн тулд нөхцөл байдалд нийцсэн, оновчтой арга ашиглан зардлыг зөв тооцоолох нь чухал. Энэ зорилгоор хэд хэдэн арга техникийг хослуулан ашиглах нь зүйтэй гэсэн дүгнэлтэд хүрч байна.

НОМ ЗҮЙ

[1] [International Journal of Computer Applications (0975 – 8887) Хуудас 141 – No.11, May 2016

[2] Khatibi Bardsiri,V.,D.N.A.Jawawi,S.Z.M Hashim,and E.Khatibi,"Increasing the accuracy of software development effort estimation using projects clustering",IET Software, 2012.

[3] Patil,Lalit V.,Rina M.Waghmode, S.D.Joshi,and V.Khanna, "Generic model of software cost estimation:A hybrid approach",2014 IEEE International Advance Computing Conference (IACC), 2014.

[4] Boehm,"Software Engineering Economics", Prentice Hall ,1981.

[5] Gupta,Syona,Geeta Sikka,and Harsh Verma,"Recent methods for software effort estimation by analogy", ACM SIGSOFT Software Engineering Notes, 2011.

[6] Keung.J.W.,B.A. Kitchenham,et al.”Analogy-X: Providing Statistical Inference to Analogy based Software Cost Estimation”. Software Engineering, IEEE Transaction on 34(40:471-484,2008.

[7] M.Jorgensena, “Review of studies on Expert Estimation of Software Development Effort”, Journal of Systems and Software,70(1-2):37-60, 2004.

[8] Yinhuan,Z., W.Beizhan,et al. “Estimation of software projects efforts based on function point”,Computer Science & Education. ICCSE ,4th International Conference, 2009.

[9] Mustafa,K.K Gowthaman, and R.A.Khan,”Measuring the Function Points for Migration Project: A Case Study”, American Journal of Applied Sciences, 2005.