Архитектур Загвар, дизайн Загвар, ба Объектууд

ROBERT T. MONROE, ANDREW KOMPANEK, RALPH MELTON, DAVID GARLAN

Карнеги Меллон Их Сургууль

Архитектурын хэв маяг, объект хандалтат дизайн, дизайны загварууд нь системийн дизайны мэдлэгийг олж авах, ашиглах замаар програм хангамжийн дизайн, дахин ашиглах аргыг хялбаршуулдаг арга барил гэж амлаж байна. Энэ нийтлэлд янз бүрийн аргуудын чадвар, үүрэг, давуу тал, хязгаарлалтыг судлах болно.

Software систем бүтээгчид шинэ системийн инженерчлэлд дизайны мэдлэгийг ашиглахын чухлыг улам бүр хүлээн зөвшөөрдөг. Хэд хэдэн ялгаатай боловч холбоотой арга барилууд амлалт өгдөг.

Нэг арга нь системийн дизайны архитектурын түвшинд анхаарлаа хандуулах явдал юм - харилцан үйлчлэлийн хэсгүүдийн нэгдэл болох системийн нийт бүтэц. Архитектурын дизайн нь масштаб болон зөөвөрлөх чадвар, дизайны элемент үүдэд функциональ хуваарилалт, элемент үүдийн хоорондын харилцан үйлчлэлийн протокол, боловсруулалтын хурд, төгсгөл хүртэлх хүчин чадал, гүйцэтгэлийн ерөнхий чадвар зэрэг дэлхийн системийн шинж чанаруудыг гэрэлтүүлдэг.1 Архитектурын тодорхойлолтууд нь албан бус, өвөрмөц синкрат байх: хайрцаг ба шугаман диаграммууд нь системийн үндсэн бүтцийг илэрхийлж, бэлгэдлийн утгыг тайлбарласан зохиолын хамт. Гэсэн хэдий ч тэдгээр нь систем нь түүний үндсэн шаардлагыг хангаж чадах эсэхийг тодорхойлох чухал үе шатыг өгдөг бөгөөд системийг бий болгоход хэрэгжүүлэгчдэд чиглүүлдэг.

Сүүлийн үед архитектурын тодорхойлолтыг дизайны мэдлэгийг кодчилох, дахин ашиглахад ашигладаг болсон. Тэдний хүч чадлын ихэнх нь "үйлчлүүлэгч серверийн систем", "давхаргатай систем" эсвэл "хар самбарын зохион байгуулалт" гэх мэт архитектурын хэлц үг хэллэгийг ашигласнаас үүдэлтэй.

0740-7459/97/$10.00 © 1997 IEEE

Эдгээр нь албан бус бол өргөн тархсан байдаг
тайлбарын талаарх ойлголт ба
инженерүүд хурдан холбоо бариарай

Арга тус бүр байдаг санал болгох зүйл: ийн цуглуулга төлөөллийн загварууд болон механизмууд.

тэдний дизайныг бусдад. Ийм архитектурын хэлц
үгс нь урьд өмнө байсан зүйлийг илэрхийлдэг
архитектурын хэв маяг гэж нэрлэдэг.2

Объект хандалтат парадигм
дүрслэх өөр аргыг санал болгож байна
системийн загварууд. Хамгийн энгийнээр хэлбэл,
Объект хандалтат дизайн нь хожуу өгөгдөл, зан
төлөвийг салангид хэлбэрээр багтаах боломжийг бидэнд олгодог
тодорхой интерфейсээр хангадаг объектууд
бусад объект руу; объектын булгүүд
хооронд мессеж дамжуулах замаар харилцах
өөрсдөө. OOD болох нь батлагдсан
Практикт нэлээд алдартай бөгөөд нарийн төвөгтэй
OOD аргачлалууд нь зөөлөн эдлэлийг зохион
бүтээхэд ихээхэн хэмжээний хөшүүргийг санал
болгодог, үүнд задрахад хялбар байдаг.
тогтолцоог бүрдүүлэгч элементүүдэд нь
болон хуваах системийн үйл ажиллагаа
мөн эдгээр элементүүдийн дундах харилцлага.
Гэсэн хэдий ч энэ нь өөрөө сайн биш юм
бүлэг объектуудын хооронд нарийн төвөгтэй
харилцан үйлчлэлийг дүрслэхэд тохиромжтой.
Үүний нэгэн адил, хэдийгээр бие даасан объектууд
ихэвчлэн бусад хэрэглүүрүүдэд дахин ашиглах,
барьж авах, дахин ашиглах боломжтой
Олон төрлийн объектыг агуулсан дизайны нийтлэг
хэлц үгс хэцүү байж болно.

Загварын хэв маяг нь болсон
д зориулсан сонголт улам бүр түгээмэл болж байна
OOD ийн хязгаарлалтыг шийдвэрлэх. Хэдийгээр
үндсэн зарчмууд
загварын хэв маяг нь угаасаа холбоотой биш юм
OOD, энэ талаар сүүлийн үеийн олон ажил
талбай нь дизайны хэв маяг гт анхаарлаа хандуулсан
объект зохиох.4,5 Архитектурын хэв маяг гийн
нэгэн адил дизайны загвар нь зааварчилгаа өгдөг
дизайны элементүүдийг хослуулах зориулалттай
зарчимтай, батлагдсан арга замууд.

Эдгээр нь тус бүр нь ихэвчлэн нэмэлт байдаг
програм хангамжийг барьж авах арга
дизайны медлэг, програм хангамжийн дизайн
өөрсдөө давуу талтай, сул талтай. Эдгээр
программуудыг үр дүнтэй ашиглахын тулд бид
тэдгээрийг ойлгох хэрэгтэй
нэр томьёо, чадвар, ижил төстэй байдал,
болон ялгаа. Цаашлаад бид хийх хэрэгтэй
хүн бүрийн гүйцэтгэж чадах үүргийг ойлгох
амжилттай програм хангамжийн дизайн.

Програм хангамж гэж юу вэ архитектурийн дизайн гэж?

Практикт архитектурын зураг төсөл
үндсэн хоёр үүргийг гүйцэтгэдэг. Нэгдүгээрт,
энэ нь программ хангамжийн системийн дизайнеруудын
бодож болохуйц хийсвэрлэлийн түвшинг өгдөг
системийн зан байдал: функц, гүйцэтгэл,
найдвартай байдал гэх мэт. Хийсвэрлэх замаар
хэрэгжилтийн нарийн ширийн зүйлээс хол, а
архитектурын сайн дүрслэл нь а
системийн дизайн нь оюуныэрч хүчтэй
мөн хамгийн чухал шинж чанаруудыг илчилдэг
түүний амжилтанд. Энэ нь ихэвчлэн тодорхойлоход
ашигладаг гол техникийн баримт бичиг юм
санал болгож буй шинэ систем нь хэрэгжих эсэх
түүний хамгийн чухал шаардлагыг хангах.

Хоёрдугаарт, архитектурын зураг төсөл
системийн "ухамсар" болж үйлчилдэг
хөгжихийн хэрээр. Системийн дизайны чухал
таамаглалыг тодорхойлсноор сайн
Архитектурын дизайн нь үйлявцыг удирдан чиглүүлдэг
системийн сайжруулалтын тухайзааж байна
системийн яамар талууд амархан байж болох вэ
системийг алдагдуулахгүйгээр өөрчилсөн
бүрэн бүтэн байдал. Барилгын зураг төслийн нэгэн адил, а
сайн баримтжуулсан архитектурын зураг төсөл
Энэ нь програм хангамжийн "даацын хана"г ыг
тодорхой болгож байгаа нь тус болохгүй
зөвхөн дизайны үед төдийгүй бүхэлдээ
системийн амьдралын мөчлөг. Цаг хугацаа өнгөрөхөд
олон үүрэг хариуцлагаа биелүүлэхийн тулд архитектурын
тайлбар хангалттай энгийн байх ёстой
системийн түвшний үндэслэл, таамаглалыг
зөвшөөрөх; практикийн хувьд ийм байх ёстой
нэг эсвэл хоёр хуудсанд багтах. Үүний үр дүнд ийм байна
ихэвчлэн шаталсан: хийсвэрлэлийн нэг түвшинд
атомын архитектурын элементүүд
ихэвчлэн илүү дэлгэрэнгүй тайлбарласан байдаг
архитектур нь доод түвшинд байна.

Архитектурын тодорхойлолт нь дараахь зүйлд
голчлон хамаатай
үндсэн асуудлууд:

Системийн бүтэц. Архитектур
Тодорхойлолт нь системийн онцлог шинж юм
өндөр түвшний тооцооллын элемент үүд болон
тэдгээрийн харилцан үйлчлэлийн хувьд бүтэц.
Өөрөөр хэлбэл, архитектур нь дизайныхаа хүрээг бүрдүүлдэг
харилцан үйлчлэлийн тохиргоо болгон шийдэл
бүрэлдэхүүн хэсгүүд. Энэ нь ялангуяа тухай биш юм
шаардлага (жишээлбэл, асуудлын талбарын
элемент үүдийн хооронд хийсвэр харилцаа) болон
хэрэгжилтийн дэлгэрэнгүй мөдээллэл
(алгоритм эсвэл өгөгдлийн бүтэц гэх мэт).
харилцан үйлчлэлд зориулсан бая лаг хийсвэрлэлүүд.

Архитектурын хооронд ын харилцан үйлчлэл

Бүрэлдэхүүн хэсгүүд нь ихэвчлэн холбогч шугам
хэлбэрээр дүрслэгдсэн байдагбая лаг үгсийн санг бүрдүүлдэг
системийн дизайнерууд. Хэдийгээр харилцан үйлчлэл
Процедурын дуудлага эсвэл
хуваалцсан өгөгдлийн хувьсагчид ихэвчлэн
илгээсэн илүү төвөгтэй мэдээгтуудыг төлөөлдөг. Жишээ
хоолой оруулах (конвенц той
файлын төгсгөлтэй ажиллах, блоклох),
үйлчлүүлэгч серверийн харилцан үйлчлэл (дүрмээр
эхлүүлэх, эцэслэх, болон
онцгой тохиолдлыг зохицуулах), үйлявдлын нэвтрүүлэг
холболтууд (олон хүлээн авагчтай),
ба мөдээллийн санд хандах протоколууд (хамт
гүйлгээг дуудах протоколууд).

Глобал шинж чанарууд. Архитектур
Дизайнууд нь ерөнхийдөө системийн үйл ажиллагааг
тодорхойлдог. Тиймээс тэдэнд асуудал гардаг
хаяг нь ихэвчлэн системийн түвшинд байдаг,
тухайлбал, төгсгөл хүртэлх өгөгдлийн хурд болон
хоцрогдол, нэг хэсгийн уян хатан байдал
систем нь нөгөөд нь бүтэлгүйтэх, эсвэл нэг нь
системд гарсан өөрчлөлтийн тархалт
системийн нэг хэсэг нь өөрчлөгдсөн (жишээ нь
дээр суурилсан платформыг өөрчлөх
систем ажилладаг).

Архитектурын хэв маяг

Аливаа дизайны үйл ажиллагааны нэгэн адил төв
Асуулт бол илүү сайн загвар гаргахын тулд
өнгөрсөн туршлагыг хэрхэн ашиглах явдал юм.
Одоогийн түрээсийн практикт архитектурын зураг төсөл
кодчилсон болон primari дахин ашигласан байна

ПРОГРАММЫН АРХИТЕКТУРИЙН ТОДОРХОЙЛОЛТ ХЭЛНҮҮД

Програм хангамжийн архитектуруудад архитектурын дизайныг тодорхойлох, тайлбарлах тэмдэглэгээг өгөх зорилгоор архитектурын дизайны төрөл бүрийн хэлүүдийг бий болгосон. ADL-ууд анхаарлаа хандуулдаг архитектурын дизайныя нз бүрийн талууд, тэдгээрийн дүн шинжилгээ Тусламж нь албан бусаас маш албан ёсны хүртэл я нз бүр байдаг. Энд зарим жишээ байна:

UniCon систем1 нь архитектурын тодорхойлолт, модулиудыг гүйцэтгэгдэх код болгон эмхэтгэхэд чиглэдэг.

Rapide2 нь зан үйлийн онцлог, архитектурын дизайны загварчлалыг чухалчилдаг.

Wright3 нь бүрэлдэхүүн хэсгийг тодорхойлох албан ёсны үндэслэлийг өгдөг харилцан үйлчлэл (холбогчоор дамжуулан) болон архитектурын хэв маяг.

Aesop System4 нь тодорхой кодчилол, хэрэглээг дэмждэг өргөн хүрээний архитектурын хэв маяг.

Домэйн тусгай программ хангамжийн архитектурын төрөл бүрийн хэл5 тодорхой хэрэглээний домэйнд тохирсон архитектурын тодорхойлолтыг дэмжих.

Дээр дурдсан ADL-ээс гадна

програм хангамжийн архитектурыг тайлбарлах зорилгоор тусгайлан боловсруулсан, хэд хэдэн ерөнхий албан ёсны техникийн хэлүүд бас байдаг ашигласан. Жишээ нь Z,6 Communicating Sequential Процесс,7 ба Химийн абстракт машин.8

Програм хангамжийн архитектурын судалгааны нийгэмлэг ухамсарлаж байна Эдгээр тэмдэглэгээ нь ихээхэн давхцаж байгаа, я лангуя а програм хангамжийн архитектурын бүтцийн талыг хүндэтгэх тодорхойлолт. ACME бол шинээр гарч ирж буй ерөнхий архитектур ком я нз бүрийн ADL болон архитектурын дизайныг хооронд нь өөрчлөхөд зориулагдсан тайлбар хэл багаж хэрэгсэл.9

Архитектурын бүдүүвчийг илэрхийлэхэд ашигладаг тэмдэглэгээ Энэ нийтлэлийн жишээн дээх хэв маяг гийн тодорхойлолтууд нь эдгээр архитектурт түгээмэл байдаг нэр томьёо, тэмдэглэгээг тусгасан болно тайлбар хэлүүд.

Ашигласан материал

1. M. Shaw et al., “Abstractions for Software Architecture and Tools to support them,” IEEE Trans. Software Eng., 1995 оны 4-р сар, хуудас 314-335.
2. DC Luckham et al., “Specification and Analysis of System Architecture using Rapide,” IEEE Trans. Software Eng., 1995 оны 4-р сар, хуудас 336-355.
3. Р.Аллен, Д.Гарлан, “Архитектурын холболтыг албан ёсны болгох,” Proc. 16 дахь олон улсын конф. Software Eng., IEEE Computer Soc. Пресс, Лос Аламитос, Калифорниа, хуудас 71-80.
4. Д.Гарлан, Р.Аллен, Ж.Окерблум, “Архитектурын дизайны орчин дахь хэв маяг гийг ашиглах нь” Proc. SIGSOFT '94, ACM Press, Нью-Йорк, 1994, хуудас 179-185.
5. W. Tracz, “DSSA Frequently Asked Questions,” Software Eng. Тэмдэглэл, 1994 оны 4-р сар, хуудас 52-56.
6. JM Spivey, The Z Notation: A Reference Manual, Prentice-Hall, Englewood Cliffs, NJ, 1989.
7. CAR Hoare, Communicating Sequential Processes, Prentice-Hall, Englewood Cliffs, NJ, 1985.
8. П.Инверарди ба А.Волф, “Химийн хийсвэр машины загварыг ашиглан програм хангамжийн архитектурын албан ёсны тодорхойлолт ба дүн шинжилгээ,” IEEE Trans. Software Eng., 1995 оны 4-р сар, хуудас 373-386.
9. Д.Гарлан, Р.Т.Монро, Д.Уайл, “ACME: Architecture Тодорхойлолт ба харилцан солилцох хэл” техник. тайлан, Карнеги Меллон их сургууль, Питтсбург, 1996 он.

албан бус дамжуулах замаар архитектурын хэлц үгс. Тухайлбал, А системийн архитектурыг тодорхойлж болно албан бусаар үйлчлүүлэгч-серверийн систем, а самбарын систем, дамжуулах хоолой, интерпритер эсвэл давхаргат систем. байхад Эдгээр шинж чанарууд нь буруу тодорхойлолтод ховорхон байдаг, тэд маш их зүйлийг илэрхийлдэг системийн бүтэц, суурь тооцоолльн загварын тухай.

Архитектурын чухал анги

Хэлц үгс нь зарим судлаачдын архитектур гэж нэрлэдэг зүйлийг бүрдүүлдэг.

хэв маяг. Архитектурын хэв маяг гийн дүрнь хоорондоо холбоотой системүүдийн гэр бүлийг илэрхийлдэг бий болгох я вдал ном олон хэв маяг гийг дэмждэг хуваалцсан бүтцийн болон семантикаар шинж чанар.2 Архитектурын хэв маяг нь дизайны түгээгдэхүүнээр хангадаг системүүдийн тодорхой ангилал. Тодруулбал, хэв маяг нь ихэвчлэн дараахь зүйлийг өгдөг дөрвөн зүйл:

Дизайн элементүүдийн толь бичиг: бүрэлдэхүүн хэсэг болон холбогч төрөл гэх мэт хоолой, шүүлтүүр, үйлчлүүлэгч, сервер, задлагч, болон мэдээллийн сан.

Загварын дүрэм, эсвэл хязгаарлалт, тэр тэдгээрийн аль найрлагыг тодорхойлох элементүүдийг зөвшөөрдөг. Жишээлбэл, дүрэм нь мөчлөгийг хориглож болно тусгай хоолойн шүүлтүүрийн хэв маяг гийг зааж өгнө үү үйлчлүүлэгч-сервер байгууллага байх ёстой n-to-one хамаарал, эсвэл тодорхой найрлагын хэв маяг гийг тодорхойлох, жишээлбэл а хөрвүүлэгчийн шугаман задрал.

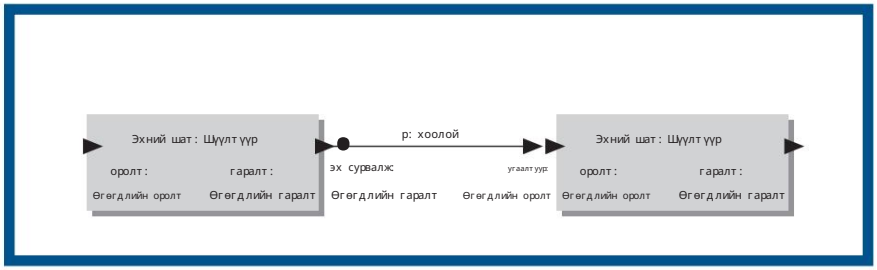
Семантик тайлбар, үүгээр Загварын дүрмээр хязгаарлагдсан дизайны элементүүдийн найрлага; нарийн тодорхойлсон утгатай байна.

Гүйцэтгэх боломжтой дүн шинжилгээийм хэв маяг гаар бүтээгдсэн системүүд дээр. Жишээ хуваарийн шинжилгээг багтаасан а хэв маяг нь бодит цагийн боловсруулалт болон мухардлыг илрүүлэхэд чиглэгдсэн үйлчлүүлэгч-сервер мессеж дамжуулж байна. Ан Шинжилгээний чухал онцгой тохиолдол бол системийг хэрэглээний генераторууд (жишээлбэл, задлагч генераторууд), эсвэл дахин ашиглахад хүргэдэг тодорхой хуваалцсан хэрэгжүүлэх суурь (хэрэглэгчийн интерфейсийн хүрээ болон хоорондын харилцааны дэмжлэг тарксан процессууд).

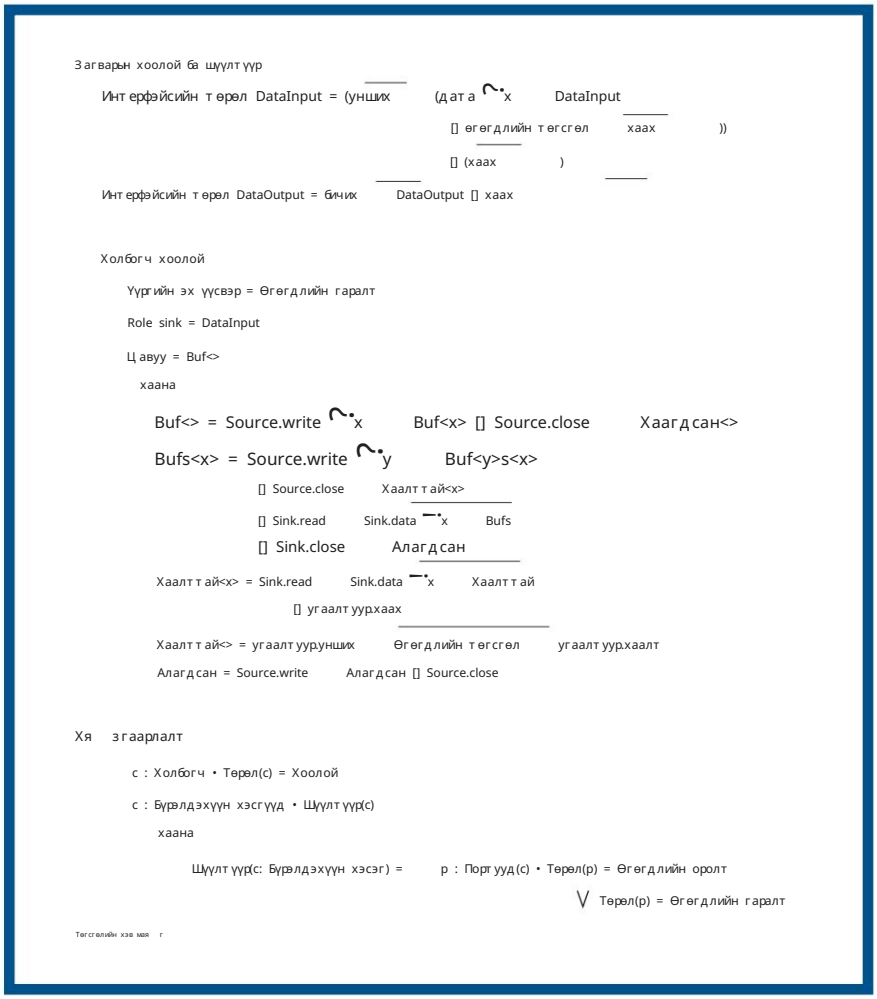
Архитектурын хэв маяг гийн хэрэглээ нь хэд хэдэн чухал ашиг тус. Нэгдүгээрт, тэр дизайныг дахин ашиглахыг дэмждэг: сайн ойлгогдсон шинж чанартай ердийн шийдэл шинэ асуудлуудад дахин хэрэглэж болно итгэл. Хоёрдугаарт, энэ нь кодыг дахин ашиглахад хүргэдэг: ихэвчлэн өөрчлөгддөггүй

Архитектурын хэв маяг гийн талууд өөрсдөө хуваалцсан хэрэгжилт. Жишээлбэл, а-д тодорхойлсон системүүд хоолой шүүлтүүрийн загвар нь даалгаврыг шийдвэрлэхийн тулд Unix үйлдлийн системийн командуудыг дахин ашиглаж болно хуваарь гаргах, синхрончлох, дамжуулах хоолойгоор дамжуулан харилцах. Үүний нэгэн адил, А үйлчлүүлэгч-серверийн хэв маяг гийг ашиглах боломжтой одоо байгаа RPC (алсын журам дуудлага) механизм ба stub үүсгэх чадварууд. Гуравдугаарт, хэрэв уламжлалт бүтэц тэй бол бусад хүмүүс системийн зохион байгуулалтыг ойлгоход хялбар болно.

ашигласан. Жишээлбэл, өгөхгүйгээр ч гэсэн гэж системийг тодорхойлдог дэлгэрэнгүй мэдээлэл үйлчлүүлэгч-серверийн байгууллага нэн даруй төрлийн хүчтэй дүр төрхийг илэрхийлдэг одоо байгаа хэсгүүд болон тэдгээрнь хооронд оо хэрхэн нийцэж байгаа. Дөрөвдүгээрт, стандартчилагдсан хэв маяг гийг ашиглах харилцан ажиллах чадварыг дэмждэг. Жишээ CORBA объект хандалтат архитектур, OSI (Нээлттэй систем Харилцан холболт) протоколын стек, ба үйлявдалд суурилсан хэрэглүүдийг нэгтгэх. Тавдугаарт, гэх мэт хязгаарлах замаар бид өмнө нь тэмдэглэсэн дизайны орон зай, архитектурын хэв маяг г ихэвчлэн тусгайлсан, тодорхой хэв маяг гийг зөвшөөрдөг шинжилгээ. Жишээлбэл, бид дүн шинжилгээ хийж болно зорилгатын хоолойн шүүлтүүрийн хэв маяг гаар бүтээгдсэн системүүд дамжуулах чадвар, хоцрогдол ба түүнээс чөлөөлөгдөх мухардмал, гэхдээ а ашигладаг өөр системийн хувьд энэ нь ахулгай биш байж магадгүй ном өөр хэв маяг г эсвэл дурын, түүрзуурын архитектур.



Зураг 1. Хоолой ба шүүлтүүрийн хэв маяг гийн энгийн системийг архитектурын тэмдэглэгээг ашиглан тодорхойлсон.



Зураг 2. Зураг 1-д үзүүлсэн системийг Wright архитектурын тайлбар хэлийг ашиглан энд зааж өгсөн болно.

ОБЪЕКТ ЧИГЛЭЛТТЭЙ ДИЗАЙН БОЛОН ПРОГРАММЫН АРХИТЕКТУР

Объект хандалтат дизайн para digm нь өөр хийсвэрлэлийг өгдөг програм хангамжийн дизайн. Хамгийн энгийнээр хэлбэл, OOD нь системийн дизайнеруудад тусгаарлагдсан өгөгдөл болон зан төлөвийг салангид байдлаар багтаах боломжийг олгодог тодорхой интерфейсээр хангадаг объектууд бусад объект руу. Мессеж дамжуулдаг хийсвэрлэлийг цавуу болгон ашигладаг объектуудыг холбож, дизайн дахь харилцааны сувгуудыг тодорхойлдог.

Хэдийгээр OOD ойлголтыг ашиглаж болно зарим архитектурын дизайныг шийдвэрлэх асуудлууд, мөн үүнийг хийх нь хүмүүсийн дунд түгээмэл байдаг Програм хангамж хөгжүүлэгчдийн хувьд объект хандалтат програмын чадвар, ашиг тусын хоорондмэдэгдэхүйц ялгаа байдаг. дизайн хийх арга барил, шинээр гарч ирж буй хүмүүсийн өгсөн арга замууд програм хангамжийн архитектур дизайны анги хэрэгсэл ба тэмдэглэгээ. Дараахь байдлаар Жишээ нь, програм хангамжийн архитектурын үзэл баримтлал нь архитекторт хийхийг зөвшөөрдөг олон, бая лаг интерфэйсийг тайлбарлах а бүрэлдэхүүн хэсэг ба бүрэлдэхүүн хэсгийн сулатын нийлмэл протоколуудыг тайлбарлах, хаах тайлбарлахад хэцүү харилцан үйлчлэл уламжлалт объект хандалтат ойлголт, тэмдэглэгээг ашиглах.

Янз бүрийн чадварыг харуулахын тулд хэв маяг гт суурилсан програм хангамжийн архитектурын дизайн ба сүүлийн үеийн объектод чиглэсэн дизайны хувьд 1-ээс 5-р зурагт үзүүлсэн энгийн системийг авч үзье.

Зураг 1 ба 2-т нийтлэг архитектурын тэмдэглэгээг ашигладаг (хуудас дээрх архитектурын тайлбар хэл дээрх хайрцагласан текстийг үзнэ үү. 45) архитектурын үзэмжийг танилцуулах систем. 3-аас 5-р зурагт тайлбарлав -ийн аажмаар илүү боловсронгуй хувилбарууд Объектыг ашиглан ижил систем Загварын техник ООД тэмдэглэгээ.3

Зураг 1-д системийн архитектурыг хоолой ба шүүлтүүрээр дүрсэлсэн болно Бүрэлдэхүүн хэсгүүд болон холбогчдын дизайны үгсийн санг тодорхойлсон хэв маяг г. онд хоолой ба шүүлтүүрийн хэв маяг гийн хувьд бүх бүрэлдэхүүн хэсэг нь урсгалыг хувиргадаг шүүлтүүрүүдэд өгөгдлийн болон тусгайлан бичсэнээр хангана оролт ба гаралтын интерфейс. Загварын бүх холбогч нь хоолойтой хоорондын хоёртын хамаарлыг дүрсэл хоёр шүүлтүүр ба өгөгдөл дамжуулах протокол. Хоолой бүр хоёр интерфейстэй: эх үүсвэр Үүнийг зөвхөн шүүлтүүрт холбож болно гаралтын интерфейс, мөн чадах угаалтуур зөвхөн шүүлтүүрийн оролтод хавсаргана интерфейс. Зураг 2-т илүү ихийг харуулав ашиглан энэ хэв маяг гийн албан ёсны тодорхойлолт Райтын тэмдэглэгээ.7 Райтын хэв маяг г тодорхойлолт нь семантикийг тодорхойлдог ашиглаж болох дизайны элементүүд хэв маяг г (хоолой ба шүүлтүүр), хамт хэрхэн хийхийг тодорхойлсон хя згаарлалтын багц дизайны элементүүдийг бүрдүүлж болно хоолой ба шүүлтүүрийн хэв маяг гаар системийг барих үед. Шууд хамаарал байдаг график тэмдэглэгээ болон хоорондын дизайны элементүүдийн албан ёсны тодорхойлолт. Загварын элемент бүрийг системийн график дүрслэл нь бичсэн бөгөөд төрөл нь -тэй тохирч байна

төрөл ба протоколын үзүүлэлтүүдийг өгөгдсөн
Райтын тодорхойлолтод. Тиймээс,
график диаграммын үнэндээ пүүстэй
тодорхойлолтод зориулсан семантик үндэслэл
болон дүн шинжилгээ.

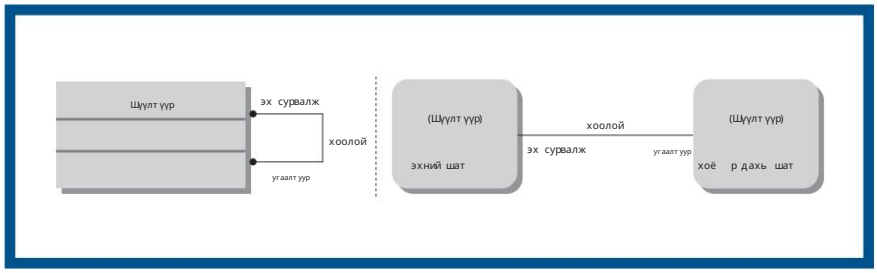
Түүврийн системын хоёр үндсэн системтэй
Бүрэлдэхүүн хэсгүүд, 1-р шат ба үе шат гэсэн шошготой
2, тус бүрнь өгөгдлийг хувиргадаг
дамжуулж, дараагийн руу илгээнэ
доод урсгалын бүрэлдэхүүн хэсэг. Бүрэлдэхүүн
хэсгүүдийн хоолойн протоколоор харилцан үйлчилдэг
Зураг 2-т заасан. хялбар болгох үүднээс,
Зураг 1 ба 2-т зөвхөн хоёр трансформацыг
харуулсан бөгөөд системийн оролтыг үл тоомсорлодог
болон гаралт.

Бид гурван ажиглалт хийж болно

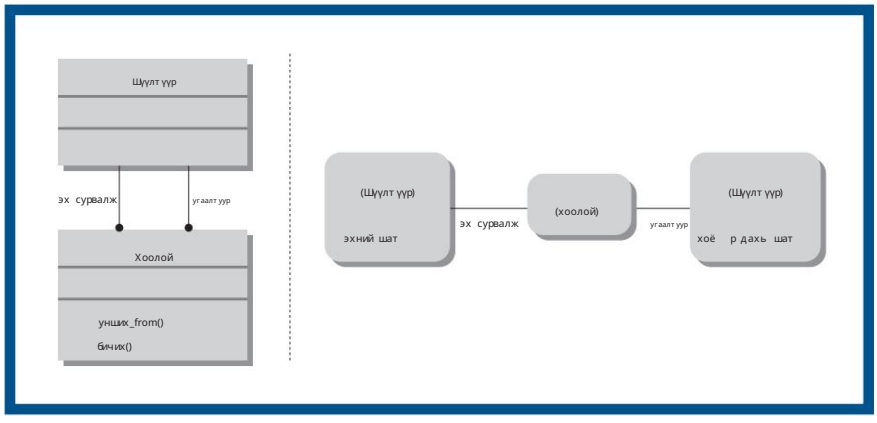
Энэ архитектурын дизайны талаар, ялангуяа
ОМТ-д суурилсан
ижил системийн дизайны Зураг 3-т үзүүлэв
дамжуулан 5. Нэгдүгээрт, шүүлтүүр хооронд
харилцан үйлчлэлийн протокол бага,
тодорхой, сайн тодорхойлсон. The
Зураг 2-т үзүүлсэн Райтын тодорхойлолт
хоолойн холбогчтой холбоотой
хоёр шүүлтүүрийн хооронд (мөн хоолойн
төрлийн бүх холбогчтой). Энэхүү тодорхойлолт
нь хоолойгоор дамжуулан өгөгдлийг дамжуулах
протокол, дараалалтай тодорхойлдог
хоолойн зан байдал, төрөл бүрийн
хоолой хангаж чадах интерфейсүүд
түүний хавсаргасан шүүлтүүрүүд. Учир нь анхдагч
програм хангамжийн архитектурын анхаарлын төвд байна
Бүрэлдэхүүн хэсгүүдийн харилцан үйлчлэлийг
тодорхойлоход энэ чадвар чухал юм.
Хоёр дугаарт, эдгээр, холбогч хоёр улаан
шүүлтүүр ба хоолой
хэвлэгдэн, сайн тодорхойлогдсон байх
интерфэйсүүд. Үүний үр дүнд хоолой нь хялбархан
төгсгөл бүрийн шүүлтүүрт үзүүлж буй үйлчилгээ.
Үүний нэгэн адил шүүлтүүр хийж болно
түүний интерфэйс бүрийг зааж өгөх
оролт эсвэл гаралтыг хангах болно, түүнчлэн
дамжуулж буй өгөгдлийн төрөл. Энэ
Энэ жишээнд, дээд урсгалын шүүлтүүр боломжтой
зөвхөн хоолой руу бичих ба доош урсгал шүүлтүүр
нь зөвхөн
хоолой, зохисгүй нэвтрэхээс сэргийлдэг
холбогч функцэд (жишээ нь
хоолойноос дээш урсгалын хоолойн заалт).
Эцэст нь хэлэхэд бага ойлголт байдаг учраас
-д суурилуулсан холбогч семантикийн
хэвлэгдэн гийн тодорхойлолтыг бид үнэлж болно
эрх чөлөөт зэрэг системийн нийтлэг шинж чанарыг
тодорхойлох загвар
мухардлаас (систем цикл агуулаагүй
тохиолдолд), дамжуулах чадвар
хувьсах хэмжээ, системийн болзошгүй хүндрэлүүд.

Зураг 1 ба 2-т үзүүлсэн загваруудыг
архитектурын загвараас ялгаатайгаар
3-аас 5-р зурагтаар ялгаатай байна
Ижил системийн OOD-уудыг илүү боловсронгуй
тайлбарлаж байна.
Эхний ОМТ диаграмм, Зураг 1,
гэсэн энгийн ангийн диаграммыг өгдөг
шүүлтүүр бүр бусадтай холбоотой байж болно
хоолойн холбоогоор шүүлтүүрүүд. Хоолой бүр
холбоо нь эх сурвалж, шингээгч үүрэгтэй
чиглэлийг зааж өгөх. Жишээ нь
Зураг 1-ийн диаграмм нь жишээг дүрсэлсэн болно
энэ ангийн бүтцийг ашиглан систем.
Эхний болон хоёр дахь шатны
шүүлтүүрүүдийн хооронд холбоо нь үнэн биш юм
Шүүлтүүрийн анги зэрэг нэгдүгээр зэрэглэлийн байгууллага болон
тиймээс дэмжих чадваргүй
Архитектур дахь хоолой шиг тодорхой,
боловсронгуй протоколын тайлбар
жишээ. Харин энэ нь ерөнхий холбоо бөгөөд дээд
урсгалын шүүлтүүр гэсэн үг юм
-ийн аль ч нийтийн аргыг дуудаж болно
доод урсгал шүүлтүүр. Хэдийгээр объектууд боломжтой
ОМТ дахь боловсронгуй байгууллагууд байх
Парадигмын хувьд объектуудын харилцан
үйлчлэлийг тодорхойлох үгсийн сан нь
архитектурын тайлбарт ашиглахад харьцангуй
хомсдолтой байдаг.

Зурвас илгээх боломжтой аливаа объект
өөр объект руу хүсэлт гаргаж болно
зорилтот объект нь нийтийн аль нэгийг нь дууддаг
аргууд. Үр дүнтэй ганц бие байдаг,
бүх объектоор хангагдсан хавтгай интерфейс
бүх объектууд. Үүний үр дүнд энэ нь хэцүү байдаг
Үүн дээр үндэслэн үзүүлж чадах үйлчилгээгээ
хялбархан архитектурын объект
Хүсэлт гаргагчийн интерфейсийн талууд



Зураг 3. Зураг 1-д үзүүлсэн ижил системийг энд гэнэн объект руу чиглэсэн тэмдэглэгээг (ОМТ) ашиглан дүрсэлсэн болно.



Зураг 4. Нэг системийн архитектурыг тодорхойлохын тулд ОМТ-ийн тодорхойлолтыг ашигладаг
Зураг 1-д үзүүлсэн. Хоолой нь одоо нэгдүгээр зэрэглэлийн загвар зохион бүтээгч байгууллага юм.

ашиглах ба хоёр объектын хоорондох холболтын төрөл.

Эцэст нь системийн шинжчанарыг тодорхойлоход хэцүү байдаг

Загварын арга нь объект хоорондын харилцан үйлчлэлийн харьцангуй нарийн төвөгтэй протоколуудыг дүрслэх боломжийг олгодог.

холболтын үгсийн сан, интерфейсийн хэгзаарлалт. Жишээлбэл, холбогдох объектын дурын аргыг хүссэн үедээ дуудах чадварын өгөгдлийн урсгалын шинжчанар, түгжрэлээс ангид байх боломжийг тодорхойлоход хүндрэл учруулдаг бөгөөд аль аль нь өмнө тайлбарласан программ хангамжийн архитектур болон архитектурын хэвмаягийн бүтцийг ашиглан харьцангуй хялбар тооцоолдог.

Зураг 4-т 3-р зурагт үзүүлсэн дизайны зарим асуудлыг шийдвэрлэх оролдлогыг харуулав. Энэ нь хоолойн холбогчийг нэгдүгээр зэрэглэлийн объект болгох замаар хийдэг. Энэ диаграммд бид хоёр шүүлтүүр объектыг холбохын тулд хоолойн объектыг ашигладаг. ОМТ тэмдэглэгээгээр ашиглан динамик болон функциональ загваруудыг холбох замаар Pipe ангид зан төлөвийн семантик нэмэх боломжтой болсон. Pipe анги нь мөн read_from() болон write_to() гэсэн хоёр шинэ аргыг нэвтрүүлсэн бөгөөд шүүлтүүрүүд хоолой дээр өгөгдөл илгээх эсвэл түүнээс өгөгдлийг уншихын тулд дуудах ёстой.

Хоолойг хоёр шүүлтүүрийн хооронд байрлуулсныхаа дараа нь дээд урсгалын шүүлтүүр нь доод урсгалын шүүлтүүрийн өгөгдлийг хүлээн авч, боловсруулж байгааг мэдэхээ больсон явдал юм. Үүний үр дүнд дээд урсгалын шүүлтүүр нь доош урсгалын шүүлтүүрийн аргуудад хандах боломжгүй болсон. Энэ нь зөвхөн тэдгээрийг холбосон хоолойд нэвтэрч, урсгалын шүүлтүүрээс ихээхэн хэмжээний хараат бус байдлыг хангаж, харилцааг холбоог дамжуулах боломжтой.

хоолойд хариуцлага хүлээх.

Гэсэн хэдий ч энэ загварт мөдэгдэхүйц хязгаарлалт байсаар байна. Хоолойн объектын хавсаргасан шүүлтүүрүүдийнхээ аль алинд нь өөрийн бүрэн аргын интерфейсийг санал болгох ёстой тул шүүлтүүрийн аль нэг нь write_to() эсвэл read_from() аргыг ашиглаж болно. Гэхдээ өгөгдлийн урсгалын зөв чиглэлийг хадгалахын тулд бид эх дүрээр тэмдэглэсэн дээд урсгал шүүлтүүрийг зөвхөн write_to() аргыг, харин шингээгчийн дүрээр тэмдэглэсэн доод урсгал шүүлтүүрийг зөвхөн ашиглахыг зааж өгөх боломжтой байх ёстой. read_from() арга. Харамсалтай нь ОМТ тэмдэглэгээ нь эдгээр хязгаарлалтыг албан ёсоор тодорхойлох боломжийг бидэнд олгодоггүй. Хоолойн чиглэл, сайн тодорхойлсон зан төлөвын дизайны шинжилгээ, баталгаатай хамтын алгоритм болно. Энэ протоколыг дагаж мөрдөх шүүлтүүрүүдийг бий болгох нь зарцуулагдахгүй боловч стандарт OOD ойлголтыг ашиглан энэ хязгаарлалтыг ерөнхийдөө тодорхой зааж, хэрэгжүүлэхэд хэцүү байдаг.

Дизайн загварууд. Хоолой ба шүүлтүүрийн хэвмаягийн системд ашиглах архитектурын хоолойн холбогчийг тодорхойлох объект руу чиглэсэн арга бариул, ямар дизайнд хэрхэн зөв оруулах дүрмүүдын олон объектын хамтын ажиллагааг шаардсан болохыг тодорхой. Шинээр гарч ирсэн дизайны загварын хэвмаягийн үзэл баримтлалын энэ асуудлыг шийддэг.

Зураг 5-д хоолой ба шүүлтүүрийн энгийн архитектурын гурав дахь буюу эцсийн засварыг үзүүлэв. Энэ удаад хоолойн бүтцийг харилцан үйлчлэгч гурван объект болгон хуваасан:

хоолойн объект нь өгөгдлийн урсгал болон буферийг хялбар, эх үүсвэрын дээд урсгалын шүүлтүүрт холбогдож, хоолойд зөвхөн write_to() интерфейсээр хангадаг.

харгалзах угаалтуурын объект нь доод урсгалын шүүлтүүрт залгагдсан бөгөөд хоолойд зөвхөн read_from() интерфейсээр хангадаг.

Энэхүү шийдэл нь шүүлтүүрийн аль алинд нь read_from() болон write_to() аргуудын аль алинд нь хандах асуудлыг шийддэг.

хязгаарлагдмал интерфэйс бүхий зуучлагч объектуудыг хангах.

Гэсэн хэдий ч энэ загварын өөрөө зохисгүй аргууд руу нэвтрэх асуудлыг бүрэн зөөлрүүлж чадахгүй. Энэ нь хоолойн зохисгүй аргууд руу хандсан шүүлтүүрийн объектоос хоолойн аргууд руу буруу хандсан эх үүсвэр, шингээгч объект руу асуудлыг шилжүүлдэг.

Хоолой, эх үүсвэр, угаалтуурын аргууд нь бүгд хоолойн холбогч загвараар бүрхэгдсэн байдаг тул тохирох хоолойн протоколын дагуу гурван объект харилцан ажиллахыг зөвшөөрсөн протоколыг тайлбарлах боломжтой; өөрөөр хэлбэл, хоолой объект нь дараалал болон буферийн бүх асуудлыг хариуцдаг, зөвхөн эх үүрэг нь хоолойн enqueue_data() аргыг дуудаж болно, зөвхөн шингээгч үүрэг нь хоолойн dequeue_data() аргыг дуудаж болно.

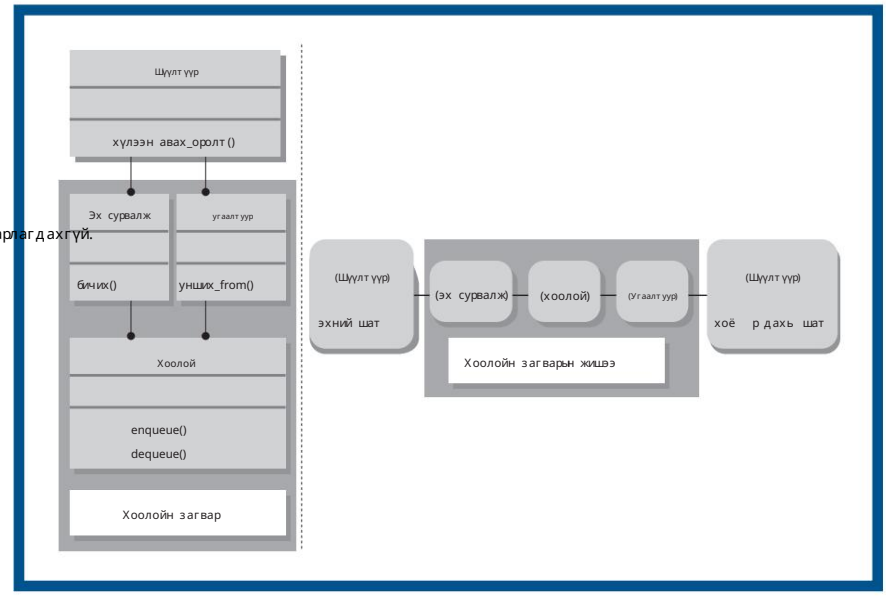
Энэ протоколын дэлгэрэнгүй мэдээллийг загвар болон түүний объектуудад кодлох боломжтой.

Загварын арга нь нэгдмэл багтаахыг хүсэхгүй байгаа объектуудын хоорондын харилцан үйлчлэлийн харьцангуй нарийн төвөгтэй протоколуудыг тайлбарлах боломжийг олгодог. Бид Шүүлтүүрийн ангилалд эх үүсвэр болон шингээгч объектуудын хангадаг олон хязгаарлалтыг тайлбарлаж болох байсан ч үүнийг хийснээр ангид ерөнхийдөө тохиромжгүй хязгаарлалтуудыг нэмж, дахин ашиглах чадварыг мөдэгдэхүйц бууруулж болзошгүй юм. Бид дизайныг тодорхой болгох, дизайны талаархи үндэслэлийг хөнгөвчлөхийн тулд үүнийг багтаахыг үнэхээр хүсч байгаа үед энэ нь харилцан үйлчлэлийн протоколыг илүү өргөн хүрээний бүтэцүүдийн дунд түгээх байсан. Загварын тодорхойлолттой харилцан уялдаатай гурван өөр төрлийн объектыг ашиглах хэрэгцээ нь энгийн байх зорилгод ихээхэн саад болж байна. Хэдийгээр бид хоолойн холболтыг ОМТ болон дизайны хэвмаягийг ашиглан загварчилж болох боловч архитектурын тэмдэглэгээ бүхий энгийн төрлийн тайлбартай сумыг зааж өгснөөр үүссэн энгийн байдал, гоёмсог байдлын ихэнх нь холбогчийн нэгдүгээр зэрэглэлийн объект байхаа больсон үед алдагддаг. OOD парадигм.

Дүгнэлт. Эдгээр жишээнүүдээс харахад архитектурын дизайн нь объектын систем болгон хамгийн сайн загварчлах албагүй хийсвэрлэлүүдийг агуулдаг бөгөөд наад зах нь объектын яг цуу ут гаараа аргын дуудлагын тусламжтайгаар харилцан үйлчлэлцдэг өгөгдлийн төрлүүдийг багтаасан байдаг. Энэ цэг нь дамжуулах хоолой ба шүүлтүүр гэх мэт мэдээллийн урсгалын хэв маяг гаар ялгаагдахгүй. Бид энгийн хэв маяг, үйлчлүүлэгч-серверт суурилсан хэв маяг, тархсан мэдээллийн баазын хэв маяг эсвэл архитектурын дизайны бусад олон хэв маяг гаар хийгдсэн архитектурын дизайны талаар ижил төстэй аргументуудыг харьцуулбарган гаргаж чадна.

Архитектурын хэв маяг нь ямар нэг бүрийн дизайны гэр бүлийн өргөн хүрээг дүрсэлж чаддаг тул объект хандалтат дизайны архитектурын дизайны хэв маяг гэж үзэх нь сонирхол татахуйц бөгөөд бүх бүрэлдэхүүн хэсгүүд нь объект бөгөөд бүх холболтууд нь энгийн нэгдэл эсвэл нэгтгэлүүд (ОМТ үгсийн санг ашиглах) юм. Үнэн хэрэгтээ, олон OOD хэрэглүүрээр дэмжигдсэн ердийн анхдагч системийн барилгын байгууламжийг хангадаг объектод суурилсан архитектурын хэв маяг гийг тодорхойлох боломжтой. Энэхүү үзэл баримтлал нь архитектурын хийсвэрлэлийг авч үздэг OOD-ийн дэд бүлэгт бүрэн үндэслэлтэй юм. Нөгөөтэйгүүр, архитектурын дизайны хамрах хүрээнээс гадуур ерөнхийд нь авч үздэг, OOD-аас шууд шийддэг дизайны хэд хэдэн асуудал байдаг. Жишээ нь асуудлын талбар, шаардлагуудыг загварчлах арга замууд, өгөгдлийн бүтэц, алгоритмыг зохион бүтээх зэрэг хэрэгжүүлэх асуудлууд багтана. Эдгээр асуудал нь програм хангамжийн хөгжүүлэлтэй холбоотой бөгөөд системийн архитектурыг төлөвлөхдөө анхаарах ёстой; Гэсэн хэдий ч архитектурын тайлбарт бүгдийг нь шууд илэрхийлж, тусгах нь зайлшгүй байх ёсгүй.

Архитектурын дизайн нь бүрэлдэхүүн хэсгүүдээс систем зохиох, эдгээр бүрэлдэхүүн хэсгүүдийн харилцан үйлчлэлтэй холбоотой байдаг. Иймээс найрлага нь системийг хийсвэрээр харах боломжийг олгодог бөгөөд ингэснээр дизайнер нь системийн түвшний дүн шинжилгээ хийж, системийн бүрэн бүтэн байдлын хязгаарлалтыг тайлбарлах боломжтой болно. Жишээ нь дамжуулах хурд болон мухардалд орохгүй байх зэрэг орно. Архитектурын дизайны эдгээр ялгаатай талууд нь хэд хэдэн зүйлийг онцолж өгдөг



Зураг 5. 1-р зурагт үзүүлсэн системийн ОМТ-д суурилсан энэхүү техникийн үзүүлэлтэд хоолойн холбогчийг дизайны загвар хэлбэрээр дүрсэлсэн болно. Холбогч интерфейсүүд (эх ба угаалтуур) нь одоо нэгдүгээр зэрэглэлийн нэгжүүд юм.

объект хандалтат дизайнтай чухал ялгаа. Хэдийгээр хоёр улаа системийн бүтцэд ерөнхийдөө хамаатай боловч архитектурын дизайн нь OOD-ээс өгөгдөл хийсвэрлэлийн илүү баялаг цуглуулгыг агуулдаг. Эдгээр хийсвэрлэлүүд нь шинэ төрлийн нарийн төвөгтэй системийн цавуу (эсвэл холбогчийг) дүрслэх чадварыг дэмждэг. Өмнө дурьдсан хоолойн холбогчоос гадна үйлдлийн систем, RPC-д суурилсан SQL асуулга эсвэл хоёр фазын гүйлгээний протокол гэх мэт n-ар холбогчийг тодорхойлох боломжтой.

Архитектурын хийсвэрлэл нь дизайнеруудад олон интерфейсийг бүрэлдэхүүн хэсгүүдтэй холбож, загвар дээрх топологик болон бусад семантик хязгаарлалтуудыг илэрхийлэх боломжийг олгодог.

Тиймээс архитектурын дизайн ч, объект хандалтат загвар ч нөгөө нь багтаадаггүй. Тэд хоёулаа хөгжлийн үйлдвэрлэлийн цаг үед тохиромжтой бөгөөд нийтлэг ойлголт, үзэл баримтлалыг хуваалцдаг. Та ОО-д суурилсан архитектурын хэв маяг гийг тодорхойлж болохын адил нарийн төвөгтэй бүрэлдэхүүн хэсгийг хэрэгжүүлэх, сайжруулахын тулд OOD ашиглаж болно.

архитектурын дизайн дахь холбогч.
Энэ хоёр хандлагын шийдвэрлэх үндсэн асуудлууд болон тэдгээрийн гаргаж буй хийсвэрлэх механизмууд нь ижил бишүү.

Архитектурын хэв маяг ба ДИЗАЙН ХЭВЭЭ

Өмнөх жишээнүүдэд дурдсанчлан уламжлалт OOD-ийн үндсэн хязгаарлалтуудын хоёр нь объектуудын бүлэг хэрхэн харилцан үйлчлэлцдэгийг тодорхойлох, дахин ашиглах объектын цуглуулгуудыг зааж өгөх, савлахад бэрхшээлтэй байдаг. Зураг 5-аас харахад дизайны загвар нь эдгээр бэрхшээлийг арилгах боломжтой. Загварын хэв маяг гийн үндсэн санаа нь нийтлэг хэлц үгсийг програм хангамжийн дизайнд олон удаа олдог бөгөөд эдгээр хэв маяг гийг тодорхой болгож, кодчилж, ижил төстэй асуудлуудад зохих ёсоор хэрэглэх ёстой.

Сүүлийн дөрөв, таван жилийн хугацаанд эдгээр хэв маяг гийг илэрхийлэх хэд хэдэн арга бий болсон бөгөөд ихэнх нь OOD-ын загварт анхаарлаа хандуулсан.4,5

Анхдагч үгсийн сан:

Балар эртний үгсийн сан	Албан бус тайлбар	Интерфейсийн хя згаарлалт	Үл хөдлөх хөрөнгө
Бүрэлдэхүүн хэсгүүд:		(порт)үүд бүрэлдэхүүн хэсгийн интерфейсийг тодорхойлох)	
Үйл я вц	OS үйл я вц. Оролтын мессежийг уншиж, үр дүнг гаралтын интерфейс рүү илгээдэг.	дор хая ж 1 асинк оролтын порт дор хая ж 1 асинхрон гаралтын порт дор хая ж 0 синхрончлолын порт	боловсруулах зардал, хувь хэмжээ, оролтын мессежийн төрөл(үүд), гаралтын мессежийн төрөл(үүд)
Нөөц	Процессуудын төлөөх бүрэлдэхүүн хэсэг.	я г 0 асинх порт дор хая ж 1 синхрончлолын порт	нөөцийн зардал
Төхөөрөмж	Урьдчилан тодорхойлсон тарифаар систем рүү мессеж илгээх.	я г 0 синхрончлолын порт я г 0 асинхрон оролтын порт дор хая ж 1 асинхрон гаралтын порт	гаралтын хурд, гаралт¯мессеж¯төрөл
Холбогч:		(үүрүүд) (хөвгүүдэн холбогч интерфейсийг тодорхойлох)	
async¯msg¯pass	Бичсэн асинхрон мессежийн суваг зурвасууд.	я г 1 асинх¯оролтын үүрэг я г 1 асинх¯гаралтын үүрэг	мессежийн төрөл
async¯msg¯pass rendevous	Async¯msg¯pass¯тай адил боловч мессеж илгээхээс өмнө давтагдах шаардлагатай. N¯ary холбогч.	дор хая ж 1 асинк¯оролтын үүрэг я г 1 асинх¯гаралтын үүрэг	мессежийн төрөл
синк хийх хүсэлт	Хоёр ртын синхрон хүсэлтийн суваг, бичсэн мессежүүд.	я г 1 синк дуудагч үүрэг я г 1 синк дуудагч үүрэг	мессежийн төрөл

Загварын дүрмүүд (жагсаалтын RTP/C загварын бүх дүрмийн дэд хэсэг юм):

- Async¯msg¯pass холбогчын зөвхөн (процесс, процесс) эсвэл (төхөөрөмж, процесс) хос бүрэлдэхүүн хэсгүүдийг холбож болно.
- Синк хийх хүсэлтийн холбогчын зөвхөн хос бүрэлдэхүүн хэсгүүдийг (боловсруулах, нөөц) холбож болно.
- Бүх процессууд хавсаргасан оролтын интерфейстэй байх ёстой.
- Холбогч бүрийн оролтын мессежийн төрөл нь гаралтын мессежийн төрөлтэй тохирч байх ёстой.
- ...

Загварт суурилсан дизайны шинжилгээ:

Шинжилгээ	Тодорхойлолт
Мессежийн замыг шалгах	Мессеж бүрийн дагуу зөвхөн хүчинтэй мессежийн төрлүүд дамжихыг баталгаажуулдаг. Мессежийн төрөл таарахгүй байгааг эрт илрүүлэх боломжийг олгодог.
Үнийн тооцоо	Үйл я вц бүрт хя налт, нөөцийг хэр олон удаа өгч болохыг тодорхойлдог.
Төлөвлөлт	Энэ дизайныг хэрэглэгчийн тодорхойлсон гүйцэтгэлийн шинж чанар бүхий нэг процессор дээр төлөвлөж болох эсэхийг тооцоолно.
Эвристикийг засах	Хэрэв системийг төлөвлөх боломжгүй бол энэхүү шинжилгээ нь саад бэрхшээлийг тодорхойлж, болзошгүй засвар сайжруулалтыг санал болгодог.

Зураг 6. Бодит цагийн үйлдвэрлэгч/Хэрэглэгчийн (RTP/C) загварын албан бус тодорхойлолт.

Гэсэн хэдий ч дизайны загварууд өргөжин тэлж байна үүнээс цааш. Тодорхойлох гурван үндсэн шаардлага байдаг

Програм хангамжийн дизайны загварыг дахин ашиглах: the дизайны домэйныг сайн ойлгох ёстой, энэ нь капсулыг дэмжих ёстой дизайны элементүүд бөгөөд энэ нь заавал байх ёстой алдартай болон цуглуулгад боловсруулсан

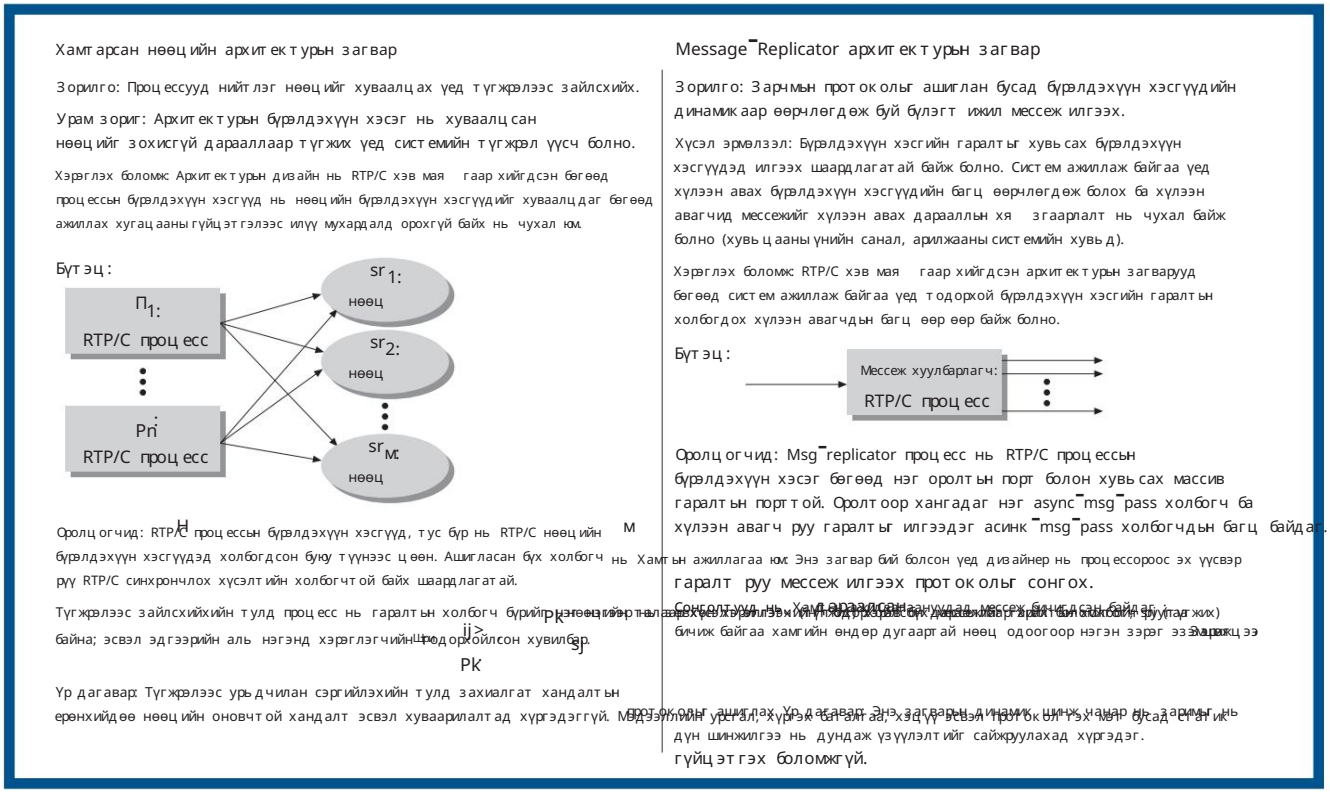
батлагдсан дизайны хэлц үгс. Загварын хэлүүд дараа нь мэдлэгтэй дизайнерууд кодчил батлагдсан загвар, дизайны хэлтэрхий, болон дараа дахин ашиглах хүрээ.

Архитектурын хэв маяг нь хоорондоо нягт холбоотой байдаг загвар дизайны хоёр рагаар. Эхлээд, архитектурын хэв маяг гэж үзэж болно хэв маягийн төрлүүд8 эсвэл магадгүй түүнээс дээш хэв маягийн хэлээр үнэн зөв.9 Архитектурын хэв маягийг тодорхойлох нь а

Гэсэн хэдий ч дизайны загварын а хамрах хүрээний талаар нэлээд өргөн тодорхойлолт дизайны загварууд. Архитектурын хэв маяг юм дизайн гэж үзсэн нь дээр байх архитектуруудад өгдөг хэл үгсийн сан ба хүрээ Тэд ашигтай дизайны хэв маягийг бий болгож чадна ОМТ гэх мэт тодорхой асуудлуудыг шийдвэрлэх хүрээ, тэмдэглэгээг өгдөг объектуудтай ажиллах. Хоёрдугаарт, а өгөгдсөн хэв маягийн багц байж болно хэлц хэрэглээ. Эдгээр хэлц үгс нь үүрэг гүйцэтгэдэг бичил архитектур, эсвэл архитектур тодорхой архитектурын хэв маягтай ажиллахад зориулагдсан дизайны загварууд. Эдгээрийн хүрээнд хүрээг бий болгосноор хээ нь ажилладаг, дизайнер ашиглан загварын хэв маягийг өргөнөөр ашиглаж болно

дүрслэх болон аналитик чадвар батлагдсан механизмуудын хамт дизайны тодорхой сорилтуудыг шийдвэрлэх загварын хэв маягийн хэлбэр.

Бид хэв маяг, архитектурыг хардаг хэв маягийг нэмэлт механизм болгон ашигладаг дизайны туршлагад зориулж. Ан архитектурын хэв маяг нь цуглуулгыг өгдөг барилгын блокийн дизайны элементүүд, дүрэм зохиохоод тавигдах хя загаарлалтууд барилгын блок, дүн шинжилгээ хийх хэрэгсэл онд бүтээгдсэн загварууддыг удирдах хэв маяг. Загварууд нь ерөнхийдөө удирдамж өгдөг өргөн ангиллын бий болгохын тулд дүн шинжилгээ хийх тодорхой домайн дахь архитектурууд, хэв маяг нь жижиг, илүү тодорхой асуудлуудыг шийдвэрлэхэд төвлөрдөг өгөгдсөн хэв маяг (эсвэл олон загвар).



Зураг 7. RTP/C загварын хоёр загвар архитектурын загвар.

Мөн хэв маяг гийг анхаарах нь чухал юм
архитектур байх албагүй. Үнэхээр олон
сүүлийн үеийн гарын авлага дахь хэв маяг 4,5 хэллэцээр
системийн бүтцийн асуудлаас илүү доод түвшний
программыг нэгтгэх механизмын шийдэлтэй.

Загвар ба хэв маяг гийн жишээ. Архитектурын
хэв маяг гийн хамрах хүрээ, зорилго, мөн
тэдгээртэй хэрхэн холбогдож байгааг харуулахын тулд
дизайны хэв маяг гийн хувьд Зураг дээр өгөгдсөн
архитектурын хэв маяг гийн тодорхойлолтыг анхаарч үзээрэй
6. Бодит цагийн үйлдвэрлэгч/Хэрэглэгчийн хэв
маяг гэж тодорхойлсон энэ хэв маяг нь
Архитектуруудад тавихад тусалх зорилготой
хамтдаа бодит цагийн мультимедиа систем
нэгтгэсэн процессортой компьютер дээр ажиллаж байна.10
Зураг 6-д RTP/C хэв маяг гийн албан бус
тайлбарыг онцлон харуулав
онд бүтээгдсэн загварт хэрэглэгддэг (анхны)
загварын толь бичгийн төрлүүд
хэв маяг, дизайны дүрэм, хяналтууд
Элементүүдийг хэрхэн бүрдүүлж болохыг зааж
өгөх, дизайн дээр хийж болох дүн шинжилгээ.
RTP/C
загварын тодорхойлолтын анхдагч барилгын
блок, удирдамжийн багцыг тодорхойлдог
нэлээд өргөн хүрээг нэгтгэх
хангалттай сайн ойлгогдсон домэйн дотоод
системүүд.

Ийм нарийн тодорхойлсон хэв маяг тай байсан ч гэсэн,
Гэсэн хэдий ч харьцангуй бетон хийцтэй хэв
маяг нь чухал үүрэг гүйцэтгэдэг. The
RTP/C команд дизайны элементүүд болон
удирдамжийн байж болох хэлийг бүрдүүлдэг
илүү нарийвчилсан, бетон барихад ашигладаг
тодорхой асуудлуудын шийдэл. Энэ
хэв маяг нь сайн ойлгогдож өгдөг
сайн тодорхойлсон үгсийн сангийн хүрээ
бие даасан дизайны элементүүдийг бүрдүүлэх
бодит цагийн шинжилгээг дэмжих зарчмын арга
замаар. Зураг 7-д хийсэн хоёр хяналбаршуулсан
дизайны загварыг харуулав
RTP/C хэв маяг г-санах ойн сэрээ болон мессеж
чадвар, ашиглалтын үр дагавар гэх мэт
мэдээллийг өгдөг.
Бид эдгээр хэв маяг гийг ашиглан харуулсан
1995 оны номонд оруулсан бүтэц
Эрих Гамма болон түүний хамтрагчид.4 Энэ
хүрээ архитектурын хувьд сайн ажилладаг
хэв маяг, түүнчлэн ОО загварын хувьд, хамт
Голялгаа нь архитектурын хэв маяг нь
илүү тодорхой зүйлийг харуулдаг
дизайны асуудлуудын багц (өмнө дурдсанчлан
"Програм хангамжийн архитектур гэж юу вэ")
ОО загвараас илүү. Ялгаатай ОМТ болон
Объектуудыг ихэнх OOD хэв маяг гийн гарын
авлага, үгсийн сан, дүрэм журамд дизайны хэв
маяг гийг харуулахад ашигладаг.
архитектурын хэв маяг гийг тодорхойлоход ашиглаж болно
архитектурын дизайны загварууд.
Дараа нь ОМТ болон
-аас дизайны хэв маяг гийн тэмдэглэгээ
OOD загварын гарын авлагыг ашиглаж болно
архитектурын хэв маяг гийг мөн тодорхойлох. онд
үнэн хэрэгтээ, дизайны загваруудын хэд хэдэн
Гамма болон түүний хамтрагчид тайлбарлав
архитектурын зураг төсөлд хамаарах нь харагдаж байна.8
Жишээлбэл, фасадны загвар орно
Энэ нь Observer загвар болох объектуудын
цуглуулгад нэгтгэсэн интерфейсээр хангадаг
Энэ нь объектуудын хоорондын уялдаа
холбоог хадгалах механизмыг тодорхойлдог (эсвэл
бүрэлдэхүүн хэсэг), стратегийн загвар
Энэ нь алгоритмын микрофоны сонголтыг
интерфэйсийн шийдвэрээс хэрхэн салгах талаар заадаг.
Бүртгэгдсэн загваруудын аль нь чухал
зөвхөн архитектурын хэв маяг байх. Бүгд
доод түвшинд хэрэглэх боломжтой
дизайн (дэлгэрэнгүй дизайн гэх мэт
хэрэгжүүлэх код). Үүнээс гадна
энд жагсаасан архитектурын хэв маяг,
Гамма нар дахь хэд хэдэн хэв маяг.
Жишээлбэл, энэ номын архитектурын асуудлыг
шийдэж чадахгүй. Үйлдвэрийн арга
болон Flyweight загваруудыг. Энэ хоёр улаа
Жишээ нь, хэв маяг нь архитектурын ерөнхийд
нь тодорхойлсоноос доогуур түвшний
хэрэгжилтийн асуудлыг шийддэг.
Тиймээс, архитектурын дизайны хэв маяг
болон объект хандалтат дизайны загварууд нь
энгийн жишээнүүд
бүх загварын хэв маяг гийн ангилал. Дургүй
зөв загвар хэв маяг г, Гэсэн хэдий ч, а
архитектурын хэв маяг нь хэллээр хангадаг
гэр бүлийг дүрслэх хүрээ
сайн боловсруулсан програм хангамжийн архитектурын .

Загварын үүрэг нь архитектурын нийтлэг хэлц үгсийн архитектурын жишээ, хэв маяг гийг хоёр уланг нь илэрхийлэх хэлээр хангах явдал юм. Үүний үр дүнд архитектурын хэв маяг гийн үндэс суурь болсон бүтэц, үзэл баримтлалууд нь олон тооны бус, харин ОМТ гэх мэт OOD аргачлалын үндэс суурьтай харьцуулах боломжтой юм.

Гамма болон түүний хамтран ажиллагсдын өгсөн загвар гэх мэт дизайны хэв маяг.4 Архитектурын тодорхой хэв маяг гийг дизайны хэв маяг гийн жишээ гэхээсээ илүү хэв маяг гийг бүтээх хэл гэж үзэх нь дээр.

Архитектур, архитектурын хэв маяг, объект, дизайны загвар зөөлөн эдлэлийн дизайны нэмэлт талууд. Хэдийгээр эдгээр дөрвөн аргын авч үзсэн програм хангамжийн дизайны асуудал, талууд нь зарим талаараа давхцаж байгаа боловч аль нь ч нөгөөг нь бүрэн багтаадаггүй. Тус бүр нь төлөөллийн загвар, механизмын цуглуулгад санал болгох зүйлтэй байдаг. •

ТАЛАРХАЛ

Бид Роберт Алленд түстай тайлбар өгсөнд баярлалаа. Энэхүү судалгааг Үндэсний Шинжлэх Ухааны Сан №1 тэтгэлгийн хүрээнд ивээн тэтгэсэн. CCR 9357792 ба төгсөлтийн судалгааны тэтгэлэг; Райт лаборатори, Нисэхийн системийн төв, Агаарын цэргийн хүчний материальн командлал, USAF; Дэвшилтэт судалгааны төслүүдийн агентлагийн бүцалтгүй тусламжийн №. F33615-93-1-1330; болон Siemens Corporate Research.

Ашигласан материал

1. М.Шоу ба Д.Гарлан, Програм хангамжийн архитектур: Шинээр хөгжиж буй салбар дахь хэтийн төлөв, Прентис Холл, Энглвуд Клиффс, NJ, 1996.

2. Г.Абоуд, Р.Аллен, Д.Гарлан, "Програм хангамжийн архитектурт үгтэй өгөх хэв маяг гийг ашиглах" Proc. SIGSOFT '93: Foundations Software Eng., ACM, New York, 1993. Мөн Software Eng. Тэмдэглэл, 1993 оны 12-р сар, 9-20-р хуудас.

3. J. Rumbaugh et al., Object-oriented Modeling and Design, Prentice-Hall, Englewood Cliffs, NJ, 1991.

4. E. Gamma et al., Design Patterns: Elements of Reusable Object Oriented Design, Addison-Wesley, Reading, Mass., 1995.

5. В.При, Объект хандалтат програм хангамж хөгжүүлэх дизайны загвар, Addison-Wesley, Reading, Mass., 1995.

6. D. Perry and A. Wolf, "Foundations for the Study of Software Architecture", ACM Software Eng. Тэмдэглэл, боть. 17, №4, 1992 оны 10-р сар, 40-52-р тал.

7. Р.Аллен, Д.Гарлан, "Архитектурын холболтыг албан ёсны болгох," Прок. 16 дахь олон улсын конф. Software Eng., IEEE Computer Soc. Пресс, Лос Аламитос, Калифорниа, 1994, хуудас 71-80.

8. M. Shaw, "Some Patterns for Software Architecture", Pattern Languages of Program Design, Vol. 2, J. Vlissides, J. Coplien, and N. Kerth, edits., Addison Wesley, Reading, Mass., 1996, хуудас 255-269.

9. NL Kerth, "Caterpillar's Fate: A Pattern Language for Transformations from Analysis to Design", Pattern Languages of Program Design, JO Coplien and DC Schmidt, ред., Addison-Wesley, Reading, Mass., 1995.

10. К.Жеффай, "Бодит цагийн үйлдвэрлэгч/хэрэглэгчийн парадигмыг ашиглан, урьдчилан таамаглаж болох бодит цагийн системийг бүтээх парадигм" Прок. 1993 он ACM/SIGAPP Symp. Хэрэглээний тооцоолол, ACM Press, Нью-Йорк, 1993, хуудас 796-804.

Роберт Т.Монро нь Карнеги Меллоны их сургуулийн Компьютерийн шинжлэх ухааны тэнхимд докторын зэрэг хамгаалсан. Тэрээр Карнеги Меллонд компьютерийн шинжлэх ухааны магистр, Мичиганы их сургуульд бакалавр зэрэгтэй. Түүний судалгааны сонирхол нь програм хангамжийн дизайны хэрэгслүүд, програм хангамжийн архитектур, програм хангамжийн дизайны мэдлэгийг илэрхийлэх хэл юм. Тэрээр IEEE Компьютерийн Нийгэмлэг болон МУЗ-ийн гишүүн юм.

Ралф Мелтон бол Карнеги Меллон их сургуулийн Компьютерийн шинжлэх ухааны тэнхимийн төгсөх ангийн оюутан юм. Тэрээр Стэнфордын их сургуульд бакалаврын зэрэгтэй. Түүний судалгааны сонирхолд програм хангамжийн архитектур, дизайны фрагмент, тэдгээрийн найрлагыг дүрслэх албан ёсны аргуудыг ашиглах зэрэг багтдаг.

Эндрю Компанек бол Карнеги Меллоны их сургуулийн Компьютерийн шинжлэх ухааны сургуулийн ABLE судалгааны бүлгийн судалгааны программист юм. Тэрээр сая хан Карнеги Меллоноос математик, компьютерийн шинжлэх ухааны чиглэлээр бакалаврын зэрэг авсан. Түүний одоогийн ажил нь програм хангамжийн архитектур зориулсан дүрслэл, автоматжуулсан зохион байгуулалтын хэрэгслүүдийн дизайн, боловсруулалтыг багтаасан болно.

Дэвид Гарлан нь Карнеги Меллоны их сургуулийн компьютерийн шинжлэх ухааны дэд профессор бөгөөд ABLE төслийг удирддаг. Түүний судалгаа нь програм хангамжийн архитектур, дахин ашиглах боломжтой дизайныг бүтээхэд албан ёсны аргуудыг ашиглах, програм хангамж хөгжүүлэх орчинд чиглэдэг. Тэрээр Карнеги Меллонд докторын зэрэг хамгаалсан бөгөөд Амхерст коллежийн бакалавр, Англиийн Оксфордын их сургуульд магистрын зэрэг хамгаалсан. Тэрээр IEEE Компьютерийн Нийгэмлэг ба МУЗ-ийн гишүүн юм.

Энэ нийтлэлийн талаарх асуултыг Карнеги Меллон Их Сургуулийн Компьютерийн Шинжлэх Ухааны Сургуулийн Роберт Монрод, 5000 Forbes Ave., Pittsburgh, PA 15213; bmonroe+@cs.cmu.edu.