< Програм хангамжийн архитектур ба зохиомж >

(Лекц 11 ба 12 тэмдэглэл)

Х. Очирсүх

ХШУИС, Програм хангамжийн 3 курсын оюутан, [20b1num0429@stud.num.edu.mn](mailto:20b1num0429@stud.num.edu.mn)

# Оршил

Энэхүү лекцийн 10 ба 11 хүрээнд зохиомж ба шинжилгээний харьцуулалт, зохиомжийн үеийн үндсэн ойлголтууд, мөн архитектурын ойлголтын талаар судлах

# Зорилго

Лекцүүдийн хүрээнд үзэх шаардлагатай агуулгуудыг судлаж, өөрийн үгээр тайлбарлаж тэмдэглэл бичих. Зорилгодоо хүрэхийн тулд дараах зорилтуудыг дэвшүүлсэн байгаа.

1. Лекцийн материал болон лекцийн үзэж ойлгох
2. Шинжилгээ болон зохиожийн ялгаа ойлгох
3. Объект хандлагат зохиомжын талаарх ойлголтыг ойлгох
4. Архитектурын зохиож гэж юуг тайлбарлах
5. Зохиомжын дүрмийг судлах

Объект хандлагат зохиомж ба шинжилгээний үе хоорондоо ялгаатай байдаг.

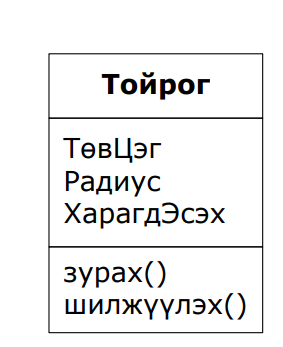
|  |  |
| --- | --- |
| Шинжилгээ | Зохиомж |
| Бизнесийн шийдэл | Техникийн шийдэл |
| Хэрэглэгчийн талаас харсан асуудлын талбар | Хөгжүүлэлтийн талаас харсан асуудлын талбар |

Объект хандлагат зохиомжын үндсэн хийгдэх ажил:

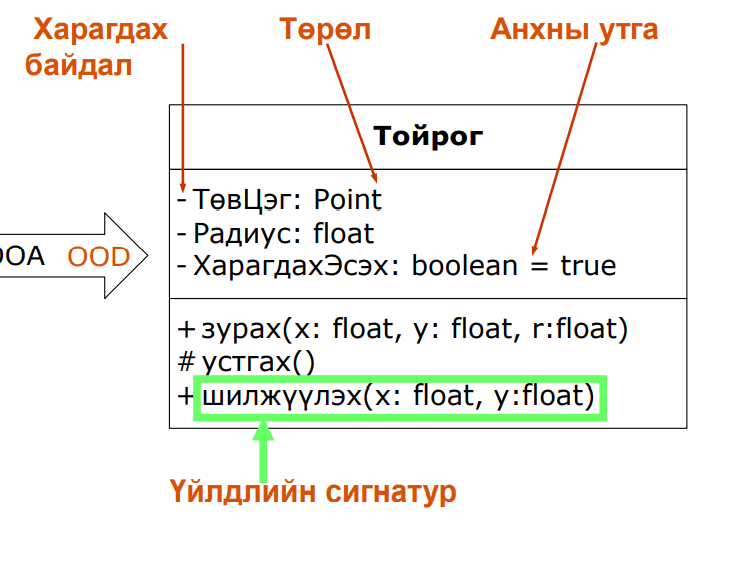
1. Шинжилгээний үед хийгдсэн ажлыг улам өргөтгөдөг.
2. Загварыг нарийвчлан сайжруулах
3. Архитектур гаргаж, бүтэцлэлт хийдэг.

Объект хандлагат загвар нь Шинжилгээнээс зохиомжид тасралтгүй шилжилт. Бүтцийн эвдрэлгүй!

Бид шинжилгээний үед классын диаграммыг дүрслэхдээ:



Объект хандлагат зохиомжын үед нэмэлтээр харагдах байдал, төрөл, анхны утга, үйлдлийн сигнатурыг тусгаж өгч байдаг.



Тус бүрд дэлгэрэнгүй авч үзье

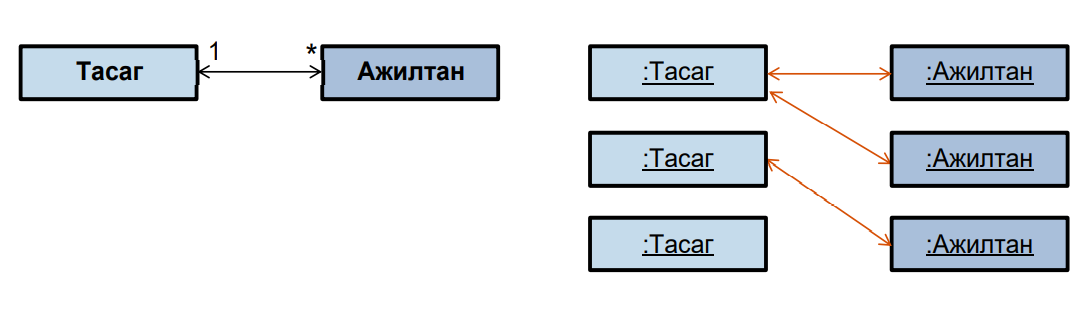
|  |  |  |
| --- | --- | --- |
|  | Шинжилгээ | Зохиомж |
| Харагдах байдал | : Бүх шинжүүд классаас гадагш нууцлагдсан бөгөөд зөвхөн үйлдлээр л уншиж, өөрчилж болно. | • + public: Бусад бүх класст харагдана  • # protected: Класс дотор мөн дэд класст харагдана  • - private: Зөвхөн класс дотор харагдана |
| Үйлдлийн сигнатур | Өгөгдлийн битүүмжлэл болон нууцлалтай байдаг. | • Зохиомж болон хэрэгжүүлэлтийн үед харагдацыг олгосноор битүүмжлэл болон Далдлалтыг салгадаг |
| Холбоосын жолоодлого | Ихэнх холбоос 2 чиглэлтэй байдаг. | Холбоос нэг юм уу аль эсвэл 2 чиглэлтэй байдаг. |

Холбоосын заалтан хувьсагч тусламжтайгаар хэрэгжүүлэх

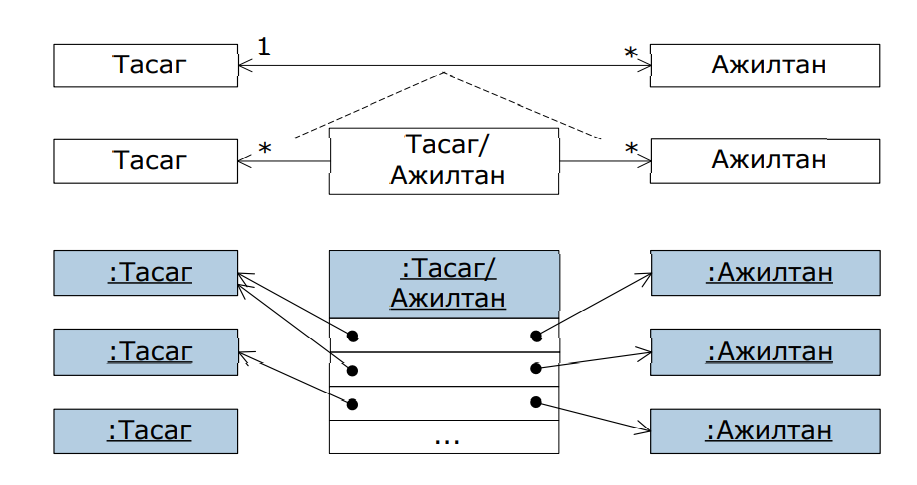
Холбоосын чиглэл болгоны объектийн заалтан хувьсагч ашиглан хэрэгжүүлэхийг хэлнэ.

Харьцааны тоо 0..1 Нэг ширхэг заагч

Харьцааны тоо 1-ээс их бол Олон ширхэг заагчтай гэсэн үг.



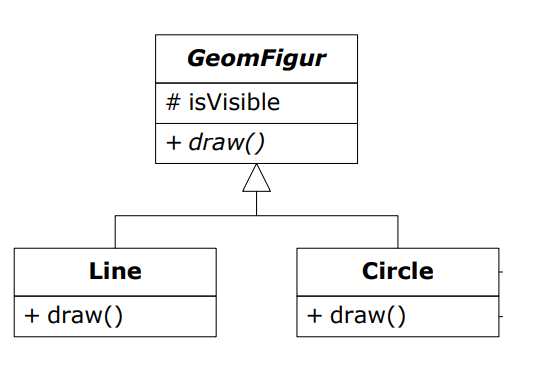
Холбоосыг классын тусламжтайгаар хэрэгжүүлэх

Хоорондох холбоосыг классаар хэрэгжүүлэхийг хэлнэ.

Хийсвэр класс:

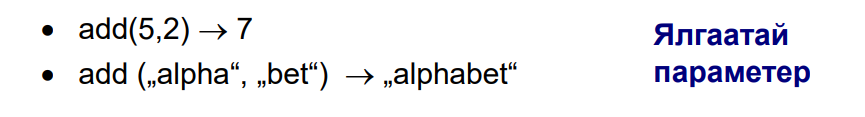
Хийсвэр функцыг агуулдаг объект үүсгэж болдоггүй зөвхөн удамшуулж ашигладаг класс юм. Олон хийсвэр үйлдэлтэй түүнийг дэд классуудад хэрэгжүүлдэг.

Объект хандлагат ЗНХ:



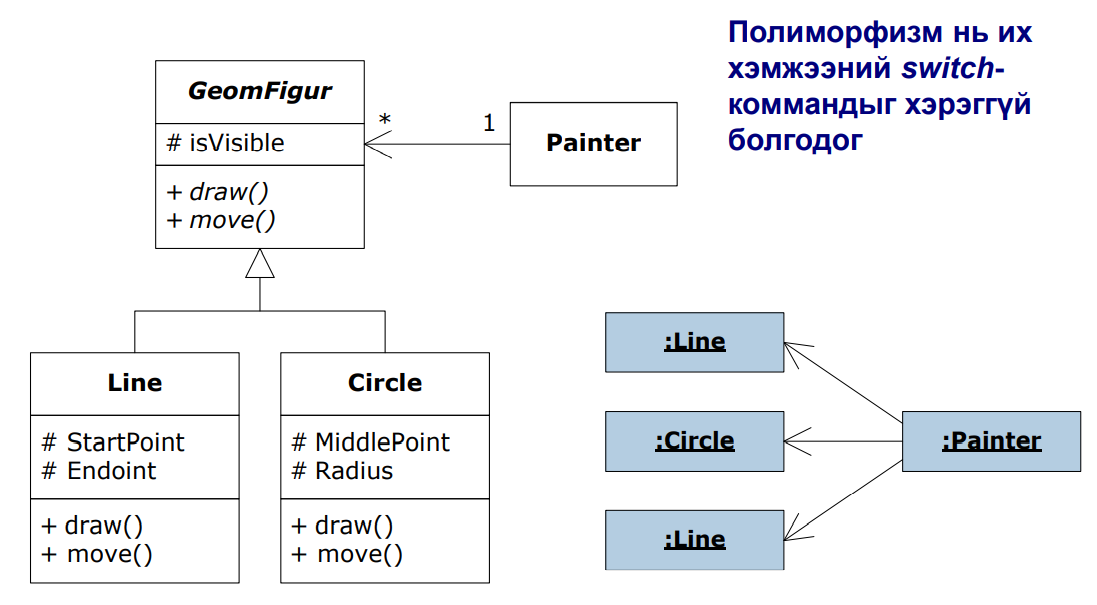
Үйлдлийг дахин тодорхойлох:

Нэг ижил нэртэй харин бичдэсээрээ ялгаатай утга хэрэгжүүлэлт юм.



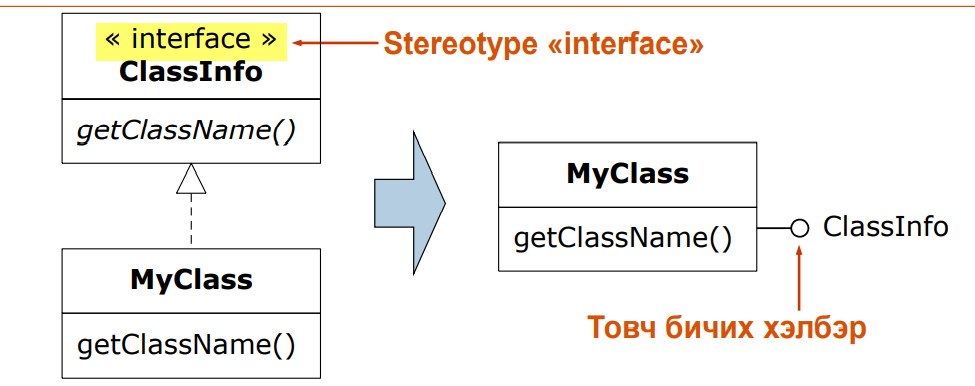
Полиморфизм

Олон төрлийн классын объектуудын хооронд нэг ижил мэдээг явуулахад объект болгон нь өөрийн онцлог, хэв маягаар хүлээх авахыг хэлнэ.



Интерфэйс:

Интерфэйс бол класс хоорондын тохиролцоо юм.

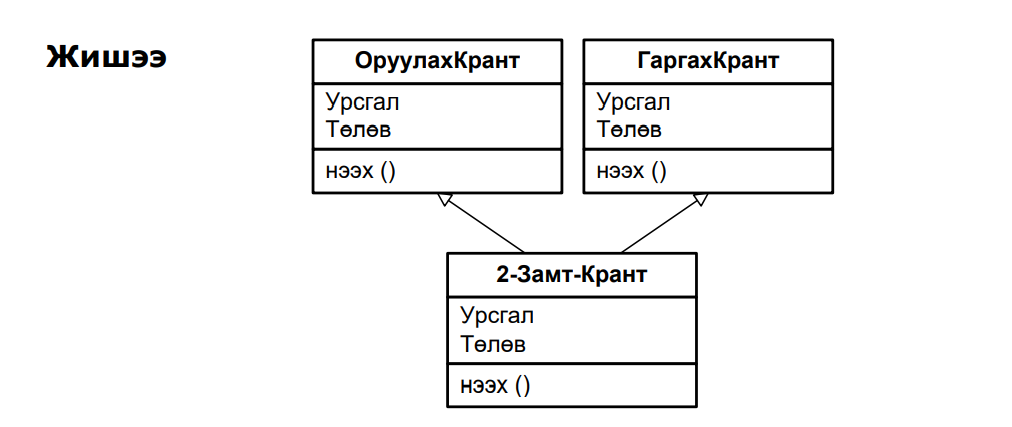


Энгийн болон нийлмэл удамшил

Энгийн удамшил бол мод бүтэц юм. Өөрөөр хэлбэл класс бүр ихдээ нэг эцэг класстай байна.

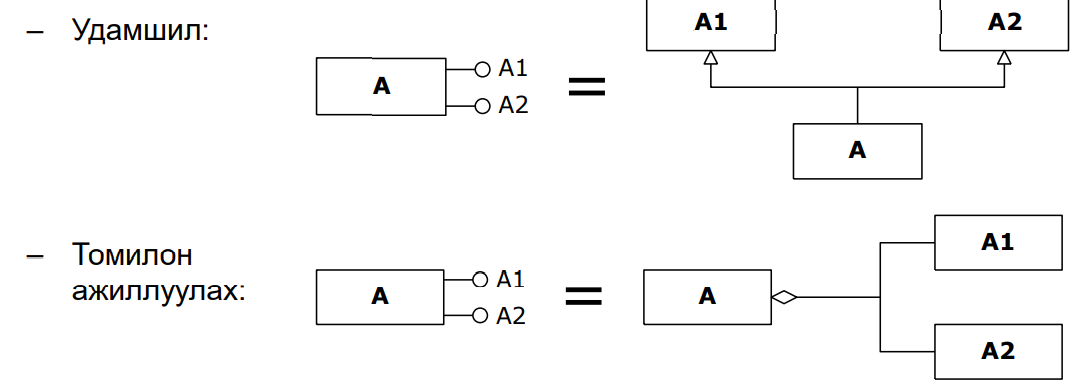
Нийлмэл удамшил:

Класс бүр нэгээс олон эцэг класстай байхыг хэлнэ. Сүлжээ бүтэцтэй байна.



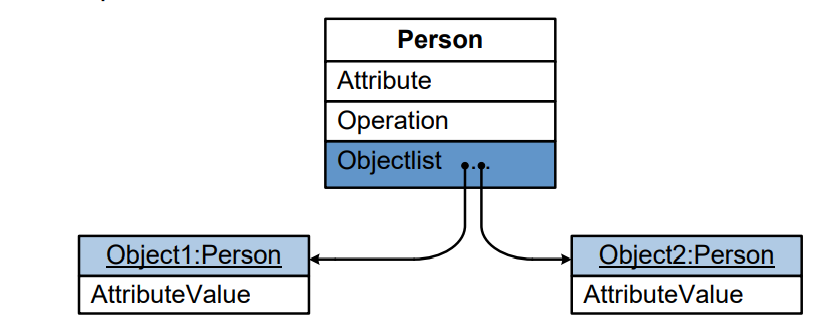
Томилон ажиллуулах нийлмэл удамшлын нэг хувилбар

Объект зурвасыг бие даан ганцаараа бүрэн хэмжээгээр тайлж чадахгүй, харин өөр нэгэн объект дамжуулах механизм юм. Бүрдлийн тусламжтайгаар тэрхүү салгасан классыг холбох

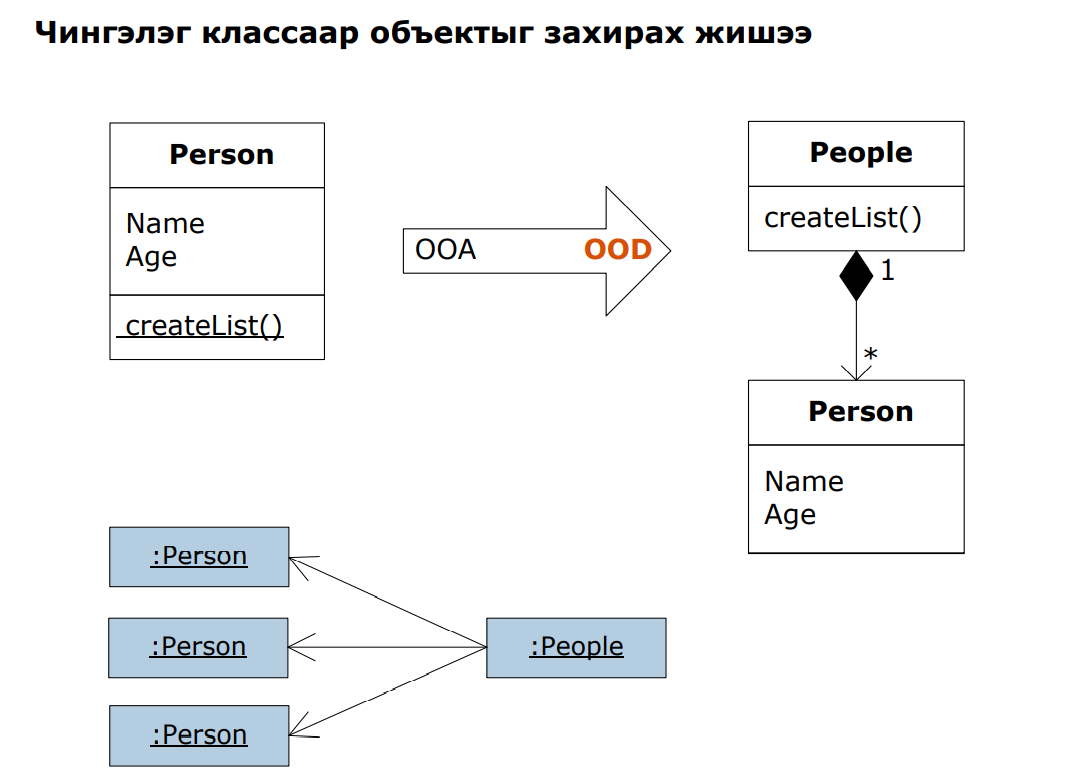


Объект захиран зохицуулалт:

Объект бүр ямар классаас үүссэн гэдгээ мэдэж байдаг, харин класс нь өөрийн объектоо мэддэггүй үүнийг програм дээр статик хувьсагчаар объектоо тоолж чадна. Объект захиран зохицуулалт гэдэг нь класс өөрийн объектыг үүсгэх болон устгахдаа таньж мэднэ гэсэн үг юм. Програм нь статик хувьсагчаар илэрхийлж болно.



Чингэлэг классын тусламжтайгаар объектыг захирах

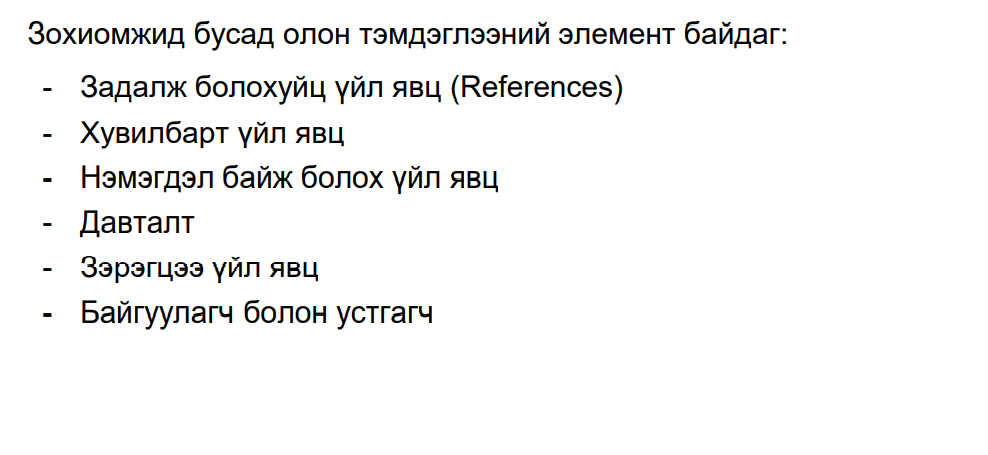


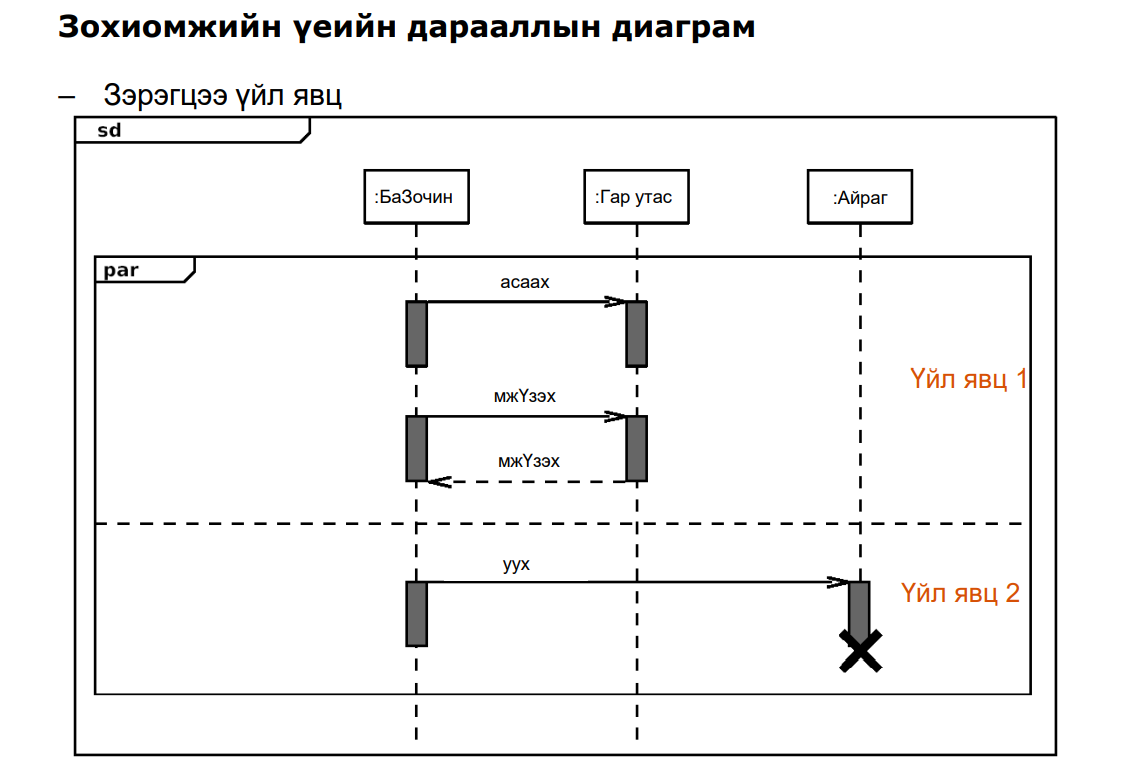
Объектын харилцан ажиллагаа:

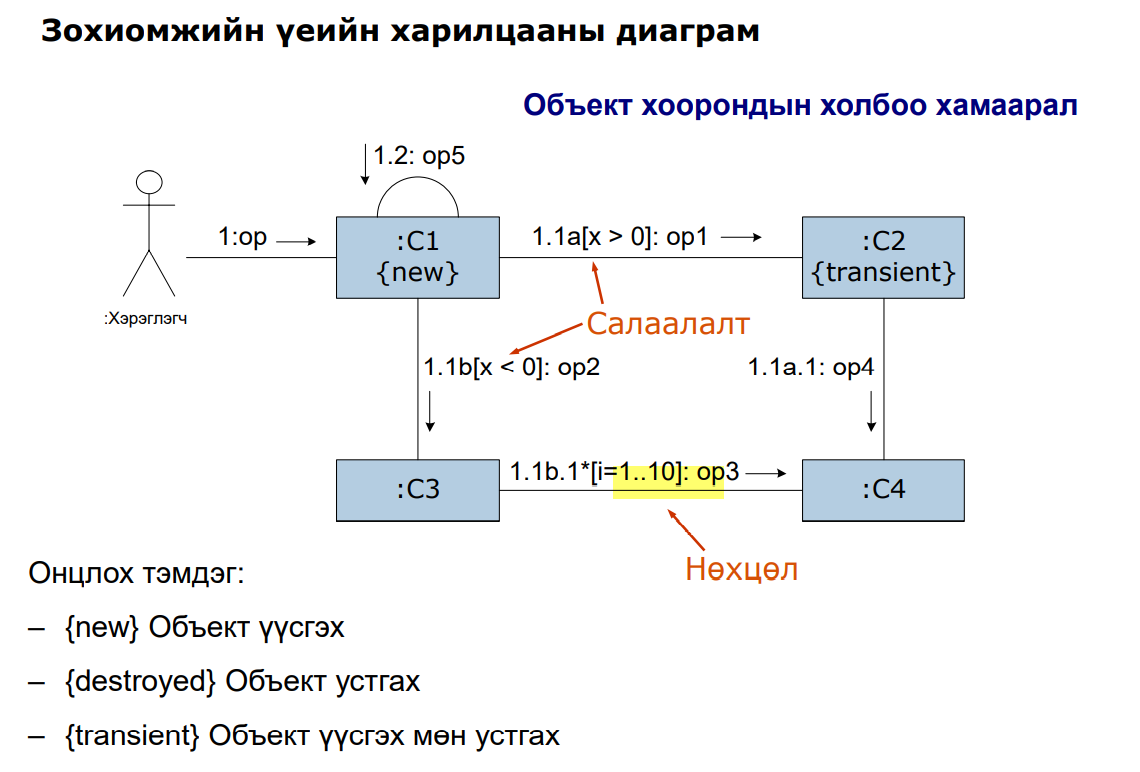
Дарааллын диаграм(Хугацааны талаас илүү харсан) болон харилцааны диаграммаас(Объект хоорондын харьцаа)тогтоно.

Зохиомжын үеийн дарааллын диаграм:

Шинжилгээний үеийн диаграммыг өргөтгөж өгдөг.







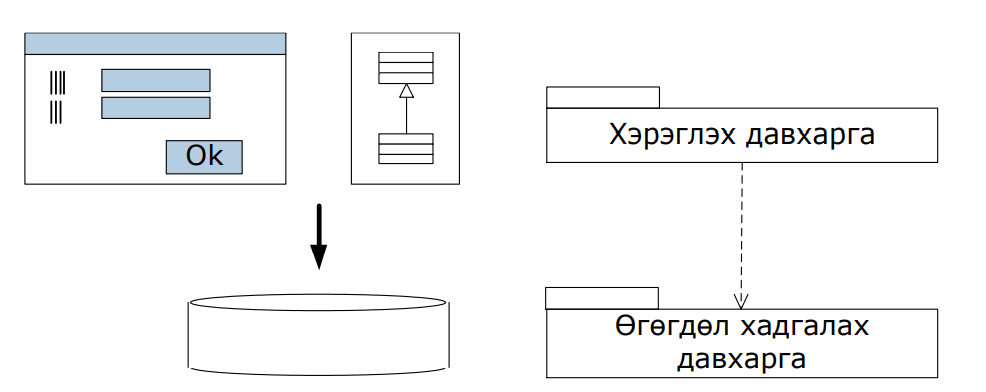
Архитектурын зохиомж:

Програм хангамжын архитектурын зорилго бол бизнес загварт тодорхой хязгаар нөхцөлийг авч үзэн зохиомжийн загварыг хөгжүүлэх юм.

Програм хангамж урт удаан оршин байхын тулд сайн архитектуртай байх явдал юм. Систем аль болох хоорондоо бага хамааралтай бүтэцлэх шаардлагатай байх ёстой. Архитектур нь хоёр давхаргат, гурван давхаргат, олон давхаргат архитектур байдаг.

Хоёр давхаргат архитектур нь

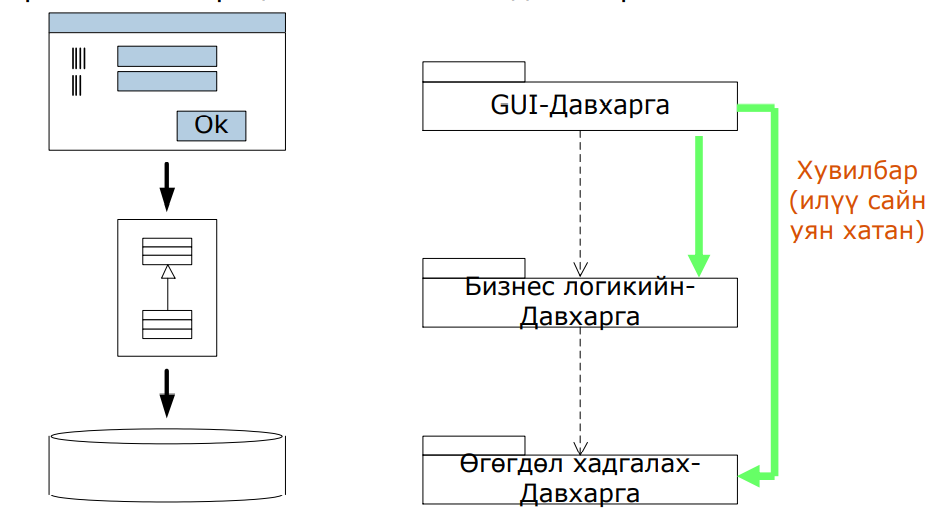
Зөвхөн хэрэглэгчийн харьцах хэсэг болон бизнес загвараас(Хэрэглэгчийн загвар) болон өгөгдлүүдийг хадгалах өгөгдлийн загвараас бүрддэг. Жишээ нь

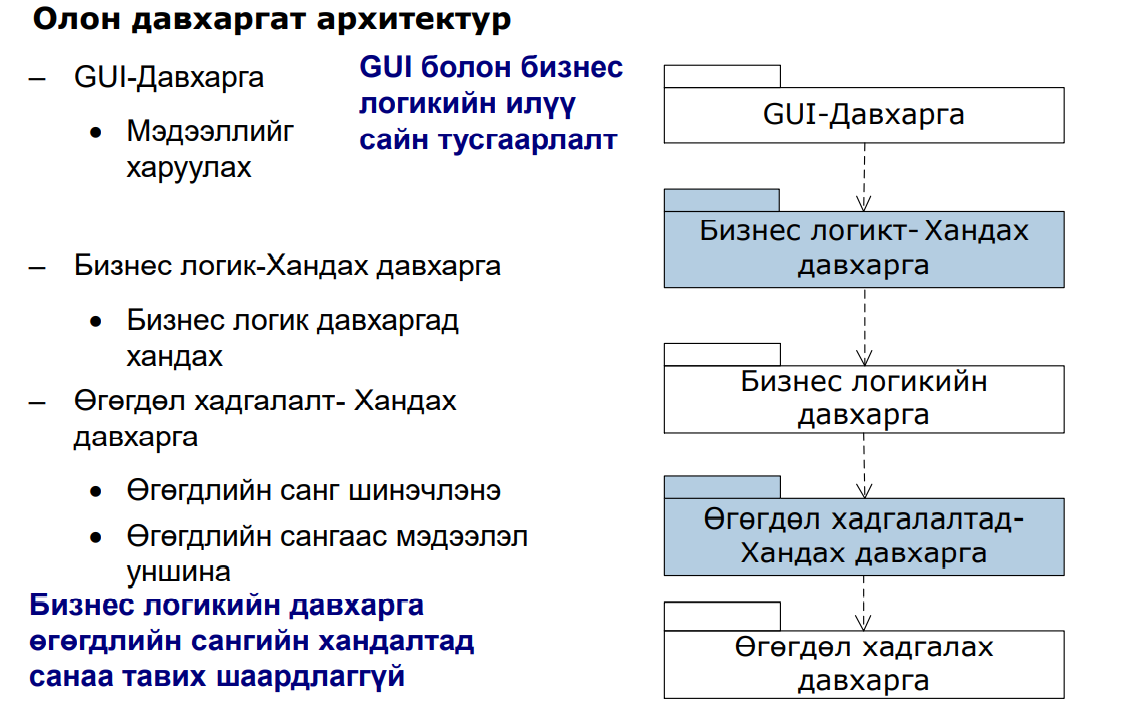


Аливаа зүйлд сул тал байдаг бизнесийн ойлголт хэрэглэх интерфэйё цугтаа байдаг. Системийг бага хамааралтай болтол жижиглэх ёстой байдаг.

Гурван давхаргат архитектур бол:

Бизнес ойлголт болон хэрэглэгчийн интерфэйсийг салгаж өгсөн. Хэрэглэгчийн харьцах хэсгийг өөрчлөхөд хялбар байдаг.



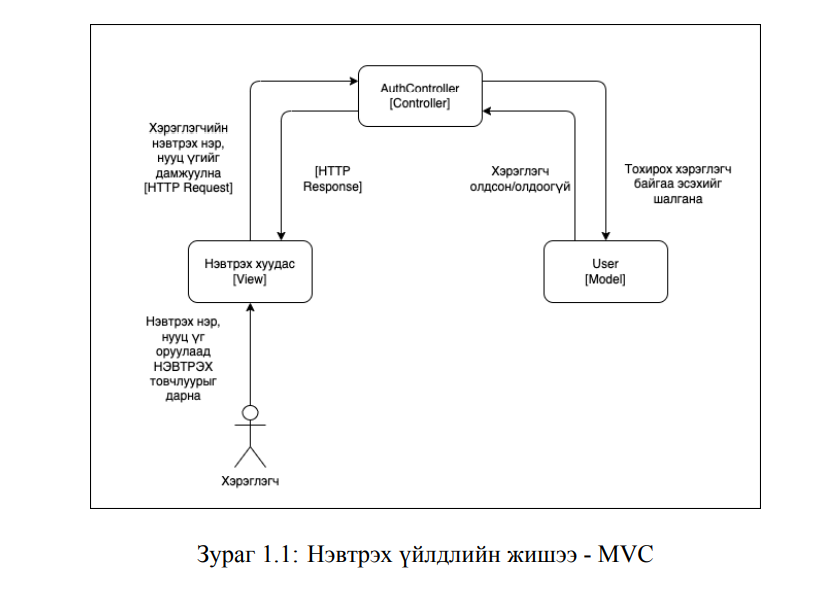


Нэмэлт мэдээлэл:

* 1. Програм хангамжийн архитектур Архитектур нь системийн зураг төсөл (blueprint)-г илэрхийлдэг. Энэ нь системийн нарийн төвөгтэй байдлыг зохицуулж, бүрэлдэхүүн хэсэг хоорондын харилцаа болон зохицуулалтын механизмыг зохион байгуулдаг. • Энэ нь гүйцэтгэл, аюулгүй байдал зэрэг чанарын нийтлэг шинжүүдийг оновчтой болгох замаар техникийн болон үйл ажиллагааны шаардлагуудад тусгагдсан бүтэцлэгдсэн шийдлийг тодорхойлдог. • Цаашилбал, энэ нь програм хангамжийн хөгжүүлэлттэй холбоотой байгууллагын талаарх чухал шийдвэрүүдийг багтаадаг бөгөөд эдгээр шийдвэр тус бүр нь чанар, тогтвортой байдал, гүйцэтгэл, эцсийн бүтээгдэхүүний нийт амжилтанд ихээхэн нөлөөлдөг.
  2. 1.2 MVC (Model-View-Controller) архитектур MVC архитектур нь хэрэглэгчийн интерфэйс, өгөгдөл, бизнес логикыг хэрэгжүүлсэн өргөн хэрэглэгддэг програм хангамжийн зохиомжийн хэв загвар юм. (software design pattern) Энэ нь програмын харагдах хэсэг болон бизнес логикыг тусад нь байлгадаг.
  3. 3 1. Model: Өгөгдлийн сангаас унших, бичих

2. View: Хэрэглэгчийн харьцах хэсэг

3. Controller: Хэрэглэгчийн харьцах хэсгийн хүсэлтийг боловсруулах, хамаарах үйлдлийг гүйцэтгэх, model-тэй харьцах Хэрэглэгч тодорхой үйлдлийг хийхэд жишээ нь “Нэвтрэх” товчлуурыг дарах үед viewс хамаарах controller-г дуудна. Controller хэрэглэгчийн нэвтрэх нэр, нууц үгтэй тохирох хэрэглэгчийн бичилт байгаа эсэхийг шалгах User model-н үйлдлийг дуудна. Хүсэлтийн үр дүнд view өөрчлөгдөнө (re-render).

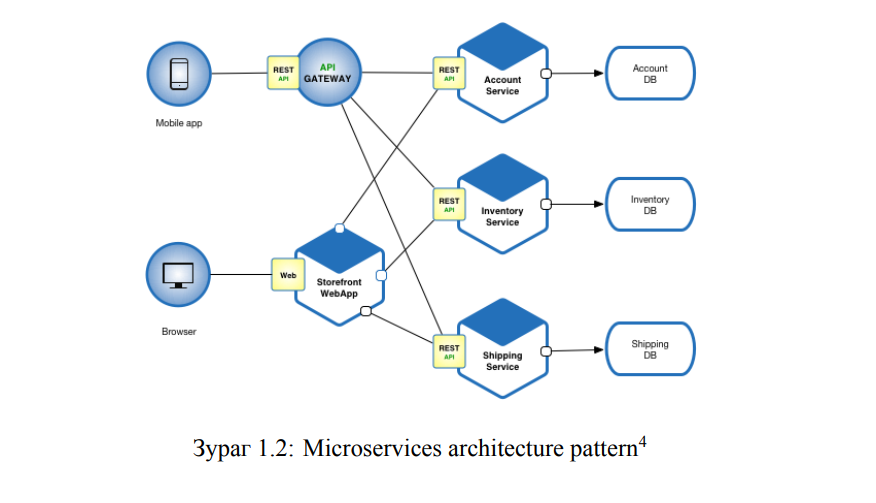


Давуу тал • Хөгжүүлэлтийн процесс хурдан. Гурван түвшинд хувааснаар хөгжүүлэгчид эдгээр хэсгүүдийг хувааж хөгжүүлэх боломжтой.

4 • Multiple views. Аливаа model-д зориулан хэд хэдэн view компонентуудыг зохион байгуулах боломжтой. • Asynchronous дэмждэг. MVC архитектур нь Javascript болон түүний framework-уудтай маш сайн зохицдог бөгөөд энэ нь хөгжүүлэгчдэд илүү хурдан ачаалах вебийг бүтээх боломжийг олгодог. • Өөрчлөхөд хялбар. Хэрэглэгчийн харьцах хэсэг болон бизнес логикийг салгаснаар аль нэг давхаргад шууд өөрчлөлт оруулах боломжтой. • MVC model нь формат хийгээгүй өгөгдлийг буцаадаг. Формат хийгээгүй өгөгдлийг буцааснаар MVC нь танд өөрийн view engine-г бий болгох боломжийг олгоно. Жишээлбэл, HTML ашиглан ямар ч төрлийн өгөгдлийг форматлах боломжтой боловч нь MVC-н тусламжтайгаар Macromedia Flash эсвэл Dream viewer ашиглан өгөгдлийг форматлах боломжтой. Ижил бүрэлдэхүүн хэсгүүдийг ямар ч интерфэйстэй дахин ашиглах боломжтой болгосноор хөгжүүлэгчдэд тустай. • SEO friendly. MVC платформ нь SEO-д ээлтэй веб програмуудыг хөгжүүлэхэд ихээхэн дэмжлэг үзүүлдэг. Тодорхой програмаас илүү олон хандалтыг бий болгохын тулд MVC нь SEO-д ээлтэй RESTful URL-уудыг хөгжүүлэх хялбар аргыг санал болгодог. • TTD (Test-Driven development)-г дэмждэг. MVC-ийн гол давуу тал нь туршилтын үйл явцыг ихээхэн хөнгөвчлөх явдал юм. Том хэмжээний програмыг олон түвшинд хувааж дибаг хийхэд хялбар болгодог ба ингэснээр програмыг нэгжийн тестийн хамтаар хөгжүүлэх боломжийг олгоно.1 1.2.2 Сул тал • Нарийн төвөгтэй байдал (Complexity). Цаг хугацаа өнгөрөхөд MVC архитектурын загвар нь шууд бусаар шинэчлэгддэг бөгөөд энэ нь шийдлийг улам хүндрүүлдэг. Нэмж дурдахад 1Benefits of using MVC, https://www.geeksforgeeks.org/benefit-of-using-mvc

5 хэрэглэгчийн интерфэйсийн код ихсэнээр дибаг хийхэд улам төвөгтэй болдог. • Ойлгоход төвөгтэй. MVC архитектур нь үнэн хэрэгтээ ойлгож суралцахад тийм ч хялбар биш. Нарийн төвөгтэй бүтэц болон давтамжтай шинэчлэлт нь системийн бүхий л бүрэлдэхүүн хэсэг болон хэрэглэгчийн харьцах хэсгийн код дээр тохиодог. • Ойр ойрхон өөрчилж, шинэчлэхэд зардал өндөр. Хэрэв model нь давтамжтайгаар өөрчлөгдөөд байвал view-г ч өөрчлөх хүсэлтүүд ихээр гарч болох юм. Цаашилбал view нь хэрэглэгчийн харьцах хэсэг тул өөрчлөхөд хугацаа их шаардагдаж, үр дүнд view компонентуудад шинэчлэл бүр дээр алдаа гарах магадлал бий болно. • Үйлдлүүд (methods) нь хориглолттой байх ёстой. MVC архитектурын 3 дахь бүрдэл хэсэг нь Controller. View компонентуудад тодорхой үзэгдэл тохиоход үйлдлүүдийг дуудаж ажиллуулдаг. Гэхдээ эдгээр нь хандалтын ямар ч хориглолтгүй байдаг.

1.2.3 Дүгнэлт MVC архитектурын давуу тал нь сул талаа нөхдөг. Сурах, ойлгоход хэцүү ч боломжгүй зүйл биш. Мөн нөгөөтэйгүүр MVC архитектурт суурилсан програмуудаар үйлчлүүлэгчдийнхээ хэрэгцээг хангадаг олон веб хөгжүүлэлтийн компаниуд байдаг. 1.3 Microservices архитектур Microservices архитектур нь том хэмжээний нарийн төвөгтэй програмыг хурдан, алдаагүй, илүү үр дүнтэй байдлаар хөгжүүлэхэд зориулагдсан бөгөөд системийн үйл ажиллагааг хэд хэдэн бие даасан сервисүүдэд хувааж зохиомжилсон системийн архитектур юм. Сервис бүр нь өөрийн өгөгдлийн сантай байх бөгөөд өөр хоорондоо энгийн API-н тусламжтай холбогдон ажилладаг. 3



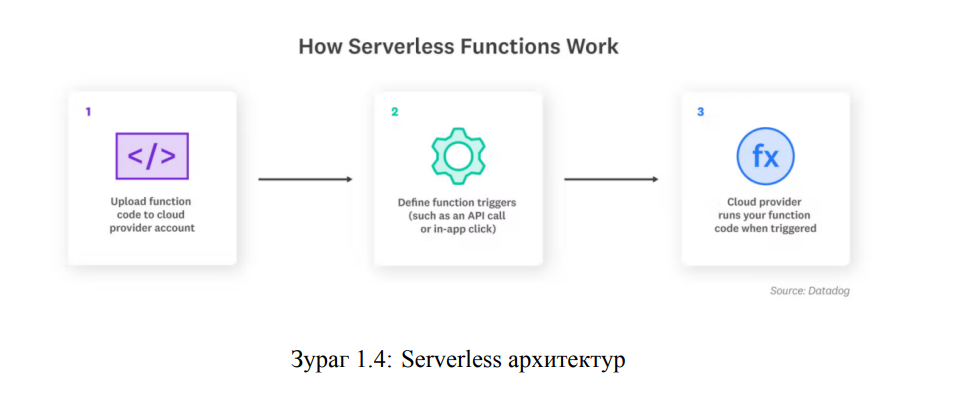
MVC архитектур дээр ашигласан хэрэглэгч нэвтрэх жишээн дээр харцгаая. Хэрэглэгч нь нэвтрэх үйлдэл нь AuthService-т хамаарах бөгөөд өөрийн AuthDB гэсэн өгөгдлийн сантай байна. Үүнд хэрэглэгчийн мэдээлэл хадгалагдана. Үүнээс гадна MedicalService гэсэн микросервисийг дүрсэлсэн байна.

Давуу тал • Бүтээмжийг нэмэгдүүлдэг (Improved productivity). Системийг жижиг хэсгүүдэд хуваан хөгжүүлэлт хийж, бие даан ажиллуулах боломжтой байдаг. Хэсэг бүрийг ялгаатай хэл, технологи ашиглан хэрэгжүүлж болно. • Тэсвэрлэх чадвар өндөр (Better resiliency). Microservice дээр тулгуурласан архитектур нь ажиллагааны явцад гарах асуудлыг илрүүлэх, түүнийг шийдвэрлэх процессыг илүү хялбар болгосон. Алдаа гарсан жижиг хэсгийг олж илрүүлж засаад зөвхөн тухайн дэд сервисийг дахин ажиллуулахад /redeploy/ хангалттай. • Өргөтгөх боломж (Better scalability). Жижиг хэсгүүд дээрх өөрчлөлт, сайжруулалтыг мөн адил бие даасан байдалтайгаар хөгжүүлдэг. • Бизнесийн үйл ажиллагааг оновчтой болгодог (Optimize business functionality). Бүхэл систем биш дэд хэсэг дээр илүү анхаарч, бүрэлдэхүүн хэсгүүдийн хэрэгцээг хангаснаар бизнесийн үйл ажиллагааг илүү оновчтой болгодог.5

Сул тал • Хөгжүүлэлтийн багт Monolithic програмуудаас илүү олон төрлийн нарийн төвөгтэй байдлыг бий болгодог. – Нэгт. Сервис хоорондын харилцаа нь төвөгтэй байж болно. Програм нь хэдэн арав эсвэл хэдэн зуун ялгаатай сервисүүдийг агуулдаг бөгөөд тэдгээр нь бүгд хоорондоо аюулгүй холбогдож ажиллах ёстой байдаг. – Хоёрт. Дибаг хийх нь илүү төвөгтэй. Програм нь хэд хэдэн microservice-үүдийг агуулах бөгөөд тэдгээр нь тус бүрдээ log-уудтай, асуудлыг олж илрүүлэх нь төвөгтэй байж болох юм.

Гуравт. Unit тестийн хувьд хялбар байж болох ч integration тестийн хувьд тийм биш юм. Компонентууд нь тархсан байдалтай байдгаас хөгжүүлэгчид өөрсдийн төхөөрөмж дээр системийг бүхэлд нь тестэлж чаддаггүй. • Интерфэйсийн хяналт маш чухал. – Microservice бүр нь өөрийн API-тай бөгөөд програм нь эдгээр дээр тулгуурлан тогтвортой ажилладаг. Microservice-тэй харьцаж буй гадаад системд нөлөөлөхгүйгээр хялбархан өөрчлөлт хийх боломжтой ч API (интерфэйс) дээрх өөрчлөлт нь түүнтэй холбогдож байгаа бусад системүүдэд нөлөө үзүүлвэл энэ нь асуудлыг бий болгоно. – Microservices архитектурын загвар нь системийн үйл ажиллагаанд чухал ач холбогдолтой олон тооны API-г бий болгодог тул интерфэйсийн хяналт нь чухал юм. • Урьдчилсан зардал (up-front cost) өндөр байж болно. – Танай байгууллагад Microservices архитектурыг ажиллуулахын тулд танд аюулгүй байдал, засвар үйлчилгээний дэмжлэг бүхий хангалттай байршуулах дэд бүтцээс гадна бүх үйлчилгээг ойлгож, удирддаг чадварлаг хөгжүүлэлтийн багууд хэрэгтэй. – Хэрэв танд эдгээр зүйлс байгаа бол Microservice-т шилжихэд гарах зардал бага байж болох ч одоогоор Monolithic архитектурыг ажиллуулж байгаа ихэнх аж ахуйн нэгжүүд шилжихийн тулд шинэ дэд бүтэц болон хөгжүүлэгчийн нөөцөд хөрөнгө оруулалт хийх шаардлагатай болно.

6 1.3.3 Дүгнэлт Microservice архитектурын давуу болон сул талууд нь уламжлалт Monolithic архитектураас эрс ялгаатай бөгөөд энэ загвар нь бүх байгууллагад тохиромжтой биш юм. Гэсэн хэдий ч тодорхой шалтгаануудын улмаас энэхүү модульчлагдсан архитектурт шилжиж байна. Энэ 6The Disadvantages of Microservices, https://solace.com/blog/microservices-advantages-and-disadvantages 9 нь илүү олон байгууллагууд илүү хурдан, хялбар, уян хатан програмын хэрэгцээг бий болгож байгаа явдал юм. 1.4 Serverless архитектур Serverless архитектур нь програм хангамжийн дэд бүтцийг (infrastructure) удирдан зохицуулахгүйгээр хөгжүүлэлтийг гүйцэтгэж, системийг ажиллуулах арга. Систем нь сервер дээр ажиллах хэдий ч серверийн бүхий л менежментийг AWS, Firebase, Azure зэрэг cloud platformууд хариуцдаг



Давуу тал • Хямд. Зөвхөн өөрт нийцэх cloud үйлчилгээг худалдан авч ашиглахад л хангалттай. • Уян хатан байдал. Serverless загвар нь таны оролцоогүйгээр өргөжих боломжтой. • Нарийвчлал. Хөгжүүлэгчид үүнийг дэмждэг back-end архитектураас илүүтэй тодорхой функц дээр анхаарлаа төвлөрүүлэх боломжтой. Энэ нь илүү цэвэрхэн кодыг бий болгодог. • Хурд. Хөгжүүлэгчид серверийн багтаамжийг тооцоолох, хуваарилах дээр санаа зовж хугацаа алдахгүй.9 1.4.2 Сул тал • Аюулгүй байдал. Өөрийн өгөгдөл мэдээллийг бусад компанид хадгалуулах явдал. Serverless шийдлийг ашигладаггүй байгууллагуудын 60 гаруй хувь нь энэ загварын аюулгүй байдлаас болдог. • Нууцлал. Таны мэдээлэл бусад хүмүүс ашиглаж болохуйц cloud орчинд байршдаг. • Нарийн төвөгтэй байдал. Хэрэв ямар нэг зүйл зохих ёсоор ажиллахгүй байвал асуудлыг олж тогтооход хэцүү. • Гэрээ. Зарим борлуулагчид үйлчлүүлэгчтэйгээ урт хугацааны гэрээ байгуулдаг.1

Дүгнэлт

Энэхүү лекц 10 болон 11ийн тэмдэглэл бичснээр лекцийн ойлголтуудаа бататгаж чадсан. Хамгийн гол чухал ойлголт болох зохиомж нь шинжилгээний үеийн ялгарах ойлголтууд илүү хөгжүүлэгч талаас харж класс диаграмм байгуулж сурсан. Сайн архитектур бол програм хангамжын системийн урт наслах үндэс байдаг.