**Архитектурын хэлбэр**

# Програм хангамжийн архитектур (Software Architecture)

**2012**

С. Бадрал

# Агуулга

- Нэр томьёо
- Төвшинт загвар
- Pipes & Filters
- Объект хандалтат зохион байгуулалт
- Peer to Peer
- Repository/Blackboard
- SOA
- Quasar архитектур
- Дүгнэлт

# Нэр томьёо

- Architecture style = Архитектурын хэлбэр
- Layered pattern = Төвшинт загвар
- Rule based = Дүрэмд суурилсан
- Disributed system = Тархмал систем

# Архитектурын хэлбэрийн тодорхойлолт

- Архитектурын хэлбэр нь ПХ системийн бүтцийн суурь зохион байгуулалтын схемийг илэрхийлдэг.
- Урьдчилан тодорхойлсон дэд системүүдийг үүрэг хариуцлага тус бүрийнх нь хамтаар нийлүүлнэ. Мөн тэдгээрийн хоорондын холбоог зохион байгуулах дүрэм ба зааврыг тусгана.
- Архитектурын хэлбэр ↔ Зохиомжийн загвар

# Түгээмэл архитектурын хэлбэрүүд

- Өгөгдлийн урсгал
  - Batch дараалал
  - Pipes & Filters
- Call & Return
  - Main program & subroutine
  - ОХ системүүд
  - Иерарх төвшинт
- Хамааралгүй компенентүүд
  - Холбооны процессүүд (Communication processes)
  - Үзэгдэлт системүүд (event based)
- Виртуал машинууд
  - Interpreters
  - Дүрэмд суурилсан системүүд (Rule based)

# Түгээмэл архитектурын хэлбэрүүд

Өгөгдөлд төвлөрсөн системүүд

- Өгөгдлийн сангууд
- Гипертекст системүүд
- Blackboards / Repository

- Тархмал системүүд
  - Клиент сервер
  - Сервис хандалтат
  - Peer-to-Peer

- Архитектурын хэлбэрүүдийг хослуулах боломжтой.
  - „төвшинт систем“ (layered system) объект хандалтаар хэрэгжиж дүрэмд суурилсан (rule based system) аргачлалыг хэрэглэж болно.

# Төвшинт загвар (Layered pattern)

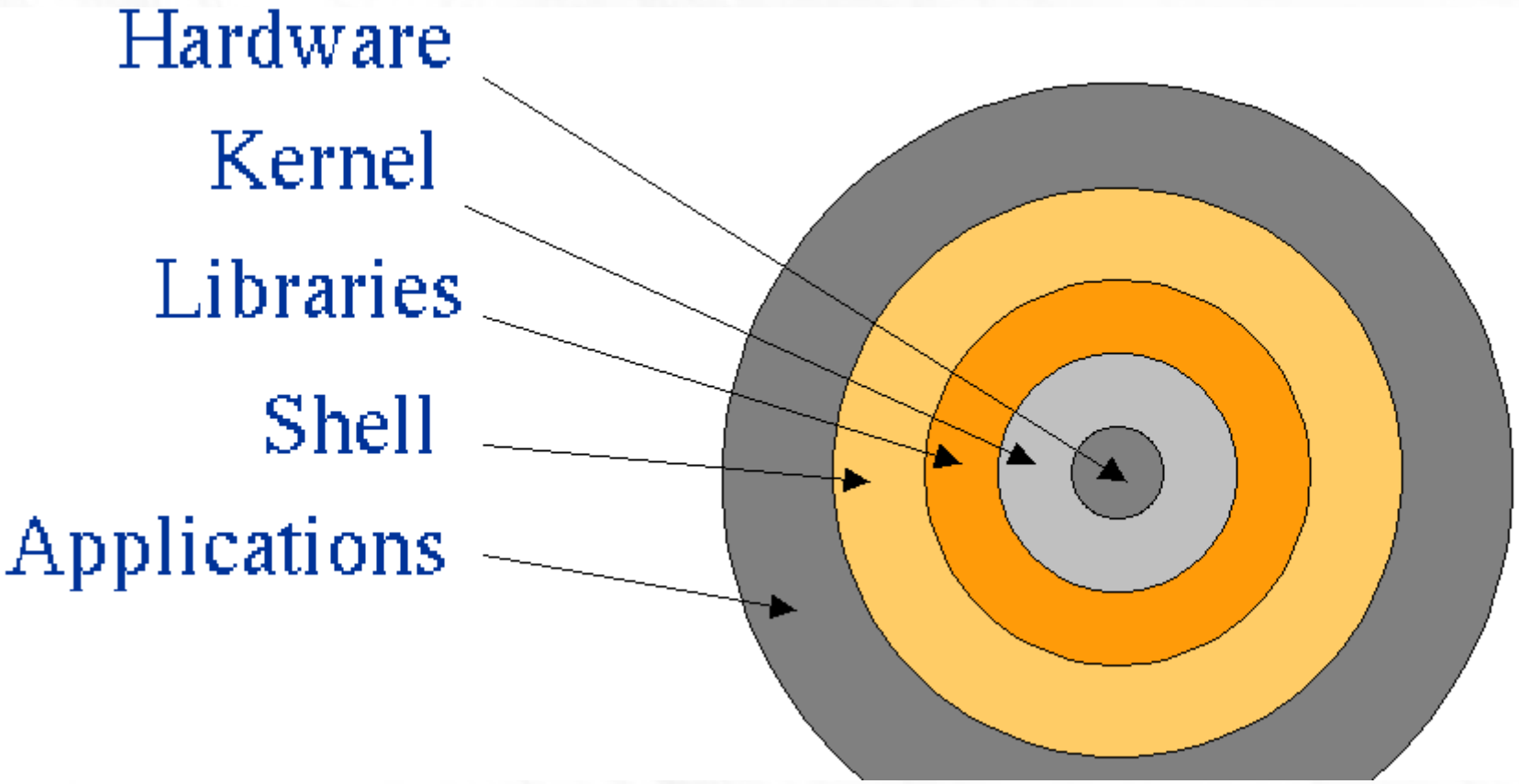
- Компонентын төрлүүдийн олонлог
  - Объектууд төвшинд зохион байгуулагдана.
- Тэдгээр компонентүүдийн тополог байрлал
  - Дээр дээрээ давхарлагдсан төвшингүүд
- Утга зүйн нөхцөлүүд / Утгат дүрмүүд
  - Ижил эсвэл доод төвшиний сервисүүд дуудагдана харин дээд төвшнийх дуудагдахгүй.
- Боломжит холбоосуудын олонлог
  - Процедурын дуудалт, RMI

# Төвшинт загварын жишээ

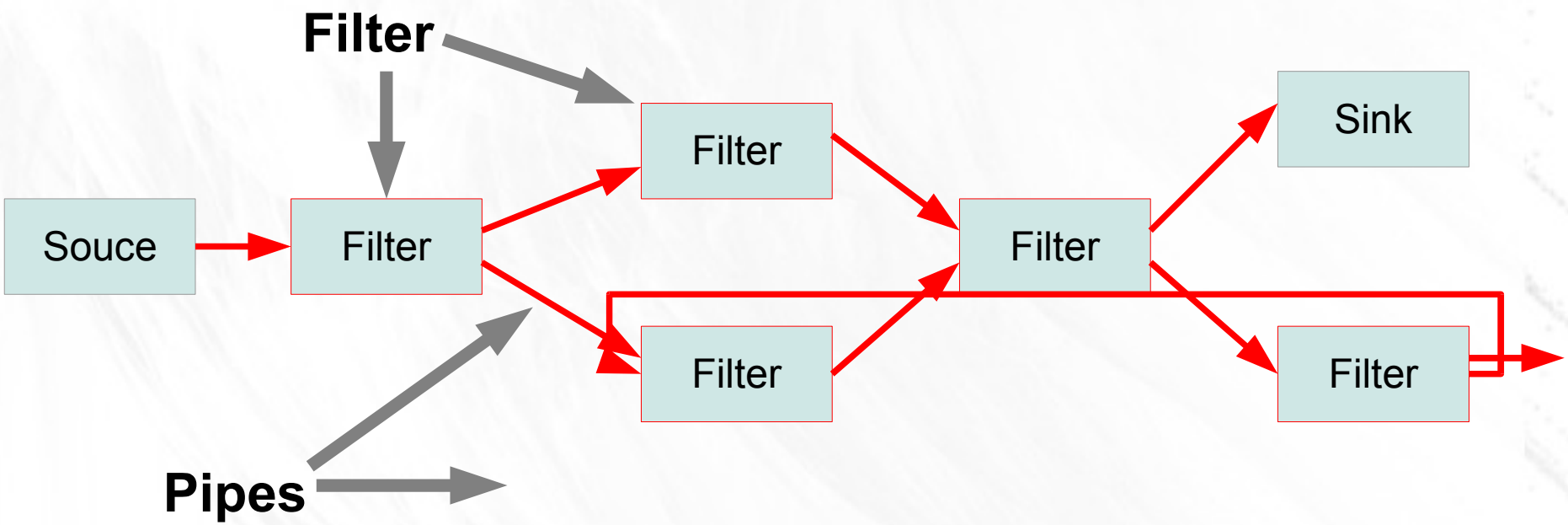
## Сүлжээний протокол

# Төвшинт загварын жишээ

- Үйлдлийн систем



# Pipes & Filters

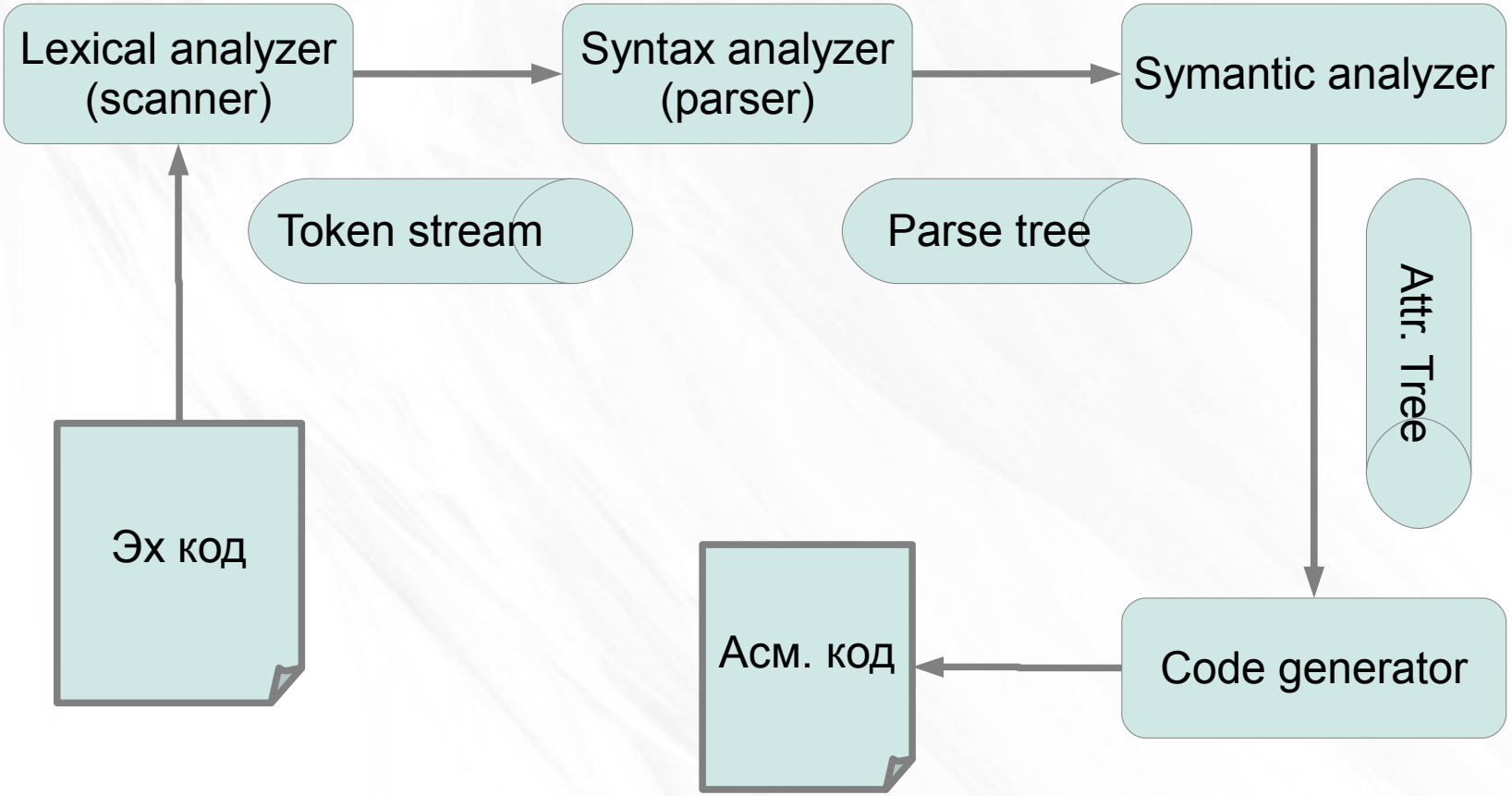



**Сонгодог жишээ нь: Юникс шелл, компилятор**

**cat text1.txt | sed 's/mongol/mongolian/g' > text1.txt.new**

# Pipes & Filters

- Компилятор



# Pipes & Filters

Давуу тал

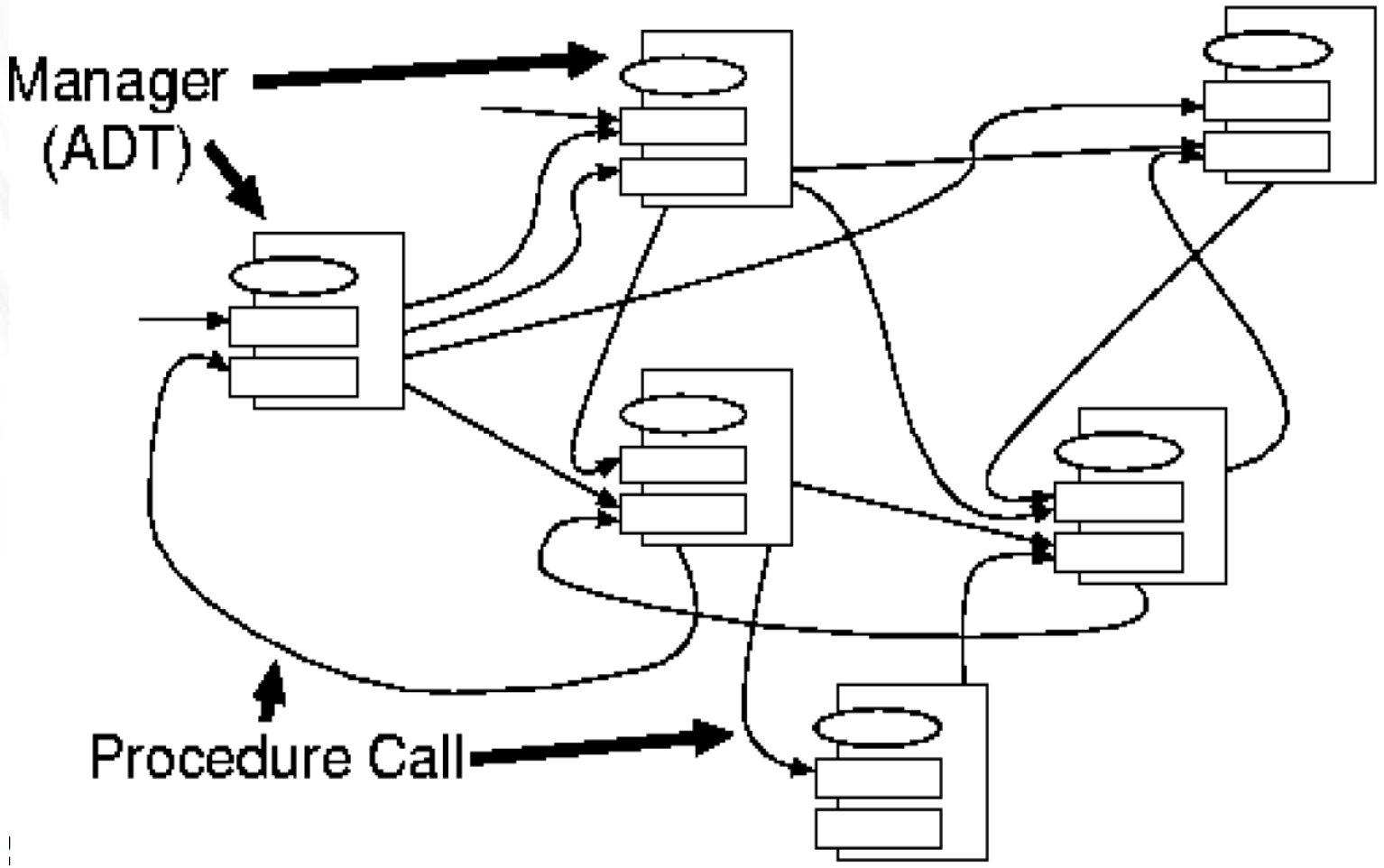
- Системийн харьцааг ойлгоход хялбар
- Шүүлтүүрүүдийг дахин хэрэглэх боломжтой
- Системийн арчлах болон өргөтгөхөд хялбар (шинэ шүүлтүүр нэмэх, хуучин шүүлтүүрийг сайжруулах)
- Бүтээмж ба түгжрэлийн анализ хийх боломжтой
- Зэрэг ажиллагааг дэмжинэ

Сул тал

- Интерактив бус
- Хүчин чадлын хувьд төдийлөн оновчтой бус

# ОХ зохион байгуулалт

Бүтэц



## ОХ зохион байгуулалт

Үндсэн шинж

- Объектын төлөв хайрцаглагдсан
- Объект нь өөрийн төлөвт хариуцлагатай
- Системийн төлөв нь олон объектоос тогтоно

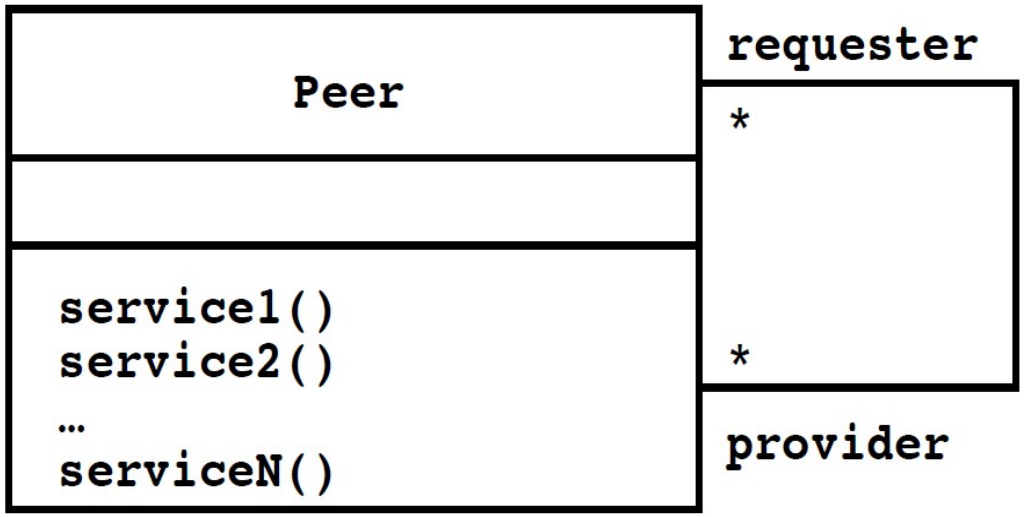
Сул тал

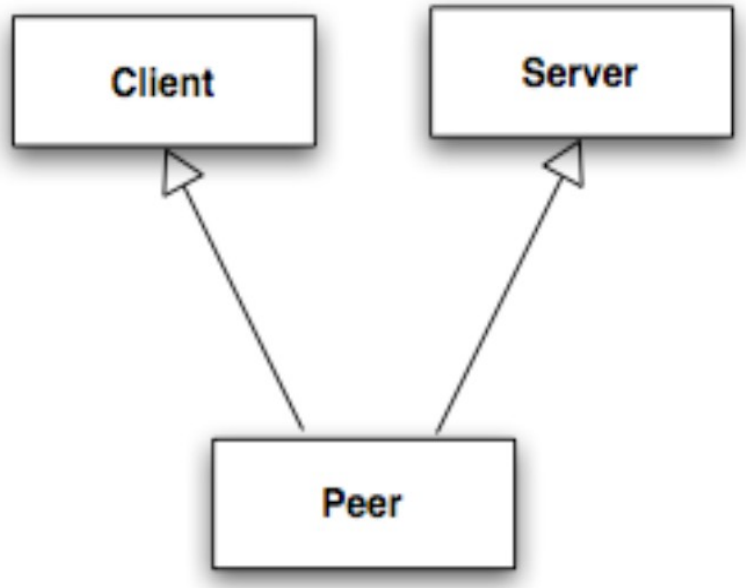
- Хэтэрхий сүлжилдсэн: Олон класс өөр классаас хамааралтай

## Peer to peer

Клиент сервер архитектурын ерөнхийлөл

Клиент нь сервер ч клиент ч байж чадна

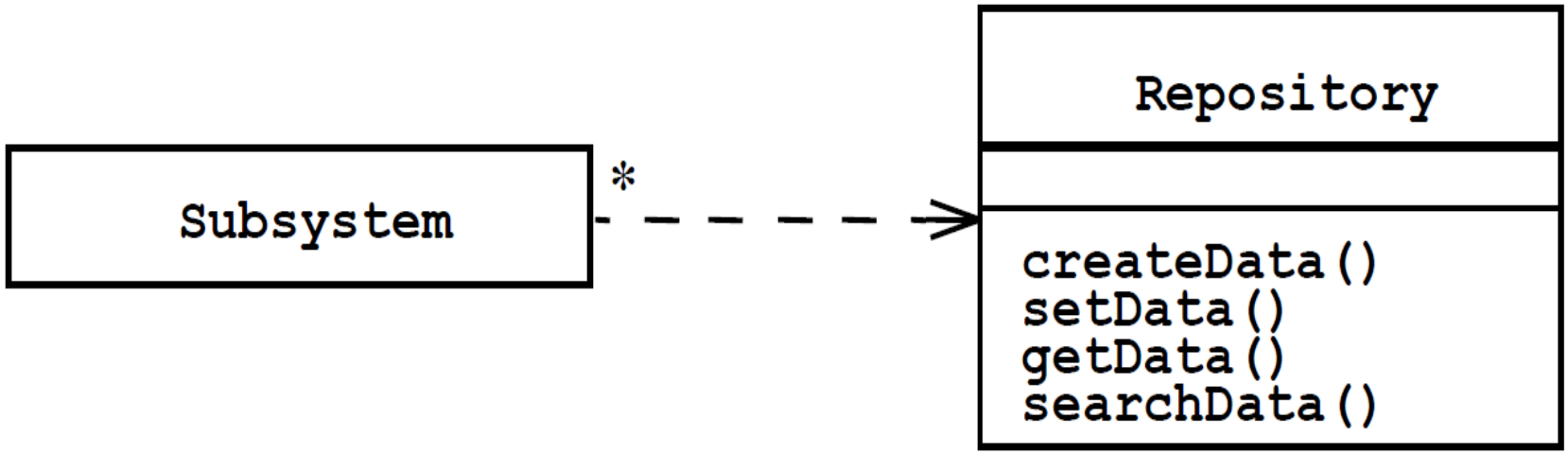



Жишээ нь: Сүлжээний протоколл

# Repository/Blackboard

- Дэд системүүд өгөгдлөө Repository хэмээх төв сангаар хандаж, өөрчилж, солилцоно
- Дэд систем бүр өөрийн өгөгдлөө эзэмшиж, шаардлагатай бол бусад дэд систем руу дамжуулна
- Дэд системүүд зөвхөн Repository-р харилцана (loosely coupled)
- Үйлдлийн удирдлага нь Repository-р бичигдэнэ.



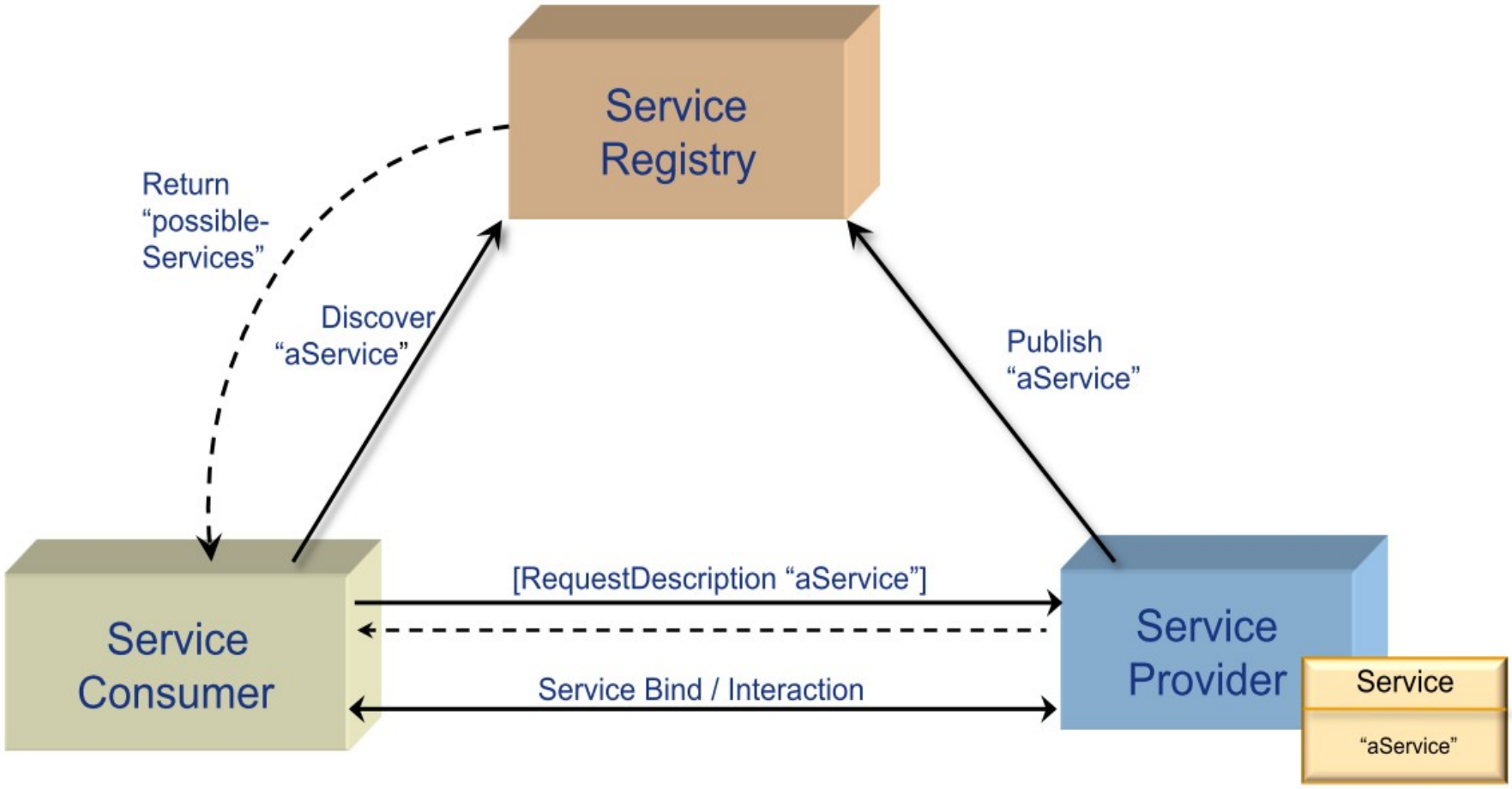
# Repository/Blackboard хэрэгжүүлэлт

- Өгөгдөл хадгалагч руу хандахыг Singleton ба / эсвэл Фасад паттернаар (зохиомжийн загвар) хийж болно
- Хадгалагч руу бичихдээ шаардлагатай дэд систем рүү мэдэгдэх хэсгийг ажиглагч паттернаар хэрэгжүүлж болно.
- Өгөгдөл хадгалагчийн Composite паттернаар хэрэгжүүлж болно
- Repository-н буцаах (undo) үйлдлийг memento паттернаар хэрэгжүүлж болно

- Өгөгдөл хадгалагч -> класс диаграмм.
- Өгөгдөлд хандах протоколлыг -> явцын диаграмм

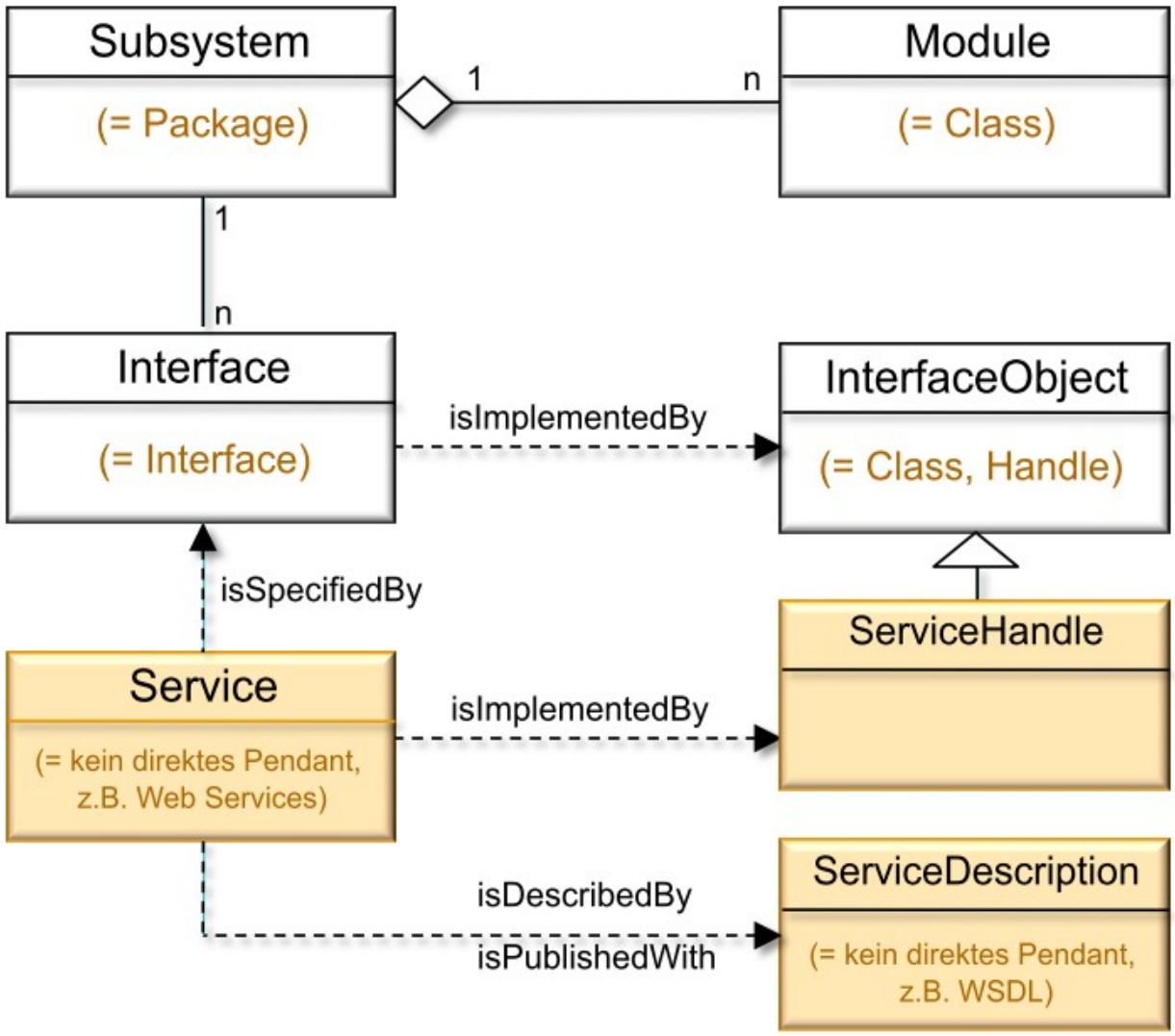
# SOA

Төвөгтэй янз бүрийн систем, програм, худалдааны функц Процессуудыг (Enterprise Application Integration, EAI) холбох зорилго бүхий архитектурын хэлбэр.



# SOA

Сервисүүд дэд системийн зурвасуудад залгагдана.



# SOA

- Сервисүүд нь автономит
  - Сервис тус бүр хоорондоо хамааралгүйгээр арчилах, хөгжүүлэх, суурилуулах, хувилбар үүсгэх боломжтой
- Сервисүүд нь тараагдах боломжтой
  - Сервис тус бүр холболтын протоколл нь л дэмжиж байвал локал эсвэл алс гэх мэт сүлжээний хаана ч байрлах боломжтой.
- Сервисүүд нь бага хэрэгдсэн
  - Сервисүүд нь бие биеээсээ хамааралгүй бөгөөд тус бүр зурвас ашиглаж байхдаа програмыг таслахгүйгээр шинэчлэгдэж өөрчлөгдөх боломжтой.
- Сервисүүд нь класс бус схем ба гэрээг хамтран хэрэглэдэг
  - Сервисүүд харилцахдаа класс бус схем, гэрээг ашигладаг
- Нийцтэй байдал нь бодлого дээр суурилдаг.
  - Бодлого нь энд дамжуулал, протоколл, аюулгүй байдал

## Quasar архитектур

SD&M германы толгой захиалгын ПХ үйлдвэрлэгчдийн нэг

QUASAR нь SD&M-н стандарт архитектур

Үндсэн шинжүүд

- ПХ-н ангилал (blood groups)
- Компонент хандалтат хөгжүүлэх процесс
- A-TI-I архитектурын хэлбэр
- http://www.openquasar.de

## Програм хангамжийн цусны бүлэг

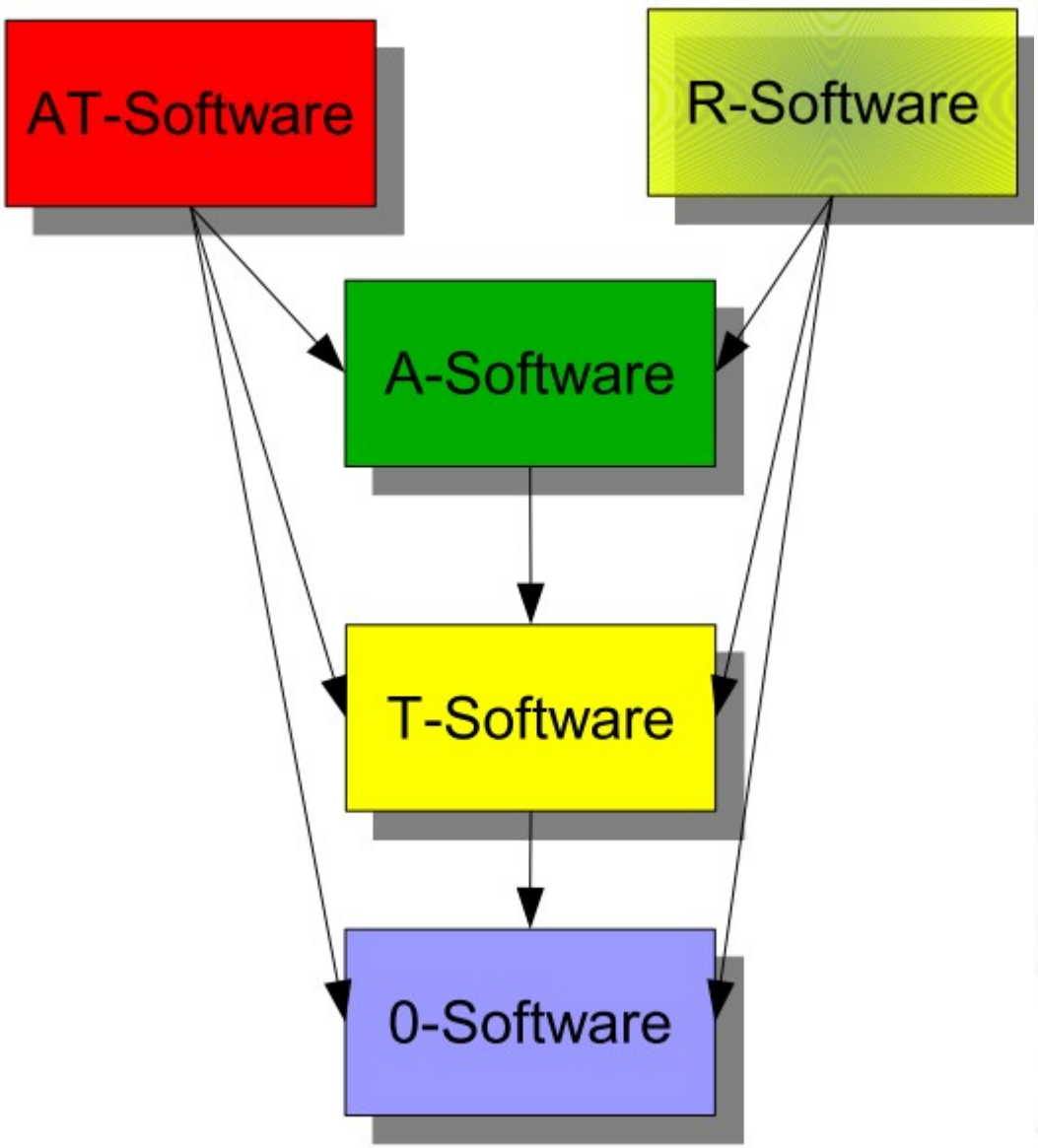
Дахин хэрэглэгдэх байдлаар нь ПХ ангилах

- **0**: Хэрэглээ ба технологийн хараат бус байдал
  - JDK цуглуулга, C++ STL, GNU regexp
- **A**: Хэрэглээ эсвэл домэйнд хамаатай. Домэйн моделийн үндэс
  - Клиент, үйлчлүүлэгч, …
- **T**: Технологи хандалтат API, програмаас бус технологоос хараат байдал
  - JDBC, CORBA CosNaming
- **AT**: Хэрэглээ ба технологоос хараат
  - Зайлсхийх хэрэгтэй: Арчлах, дахин хэрэглэхэд хэцүү

## Програм хангамжийн цусны бүлэг

- **R**: Бизнес объектын дүрслэлийн гадаад дүрслэл рүү эсвэл эсрэгээр өөрчлөлт
  - Serialization, deserialization, encryption, decryption, packing, unpacking
  - Нэг хэлийн объктыг нөгөө хэл уруу (Ж.нь, Java → Cobol)

# Програм хангамжийн цусны бүлэг



# Архитектурын элементүүд

- **0-зурвасууд зөвхөн техникийн төрлүүдийг агуулна. (strings, collections etc)**
  - Дахин хэрэглэхэд амар
- **A-зурвасууд домэйн төрлийг агуулна (account, bill,..)**
  - A-компонентүүд програмын логик ба өгөгдлийн сангийн төвшинд амьдарна
  - Дахин хэрэглэхэд хүнд
- **T-зурвасууд техникийн API-г агуулна**
  - Хаа сайгүй хэрэгтэй
- **R-зурвасууд хоёуланг нь агуулна. Учир нь тэд дүрслэлийг өөрчилнө**
  - Middleware ба өгөгдлийн төвшинд шаардлагатай
  - A-н нэгэн тусгай төрөл
  - Ихэнхдээ тодорхойлолтоос үүсдэг. Дахин хэрэглэгдэхгүй ч дахин үүсгэгдэх боломжтой
    - XML tools, e.g., XMI (model interchange)
    - OMG MOF tools (Model-driven development)

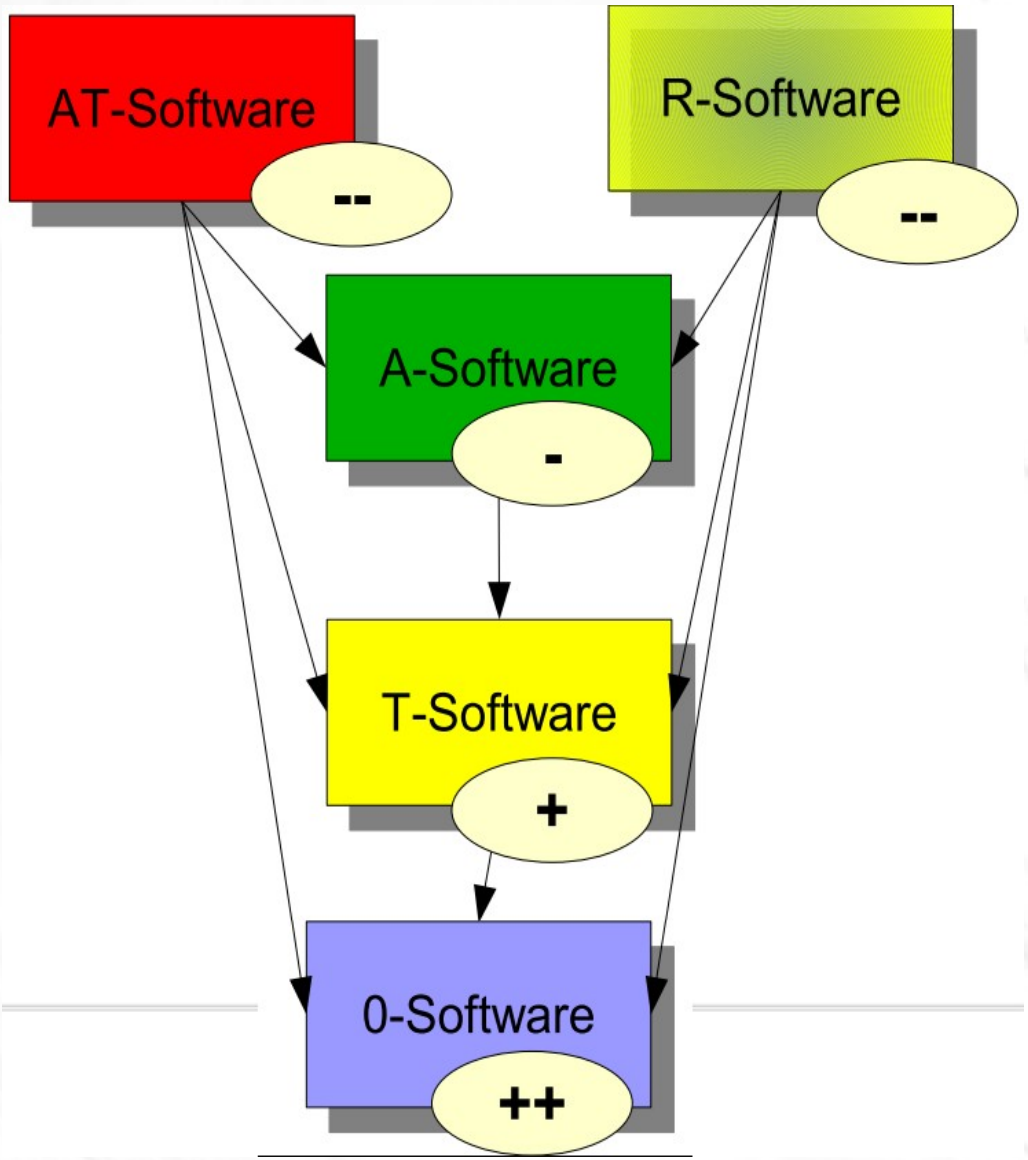
# Цусны бүлгийн зорилго

Цусны бүлгийн зорилго нь тус тусад нь илүү дахин хэрэглэхийн тулд хэрэглээ ба техникийг нэг нэгнээс аль болох хол салгах

Siedersleben's Цусны бүлгийн хууль
Зурвас, класс, компонент бүр зөвхөн нэг ПХ-н ангилалд хамаарна.

# Цусны бүлгийн зорилго

- Дахин хэрэглэх боломж нь 0-с АТ рүү буурна
  - Техникт чиглэсэн компонентууд нь хялбар дахин хэрэглэгддэг
  - Хэрэглээнийх нь хэцүү
  - Асуудалтай нь АТ
- Цусны бүлэг нь BCED-архитектурын бүх төвшинг хамарна
  - Төвшин бүрд техник, хэрэглээ, дүрслэл байдаг



# Цусны бүлгийн зорилго

- А-компонентийг Т-с дуудах нь аюултай
  - Цикл бус А-Т USES-холбоос нь эвдэрнэ
  - АТ-програм үүснэ
- Цусны бүлгийн тооцоо
  - A+0 = A
  - T+0 = T
  - A + T = AT

| | 0 | A | T | R |
|---|---|---|---|---|
| 0 | 0 | A | T | R |
| A | | A | AT | |
| T | | | T | |
| R | | | | R |

Siedersleben's AT хууль:
А ба Т-г холиход үргэлж дахин хэрэглэхэд хэцүү програм хангамж болдог.

## Цусны бүлгийн зорилго

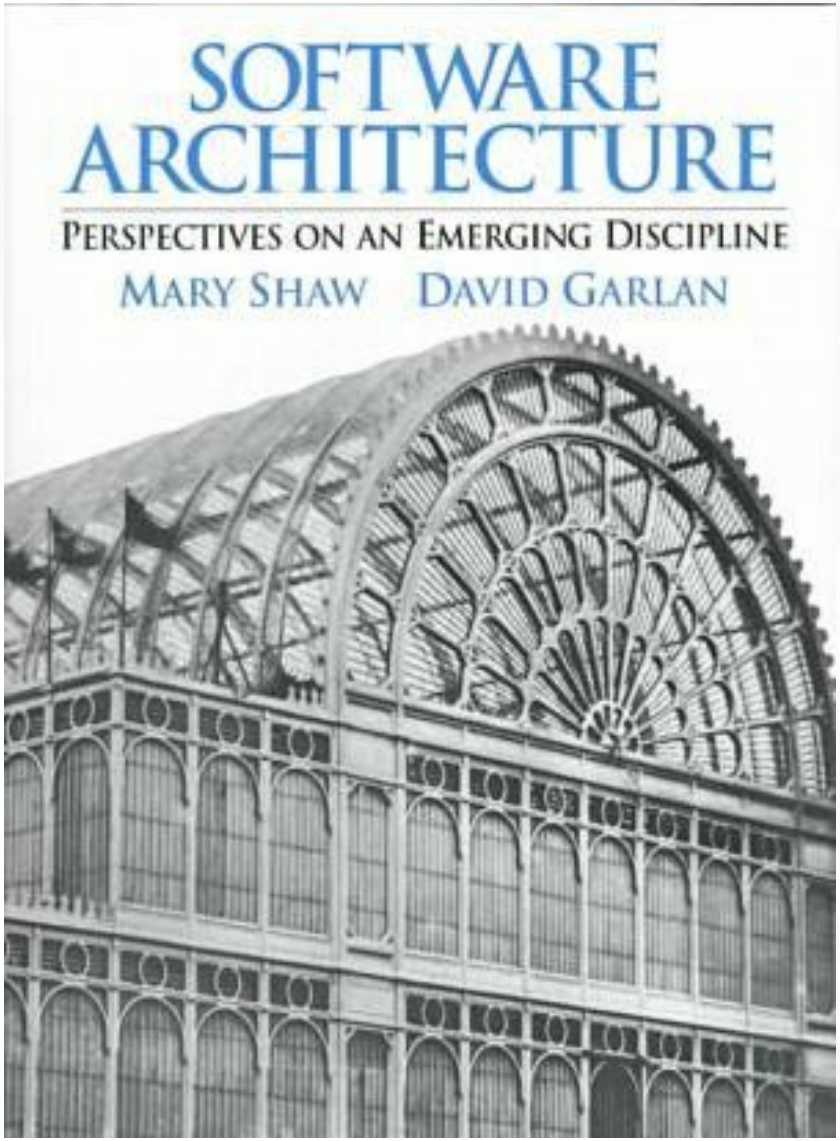
- Архитектурын ба BCED зүгээс нь ПХ-н элементүүдийг цусны бүлгүүдэд хувааж болно (A, T, 0, R, AT)
- Тодорхой бүлгүүдийг холихоос аль болох зайлсхийх (A ба T). Уяир нь

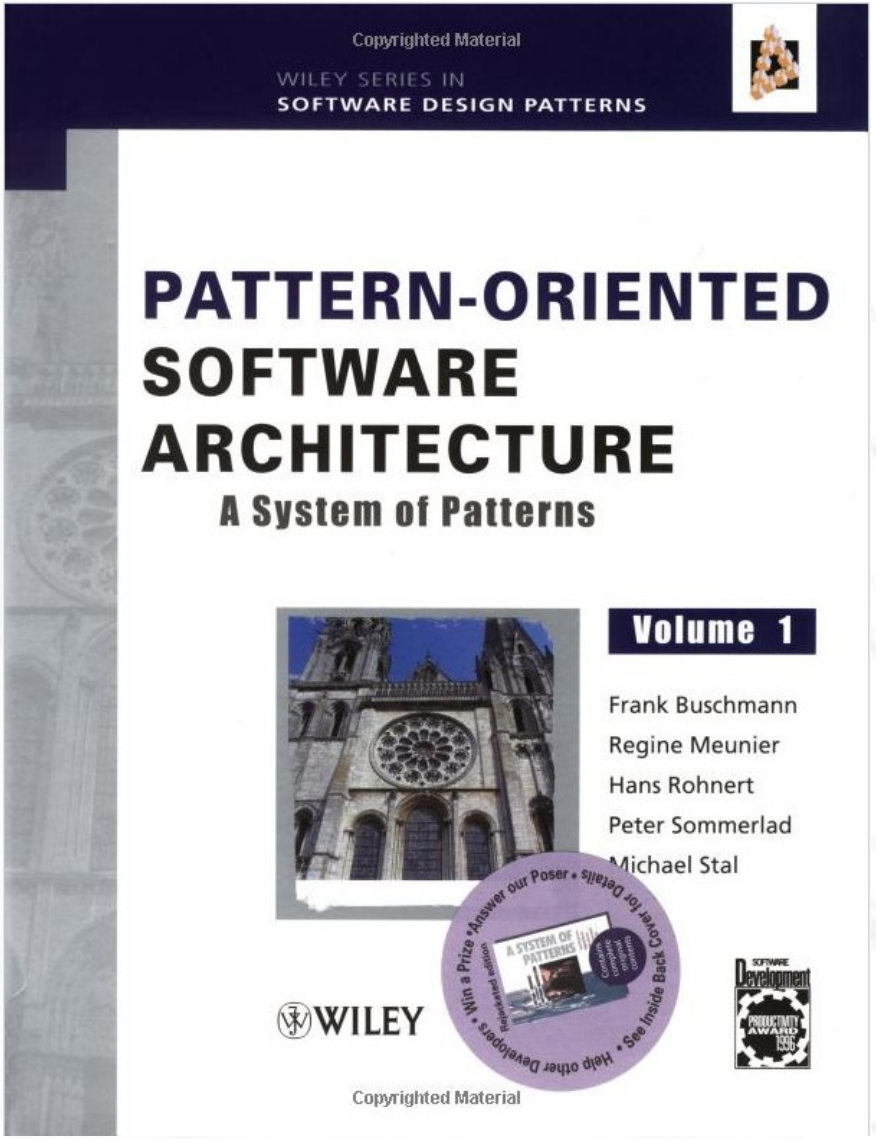
  AT-програмыг дахин хэрэглэх хэцүү
- Бүх компонент пакетыг BCED-архитектурт цусны бүлгээр нь эрэмбэлж холихоос зайлхийх

## Архитектурын хэлбэрийн хэрэглээ

- Сайн архитектур байлаа гээд тавигдсан бүх шаардлагыг хангаж тухайн системийг хэрэгжүүлж чадна гэсэн баталгаа болохгүй.
- Муу архитектур нь тавигдсан шаардлагыг биелүүлэх боломжгүй болгоно.
- Архитектурыг шинжилж тухайн архитектурын боломжит сул талыг илрүүлж, зайлсхийх боломжийг олгоно.

# Судлах материал



# Дүгнэлт

- Архитектурын хэлбэрүүд
  - Pipes & Filter
  - Төвшинт загвар
  - Peer to peer
  - OX зохион байгуулалт
  - Repository/Blackboard
  - SOA
  - Quasar