

Enterprise Application Integration (EAI) - ඉතා සරල සිංහල පැහැදිලි කිරීම

මූලික problem එක තේරුම් ගන්න

කතාව මෙහෙමයි:

ඔබ ගම්මානයක සර්පච්ඡේ ප්‍රධානියා කෙනෙක්. ගමේ:

- **Bank එකක්** - සිංහලෙන් විතරක් වැඩ කරනවා
- **Hospital එකක්** - ඉංග්‍රීසියෙන් විතරක් වැඩ කරනවා
- **School එකක්** - තමිල්තේ විතරක් වැඩ කරනවා
- **Shop එකක්** - ජර්මන් භාෂාවෙන් විතරක් වැඩ කරනවා

දැන් ගම්වැසියෙක්ට Bank එකෙන් loan එකක් ගෙන Hospital එකේ treatment එකක් ගන්න ඕන. **ඒත් මේ organizations දෙක කතා කරන්න බැහැ!**

මේක නමයි Software applications වල නියෙන problem එක!

Service Oriented Architecture (SOA) = Solution

SOA කියන්නේ: Universal translator service එකක් හදන එක.

Real Life වල කිව්වොත්: ගමට translator කෙනෙක් ගෙන්නලා, ඔහු හැම භාෂාවක්ම දන්නවා. දැන් Bank, Hospital, School, Shop හැමෝම ඔහු හරහා කතා කරන්න පුළුවන්!

SOA Evolution - History එක (අතීතය)

කතාව මෙහෙමයි:

1960s - Main Frames යුගය

උදාහරණය: පරණ කාලේ letter යවන එක වගේ

- Data cassette tape එකේ record කරනවා
- Physical ගැන යනවා destination එකට
- **ගොඩක් slow!** Letters වගේ days ගණනක් ගන්නවා

1980s - Socket Communication

උදාහරණය: Telephone කතා කරන එක වගේ

- Direct connection හදනවා දෙන්න අතර
- **Problem:** Connection කඩවුණාම කතාව end!

1990s - File Transfer (FTP/NFS)

උදාහරණය: Courier service වගේ

- Files copy කරන්න වෙනම system
- ටිකක් වේගවත්, ඒත් තාමත් complicated

2000s - RPC (Remote Procedure Calls)

උදාහරණය: Intercom system වගේ

- දුරින් functions call කරන්න පුළුවන්
- **ඒත් problems ගොඩක්:**
 - Java program එකට .NET program එකක් call කරන්න අමාරු
 - Connection details ගොඩක් දැන ගන්න ඕන

දැන් - SOAP/XML යුගය

උදාහරණය: WhatsApp වගේ modern messaging

- Universal format එකක් (XML)
- Any language, any platform work කරනවා
- **මේක තමයි SOA!**

Problems කිසිමට ඇති වුණේ? (හරිම වැදගත්!)

RPC Problems - Real Life වලින්:

Problem 1: Tight Coupling උදාහරණය: ඔබේ මිතුරන්ගේ phone numbers හැම එකක්ම මතක තියාගන්න ඕන වගේ.

- Bank system එකට Hospital system එකේ exact IP address, port, method names හැමදේම දැන ගන්න ඕන
- Hospital system එකේ කුඩා වෙනස්කමක් වුණත් Bank system එක fail වෙනවා!

Problem 2: Interoperability Issues

උදාහරණය: Sinhala කතා කරන කෙනෙකුට Chinese කතා කරන කෙනෙකු සමඟ කතා කරන්න try කරන එක වගේ.

- Java program: "String name = 'Kamal'"
- .NET program: "string name = 'Kamal'"
- **Same data, different formats!** අර්ථය same ඒත් format different

Problem 3: Platform Dependency **උදාහරණය:** Samsung phone charger Sony phone එකට connect කරන්න බැරි වගේ.

- Windows system වල build කරපු software Linux වල run වෙන්නේ නැහැ
- Each platform එකට separate versions හදන්න ඕන

Problem 4: Commercial Lock-in **උදාහරණය:** Dialog SIM ආවිච්චි කරන phone එකට Mobitel SIM දාන්න බැරි වගේ.

- Microsoft technologies ආවිච්චි කරනවා නම් Microsoft products විතරක් use කරන්න ඕන
- Very expensive!

SOA වාසි (ගොඩක් හොඳයි!)

1. Platform Independent

උදාහරණය: Universal phone charger වගේ - any phone එකටම fit වෙනවා

- Java, .NET, PHP, Python කුමක් වුණත් work කරනවා
- Operating system එක වෙනස් වුණත් problem නැහැ

2. Open Source

උදාහරණය: Public library වගේ - free විතරක් නෙමෙයි, modify කරන්නත් පුළුවන්

- Commercial licenses ක් නැහැ
- Community support

3. XML Based Communication

උදාහරණය: English වගේ international language එකක්

xml

<Customer>

<Name>කමල් පෙරේරා</Name>

<Age>25</Age>

</Customer>

- Any programming language එකට understand කරන්න පුළුවන්

4. More than just Communication

SOA කියන්නේ just technical solution එකක් නෙමෙයි:

- **Governance:** Rules and regulations
- **SLA:** Service level agreements (මොනතරම් reliable ද)
- **Security:** Authentication, authorization
- **Monitoring:** Performance tracking

සරල දෘෂ්ටිකෝණය: SOA කියන්නේ හරියට modern postal system එකක් වගේ:


- Standard envelope format (XML)
- Universal addressing system
- Reliable delivery guarantees
- Security (registered post)
- Tracking facilities

SOAP Messages - ගොඩක් වැදගත්!

SOAP කියන්නේ: Simple Object Access Protocol

හරිම Easy Explanation:

SOAP Message = Registered Letter

 SOAP ENVELOPE (ලිපිනයන් සමඟ envelope)

|

|—  SOAP HEADER (Special instructions)

- "Urgent"
- "Confidential"
- "Return receipt requested"
- Security details

└─ 📄 SOAP BODY (Actual letter content)

- Real message/data
- Request details
- Parameters

Real Example:

Bank account balance check කිරීම SOAP message:

xml

```
<soap:Envelope> <!-- ලිපිනය -->

  <soap:Header> <!-- විශේෂ උපදෙස් -->

    <Security>Username: kamal, Password: ****</Security>

    <Priority>High</Priority>

  </soap:Header>

  <soap:Body> <!-- ප්‍රධාන message -->

    <GetBalance>

      <AccountNumber>123456789</AccountNumber>

    </GetBalance>

  </soap:Body>

</soap:Envelope>
```

SOAP වාසි:

1. **HTTP Protocol use කරනවා** - port 80 (normal web traffic)
2. **Firewalls වලින් block වෙන්නේ නැහැ** - normal web page වලේ travel කරනවා
3. **XML format** - any language එකට read කරන්න පුළුවන්
4. **Standardized** - හැමෝම same format use කරනවා

සරල කිව්වොත්: SOAP කියන්නේ applications අතර standard letter format එකක්!

Point-to-Point Integration Problem (ගොඩක් වැදගත් තේරුම් ගන්න!)

Real Life Scenario:


කල්පනා කරන්න: ගම්මානයක applications 10ක් තියෙනවා:


1. Bank System
2. Hospital System
3. School System
4. Police Station System
5. Post Office System
6. Electricity Board System
7. Water Board System
8. Municipal Council System
9. Telecom System
10. Insurance System


Point-to-Point කියන්නේ:

හැම system එකම අනිත් හැම system එකත් සමඟ direct connect වෙනවා!

 Bank ↔ Hospital

 Bank ↔ School

 Bank ↔ Police

 Bank ↔ Post Office

... (and so on for EVERY system!)

Formula:

$$\text{Connections} = N(N-1)/2$$

- N = Number of systems
- 10 systems = $10(10-1)/2 = 45$ connections!
- 20 systems = $20(19)/2 = 190$ connections!

Problems:

1. Maintenance Nightmare **ප්‍රදාහරණය:** Village එකේ හැම ගෙයකම හැම ගෙයකටම private road එකක් හදන්න ඕන වගේ

- One system change වුණාම අනිත් 9 systems වලම configurations change කරන්න ඕන
- Bug fix කරන්න ගොඩක් වෙලාවක් ගන්නවා

2. Security Issues

ප්‍රදාහරණය: ගෙ door 9ක් තියෙනවා, හැම door එකටම වෙනම key එකක්

- 45 different connections secure කරන්න ඕන
- Each connection එකට separate security protocols

3. Network Traffic **ප්‍රදාහරණය:** Village roads වල traffic jam

- Too many direct connections නිසා network slow වෙනවා
- Bandwidth waste

4. Scalability Problem **ප්‍රදාහරණය:** New house එකක් village එකට add වුණාම

- New system එකක් add කරන්න ඕන නම්
- Already existing 10 systems එක්කම connections හදන්න ඕන
- 10 new connections = massive work!

5. Code Duplication **ප්‍රදාහරණය:** හැම ගෙයකම separate water tank, separate generator

- Same functionality different systems වල duplicate වෙනවා
- Database connections, security codes, error handling - හැම system එකේම repeat

Visual Representation:

Bank ↔ Hospital ↔ School

↕ ↕ ↕

Insurance ↔ ESB ↔ Police

↕ ↕ ↕

Telecom ↔ Water ↔ Electric

මේක Point-to-Point = SPAGHETTI! 🍝

ප්‍රශ්නය: Network engineer කෙනෙකුට මේ setup එක maintain කරන්න හරිම අමාරුයි!

Real World Business Impact:

Company එකක scenario:

- ERP System (SAP)
- CRM System (Salesforce)
- Email System (Outlook)
- Payroll System
- Inventory System
- Accounting System

Point-to-Point නම්:

- Each system integration = 3-6 months development
- High maintenance cost
- Each change = ripple effect through all systems
- Developer team stress!


මේ problems solve කරන්නේ ESB (Enterprise Service Bus) එකෙන්!

ESB (Enterprise Service Bus) - මහා Solution! 🚚


Real Life චලිත Perfect Example:


ESB = Pettah Bus Terminal

 Gampaha —┐

 Kandy ———┘

 Galle ———┘  BUS TERMINAL —→  Any Destination

 Jaffna ———┘ (ESB)

 Badulla —┐

 Ratnapura ┘

Before ESB (Point-to-Point):

Gampaha ↔ Kandy (Direct bus)

Gampaha ↔ Galle (Direct bus)

Gampaha ↔ Jaffna (Direct bus)

... 15 different routes = 15 different buses! 🤖

After ESB:

Everyone → Bus Terminal → Any Destination

Only 1 central terminal! 😊

ESB හි Functions:

1. Message Routing (බේ routing වගේ)

Request: "මට Hospital system එකට message එකක් යවන්න ඕන"

ESB: "OK, Hospital system එක IP 192.168.1.100 Port 8080 වල run වෙනවා, යවනවා"

2. Protocol Translation (භාෂාව translate කරන එක)

Bank System: HTTP request යවනවා

Hospital System: JMS message expect කරනවා

ESB: HTTP → JMS convert කරල forward කරනවා

3. Message Transformation (format change)

Input: {"name": "Kamal", "age": 25} (JSON)

Output: <name>Kamal</name><age>25</age> (XML)

4. Security (බිස් ticket checking වගේ)

ESB: "ඔයා authorize ද? Token එක valid ද?"

Valid නම් විතරක් message forward කරනවා

5. Load Balancing

Hospital System 3ක් running:

- Server 1: 70% load

- Server 2: 30% load

- Server 3: 20% load

ESB: Server 3 එකට message යවනවා (least loaded)

ESB Benefits - ගොඩක් වාසි!

1. Centralized Management උදාහරණය: Bus terminal එකේ announcement system

- One place එකෙන් හැම route එකම control කරන්න පුළුවන්
- New route add කරන්න easy
- Route cancel කරන්න easy

2. Reduced Connections

Before: 10 systems = 45 connections

After: 10 systems = 10 connections (each to ESB)

95% reduction! 🎉

3. Easy Maintenance උදාහරණය: Hospital system IP address change වුණාම

- Point-to-Point: 9 systems වල configurations change කරන්න ඕන
- ESB: ESB configuration එක විතරක් change කරන්න ඕන

4. Reliability

ESB Features:

- Message queuing (message lost වෙන්නේ නැහැ)

- Retry mechanism (fail වුණාම again try කරනවා)

- Error handling (problems log කරනවා)

- Health monitoring (systems monitor කරනවා)

5. Scalability උදාහරණය: Bus terminal එකට new route add කරන එක වගේ

- New system add කරන්න ඕන නම් ESB එකට connect කරන්න විතරක්
- අනිත් systems වලට කිසිම කරන්න දේවල් නැහැ

ESB කොහොමද වැඩ කරන්නේ?

Step by Step Process:

1. 🏦 Bank System: "මට customer details Hospital එකෙන් ගන්න ඕන"

↓

2. 🚚 ESB: "Message receive කරනවා, security check කරනවා"

↓

3. 🚚 ESB: "Hospital system JSON expect කරනවා, XML to JSON convert කරනවා"

↓

4. 🚚 ESB: "Hospital system load check කරනවා, available server එකට යවනවා"

↓

5. 🏥 Hospital System: "Data process කරල response යවනවා"

↓

6. 🚚 ESB: "Response receive කරල Bank system එකට return කරනවා"

↓

7. 🏦 Bank System: "Data receive කරනවා, customer screen එකේ show කරනවා"

****හරිම simple right? ESB කියන්නේ intelligent middleman කෙනෙක්! Protocol**

Translation: HTTP message එක JMS message එකක් බවට convert කරනවා 3. **Security:**

Authentication, encryption 4. **Reliability:** Message deliver වුණාද check කරනවා

WSO2 ESB - Real Implementation

WSO2 ESB කියන්නේ මොකද්ද?

WSO2 = Sri Lankan Company! LK

- Colombo වල හදුන්වන world-class ESB software
- Open source (free!)
- Enterprise grade (big companies use කරනවා)

Technology Stack - ගොඩනැගිල්ල:

1. Carbon Platform (Foundation) උදාහරණය: ගෙ foundation වගේ

CARBON PLATFORM

- └— Java based
- └— OSGi framework (plug and play components)
- └— Start/Stop components without restart
- └— Base for all WSO2 products

Real benefit: New feature add කරන්න ඕන නම් server restart කරන්න එපා!


2. Apache Synapse (Mediation Engine) උදාහරණය: Bus terminal එකේ traffic controller

APACHE SYNAPSE

- └— Message processing engine
- └— Protocol conversions (HTTP ↔ JMS ↔ FTP)
- └— High performance (non-blocking)
- └— Supports XML, JSON, Binary, Plain text

Real Business Scenario - ගොඩක් practical!

Scenario: Social Media Integration

 REQUEST: "Dropbox photos ගෙන Facebook එකට upload කරල Twitter එකෙන් friends ලට message යවන්න"

ESB Process Flow:

1. 📱 Mobile App → ESB Proxy Service

"Photos upload කරන්න ඕන"

2. 📁 ESB → Dropbox API

"Photos download කරනවා"

3. 🖼️ ESB → Image Processing

"Photos resize කරනවා (Facebook requirements අනුව)"

4. 📘 ESB → Facebook API

"Processed photos upload කරනවා"

5. 🐦 ESB → Twitter API

"Friends ලට notification යවනවා"

6. 📧 ESB → Email Service

"User ට success email යවනවා"

7. 📱 ESB → Mobile App

"සාර්ථකයි! Success response"

Traditional Point-to-Point නම්:

- Mobile app එකේ 4 different API integrations code කරන්න ඕන වෙයි
- Error handling 4 places වල implement කරන්න ඕන
- Security 4 places වල handle කරන්න ඕන

- **ESB නම්:** Mobile app එක ESB එකට විතරක් connect, අනිත් හැමදේම ESB handle කරනවා!

Apache Synapse Features:

1. High Performance

 Non-blocking HTTP/HTTPS transports

- Traditional: 1 request = 1 thread (expensive)
- Synapse: 1000s requests = few threads (efficient)

2. Protocol Support

 Supported Protocols:

- HTTP/HTTPS (web traffic)
- JMS (Java messaging)
- FTP/SFTP (file transfer)
- Email (POP3, IMAP, SMTP)
- SMS (mobile messages)
- TCP/UDP (network protocols)
- VFS (file system)
- XMPP (chat protocols)

Real Example:

Email → ESB → SMS conversion:

Customer ගේ email complaint → ESB receive → Mobile SMS ට convert → Manager ගේ phone එකට යවනවා

3. Message Format Support

 Supported Formats:


- XML (structured data)
- JSON (web APIs)
- Plain text (simple messages)

- Binary (files, images)
- SOAP (web services)

Message Flow Process:


MESSAGE FLOW

|

|—  Request comes to ESB


- Security check (authentication)
- Message validation

|

|—  Main Sequence Processing


- Message transformation
- Routing decisions
- Backend service calls

|

|—  Success Path

- Response processing
- Send back to client

|


|—  Fault Sequence (if error)

- Error logging
- Error response
- Retry mechanisms

Real Example - Banking:

 Mobile Banking App Request:


"Account balance check කරන්න මින"

 ESB Security Check:


"User authenticate කරනවා, permissions check කරනවා"

 ESB → Core Banking System:

"XML SOAP message යවනවා"

 Core Banking Response:

"Balance: Rs. 50,000" (XML format)

 ESB → Mobile App:

"{"balance": 50000}" (JSON format convert කරල)

Message Mediation

Mediator කියන්නේ: Message එකක් ගෙන modify කරල forward කරන component එකක්

Real World Example: Hotel receptionist කෙනෙක් වගේ:

1. Guest request එකක් ගන්නවා
2. Check කරනවා room available ද කියලා
3. Modify කරනවා (room number add කරනවා)
4. Housekeeping department එකට forward කරනවා

Enterprise Integration Patterns

1. Message Channel

උදාහරණය: WhatsApp chat එකක් වගේ - sender සිට receiver ට message යන path

2. Pipes and Filters

උදාහරණය: Water purification system:

- Pipe 1: Raw water
- Filter 1: Remove sediments
- Pipe 2: Semi-clean water
- Filter 2: Remove chemicals
- Pipe 3: Clean water

3. Message Router

උදාහරණය: Post office sorting center:

- Letter එනවා
- ZIP code බලනවා
- Correct post office එකට යවනවා
- Message content change කරන්නේ නැහැ

4. Message Translator

උදාහරණය: Language translator:

- Sinhala message එකක් English message එකක් බවට convert කරනවා
- XSLT mediator use කරනවා

5. Message Endpoint

උදාහරණය: Airport gate:

- Passengers message channel එකෙන් එනවා
- Gate එකෙන් airplane එකට යනවා

Proxy Services - ESB ගේ Main Components

Proxy Service කියන්නේ ESB එකේ entry point. Real life වලින් කිව්වොත් Hotel Reception වගේ!

1. Pass Through Proxy

Real Example: Bus Terminal Transfer 🚌

🚌 BUS TRANSFER SCENARIO:

|

Passenger (Gampaha → Kandy):

|— 🚌 Bus 1: Gampaha → Colombo Terminal

|— 🚶 Walk to Platform B (Pass through)

|— 🚌 Bus 2: Colombo Terminal → Kandy

ESB Example:

📱 Mobile App → Pass Through Proxy → Backend Service

|

Proxy functions:

- Simply forwards message
- No processing/modification
- Catch-all for unmatched requests
- Basic routing only

Use Case: Testing phase වල ඔබේ simple forwarding වලට perfect!

2. Secure Proxy

Real Example: Bank Vault Entry 🏦

🏦 BANK VAULT ACCESS:

|

Customer → Security Guard → ID Check → Fingerprint → Vault Door

|


Security Guard:

|— ID Identity verification

|— 🖐 Biometric check

|— 🗝 Access permissions

|— 📝 Visit logging

└─  Escort to vault

ESB Example:


 SECURE PROXY PROCESS:

|


Client Request → Secure Proxy → WS-Security Processing → Backend Service

|


Security Processing:

└─  Username/Password verification

└─  Token validation

└─  Message encryption/decryption

└─  Security audit logging

└─  Forward to unsecured backend

Real Scenario: Internet banking app → Secure proxy → Core banking (internal, no security)

3. WSDL Based Proxy

Real Example: Restaurant Franchise Menu 


 PIZZA FRANCHISE:


|


Head Office → Standard Menu (WSDL) → All Branches


|

Menu includes:

└─  Available pizzas (Operations)

└─  Prices (Parameters)

└─  Ordering process (Binding)

└─  Branch locations (Endpoints)

ESB Example:





WSDL BASED PROXY:

|

Remote WSDL URL → Download WSDL → Generate Proxy → Expose locally

|

Benefits:

- |—  Automatic service discovery
- |—  WSDL changes auto-sync
- |—  Endpoint extraction
- |—  Contract-first approach

Use Case: Third-party web service එකක් local network එකට expose කරන්න







4. Logging Proxy

Real Example: Hotel Guest Register 

 HOTEL GUEST REGISTER:

|

Every guest entry:

- |—  Check-in time: 2:30 PM
- |—  Guest: Mr. Perera
- |—  Room: 205
- |—  Vehicle: CAR-1234
- |—  Contact: 0771234567
- |—  Expected checkout: Tomorrow 12 PM

ESB Example:

 LOGGING PROXY PROCESS:

|

Request → Log Request Details → Forward to Backend → Log Response → Return to Client

|

Logged Information:

- └─ 🕒 Timestamp
- └─ ID Request ID
- └─ 📁 Service name
- └─ ✉ Request payload
- └─ 📄 Response payload
- └─ ⌚ Processing time
- └─ ❌ Error details (if any)

Real Benefits:

- Debugging වලට ගොඩක් help
- Performance monitoring
- Audit trails for compliance
- Error analysis

5. Transformer Proxy

Real Example: Currency Exchange Counter 💵💴

💵💴 CURRENCY EXCHANGE:

|

Customer brings USD → Exchange Counter → Convert to LKR → Give LKR to customer

|

Exchange Process:

- └─ 💵 Input: \$100 USD
- └─ 📊 Current rate: 1 USD = 320 LKR
- └─ 🧮 Calculation: $100 \times 320 = 32,000$
- └─ 💰 Output: Rs. 32,000 LKR

└─ 📄 Receipt with details

ESB Example:

🔄 TRANSFORMER PROXY:

|

XML Request → XSLT Transformation → Backend Service → XSLT Transform Response → JSON Response

|

Transformation Examples:

└─ 🗂️ XML ↔ JSON conversion

└─ 🏷️ Field name mapping

└─ 📅 Date format changes

└─ 🔄 Data type conversions

└─ 🌐 Namespace modifications

XSLT Example:

xml

<!-- Input XML -->

<customer>

<n>කමල්</n>

<වයස>25</වයස>

</customer>

<!-- XSLT Template -->

<customer_info>

<name>{n}</name>

<age>{වයස}</age>

<country>Sri Lanka</country>

</customer_info>

6. Custom Proxy

Real Example: Wedding Planner 🏠❤️

🏠 WEDDING PLANNER SERVICES:

|

Couple's Requirements → Wedding Planner → Coordinate all vendors

|

Custom Services:

- └─ 🎵 Music (DJ coordination)
- └─ 🍰 Catering (Menu planning)
- └─ 📷 Photography (Album creation)
- └─ 🚗 Transport (Vehicle booking)
- └─ 🌸 Decoration (Theme design)
- └─ 📅 Timeline management

ESB Custom Proxy:

⚙️ CUSTOM PROXY FEATURES:


|


Incoming Request → Custom Processing Logic → Multiple Backend Services

|

Customizations:

- └─ 🔄 Custom routing rules
- └─ 🔄 Custom transformations
- └─ 🛡️ Custom security policies
- └─ 📈 Custom QoS settings

└─  Custom endpoint configurations

└─  Custom error handling

└─  Custom monitoring

Proxy Service Selection Guide:

 WHICH PROXY TO USE?

|

└─ Simple forwarding only? → Pass Through Proxy

└─ Need security? → Secure Proxy

└─ Third-party WSDL integration? → WSDL Based Proxy

└─ Need request/response logging? → Logging Proxy

└─ Need data transformation? → Transformer Proxy


└─ Complex business logic? → Custom Proxy

Real Implementation Example:


E-commerce Integration:


 E-COMMERCE PROXY SETUP:


|

└─  Secure Proxy: Customer authentication

└─  Logging Proxy: Order tracking

└─  Transformer Proxy: Product catalog (XML→JSON)

└─  Custom Proxy: Payment processing (multiple gateways)

└─  Pass Through Proxy: Simple product queries

Benefits:

- Each proxy specialized function එකක් handle කරනවා
- Modular architecture
- Easy maintenance

- Scalable design
- Reusable components

Key Point: Proxy services combine කරල complex business scenarios handle කරන්න පුළුවන්! ටෙන Dollars දෙනවා

ESB Connector Hierarchy

Mediator (Individual worker)

↓

Sequence (Team of workers)

↓

Template (Department)

↓

Connector (Whole Company)

Service Bus vs Mediation

Service Bus: Common communication highway - A1 highway වගේ හැමෝම use කරන්න පුළුවන්

Mediation: Traffic police කෙනෙක් වගේ - conflicts handle කරනවා, directions දෙනවා

ප්‍රායෝගික Business Scenario

Example: Dropbox photos ගෙන Facebook එකට upload කරලා Twitter එකෙන් friends ලට message එකක් යවන්න:

1. ESB Proxy Service එකට request යවනවා
2. Dropbox API එකෙන් photos download කරනවා
3. Facebook API use කරලා photos upload කරනවා
4. Twitter API use කරලා message යවනවා
5. User ට notification යවනවා

හරිම වාසි:

- Applications අතර සම්බන්ධතාවය

- Centralized management
- Security හා reliability
- Different protocols support
- Easy maintenance

මෙම concepts හොඳින් understand කරගන්නාම modern enterprise applications develop කරන්න ගොඩක් easy වෙයි!