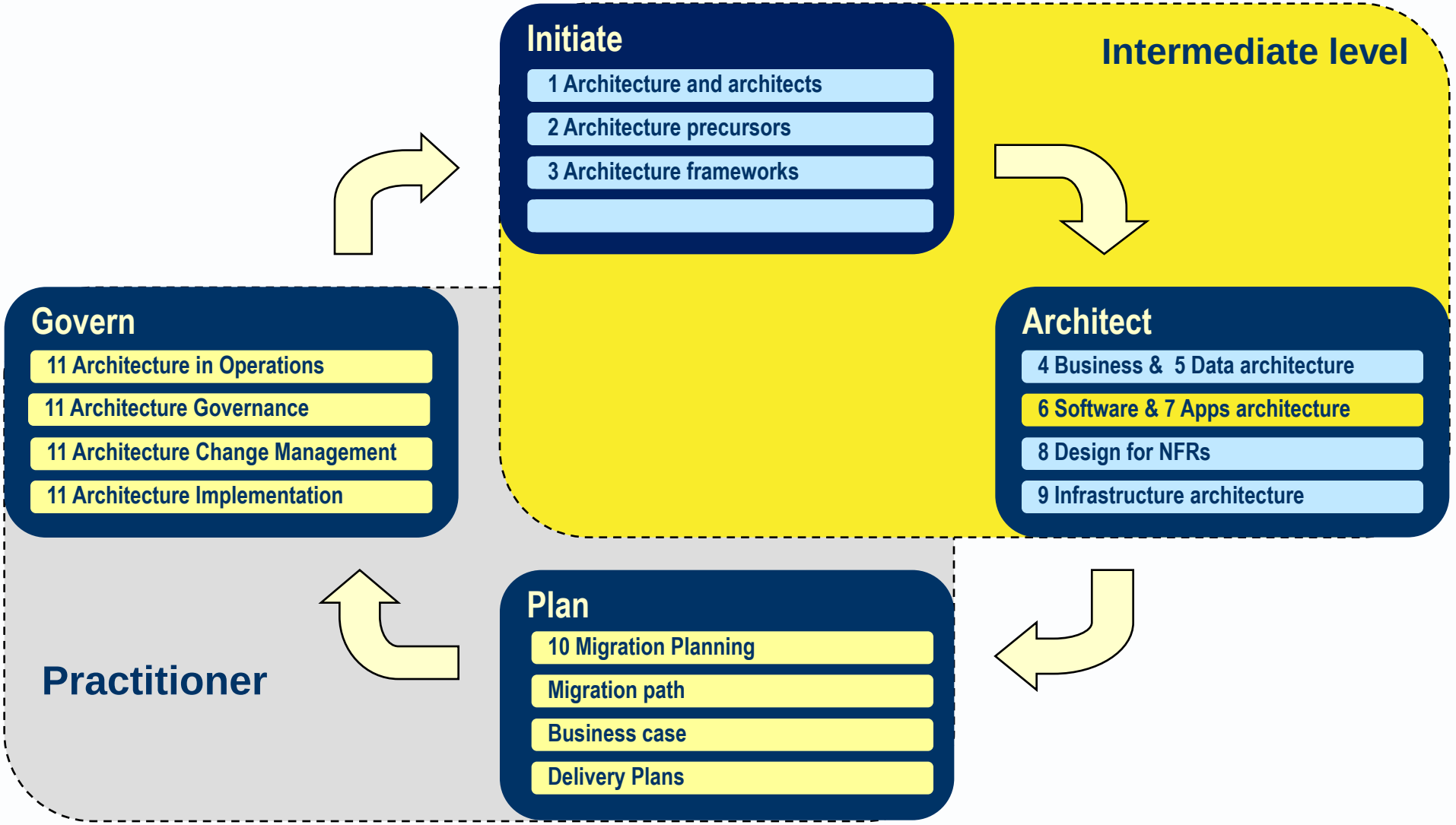
Avancier Reference Model

Applications Architecture (ESA 7)

7. Applications architecture

Avancier



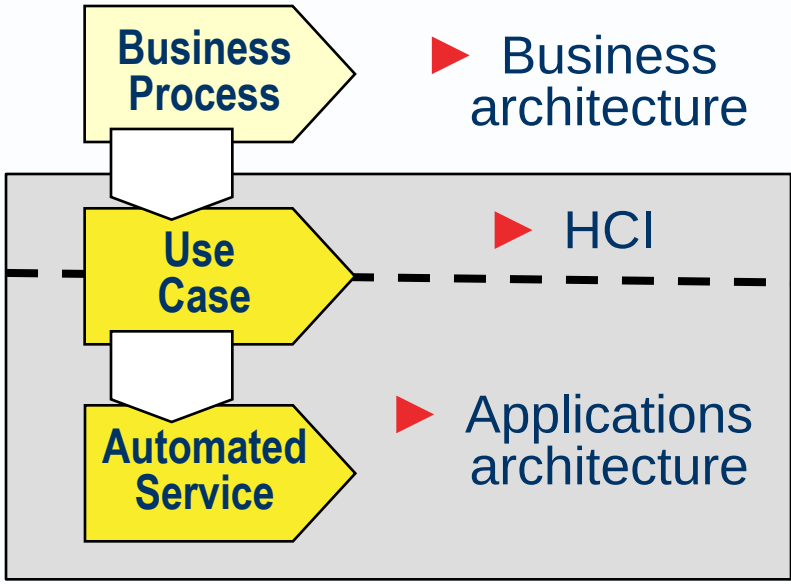
7.1 Foundation

Required behaviours	Logical structures	Physical structures
IS/App service	Application interface	Application

Application service

- [A service] or operation that an application component is required to perform
- E.g. log in, change password, submit insurance claim.
- It supports and enables a business by capturing or retrieving business data.
- It may be bundled with related application services into a portfolio or API.

- It may be a use case at the human-computer interface,
- or a wholly automated behavior.



Application interface definition

- [An interface definition] that includes services accessible by application clients.
- It identifies services and hides what performs them.
- It may be defined in a user interface or API.
- It may be implemented as a facade component in its own right.

Application component

- [A component] capable of performing automated behaviors.
- It can be a whole application or a component within one.
- It may be encapsulated behind an API, and may maintain some business data.

  - **Logical application component**
  - [An application component] specified by services offered and data maintained.
  - The specification is independent of implementation and technology.
  - It is typically mapped to one physical application component.
  - But the relationship may be more complex.
  - **Physical application component**
  - [An application component] that realises a logical application component specification.
  - It uses particular technologies.
  - It typically realises one logical application component.
  - But the relationship may be more complex.

Microservice

- [An application component] encapsulated behind an API, that is
  - large enough to be regarded as an application, but
  - small enough to suit agile development and enhancement.

- It is usually a subdivision of what could be a larger application, and maintains a subset of what could be a larger database.

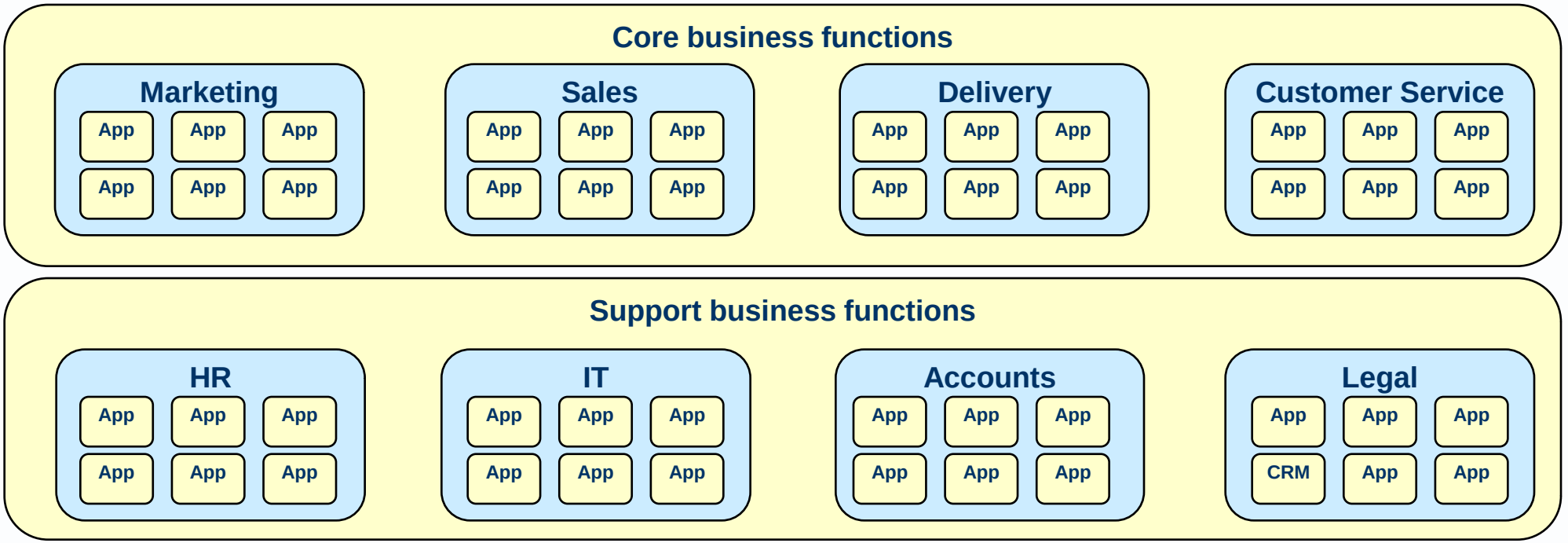
- Data integrity is maintained using messaging passing rather than ACID transaction roll back.
- So, using the BASE principle, designers must analyse the consequences of temporary data integrity and consider compensating transactions to restore it.

7.2: Application portfolio management

- **Application portfolio management** [a work process] to catalogue, classify, describe, and value the applications of an enterprise, with a view to rationalisation or optimisation of those applications.

Application portfolio catalogue

- [an artefact] listing business applications and recording their properties.
- Usually structured so as to reflect the business function hierarchy.



Some ways to classify applications

- **Classification by architecture domain**
  - Business application
  - Generic application
  - Platform application
- **Classification by business function**
  - Enterprise Resource Planning (ERP)
  - Customer relationship management (CRM)
  - Other common functions
- **Classification by value**
  - [A pattern] that usually measures both business value and technical value and may be presented in a two by two grid.

Classification by architecture domain

- **Business application** [an application] that captures or provides data to support a business role or process.
- E.g. accounting; billing, customer relationship management, enterprise resource planning, business intelligence, patient administration. It has breadth in terms of use cases supported and depth in terms of software layers.

- **Generic application** [an application] that offers universal use cases. E.g. calculator, drawing tool groupware, media player, spreadsheet, browser, word processor.

- **Platform application**: [an application] or system software that runs computer hardware or serves other applications.

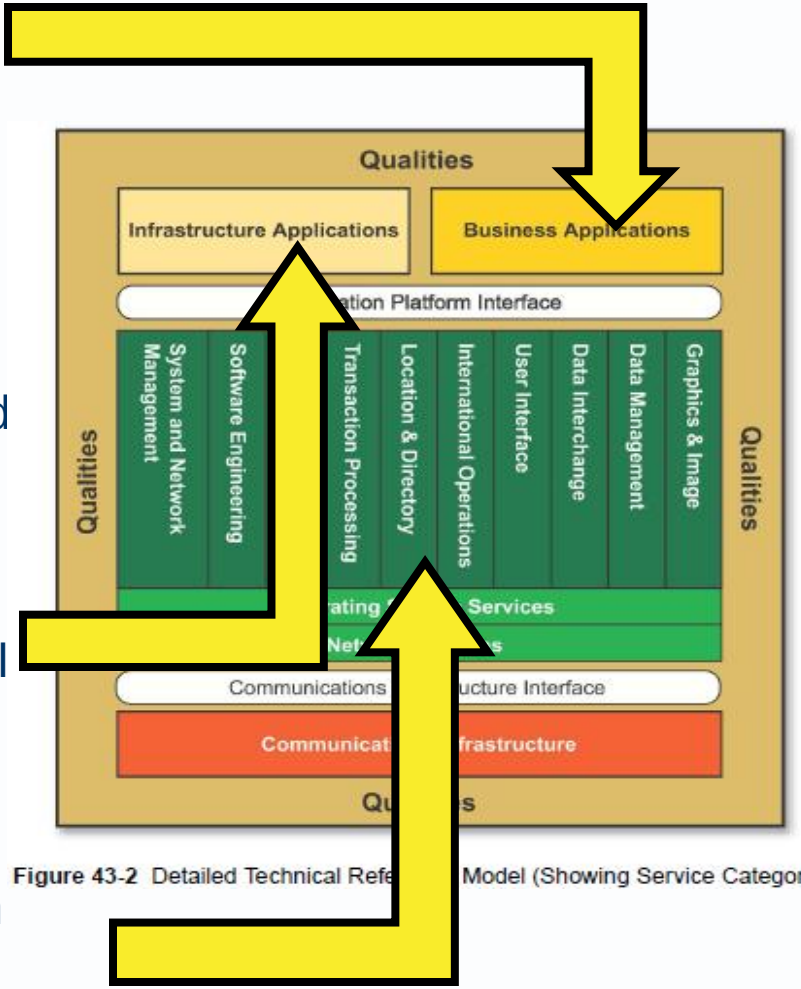


Figure 43-2 Detailed Technical Refe Model (Showing Service Categories)

Classification by business function

Avancier

- [a pattern] dividing applications by business function.

- **Enterprise Resource Planning (ERP)** [a business application] that supports the planning of how enterprise resources (materials, employees, customers, etc.) are acquired, moved from one state to another.
- It maintains data needed for some or all of Manufacturing, Supply Chain Management, Financials, Projects, Human Resources, Data Warehouse and Management Information. It can include CRM and Billing.

- http://www.evaluationcentre.com/erp_software/home.go
- 23% deploy ERP as their main application strategy
- 13% deploy ERP with other standalone packages
- 13%, deploy ERP in combination with bespoke solutions
- 23% based core systems on standalone best of breed packages
- 17% on bespoke solutions

Survey population
manufacturing sector (23%)
public sector (17%),
retail (10%)
distribution & logistics (7%),
IT & telecoms (7%)
financial services (7%).

Customer relationship management (CRM)

- [a business application] that supports the development and maintenance of mutually beneficial long-term relationships with customers.
- It helps with some or all of the following
  - attracting customers,
  - transacting business with customers,
  - servicing and supporting customer,
  - enhancing customer relationships.

A little more about CRM

- The promise of CRM technology lies in improved marketing, customer satisfaction and increased sales productivity:

- A business support system that helps with some or all of >>

- Attracting customers
  - Brand building
  - Customer value management
  - Customer modelling
  - Product development
  - Marketing operations
  - Product customisation
- Transacting business with customer
  - Sales force operations
  - Service centre operations
- Servicing and supporting customer
  - Field service operations
  - Supply chain/logistics
  - Website operations
- Enhancing customer relationships
  - Customer retention
  - Customer knowledge

Other common functions

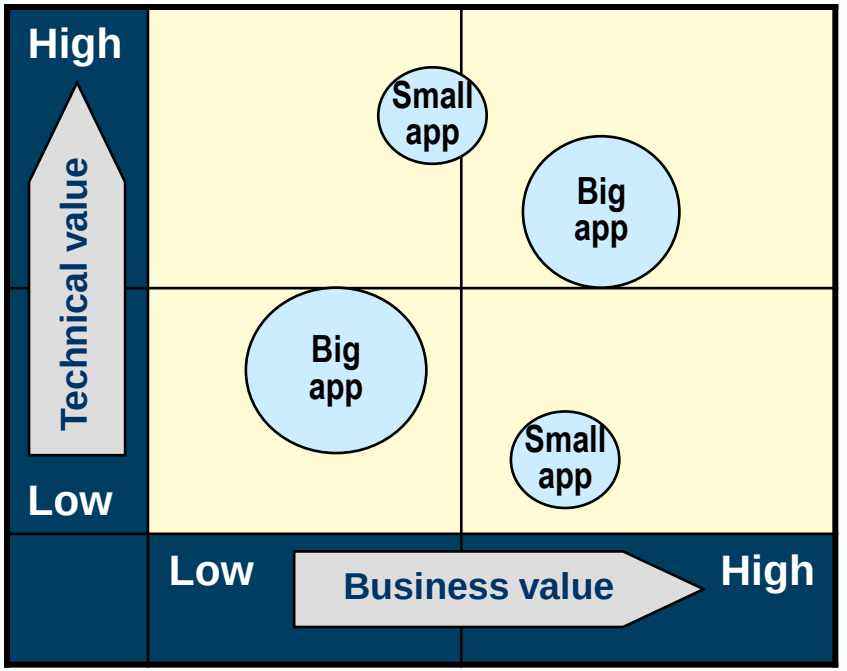
Avancier

- Accounting
- Financial Reporting
- Data Warehousing, Business Intelligence and CPM.
- Document Management, Content Management an BPM.
- HR and Payroll
- Project Management

Classification by value

- [A pattern] that may measure both business value and technical value.
- It may be presented in a two by two grid.

- Business value
  - Fit to business strategy/standard, and road map
  - Cost per user, per transaction
  - Usability, etc.
- Technical value
  - Fit to strategy/standard, and road map
  - Serviceability, maintainability
  - Incident/problem frequency

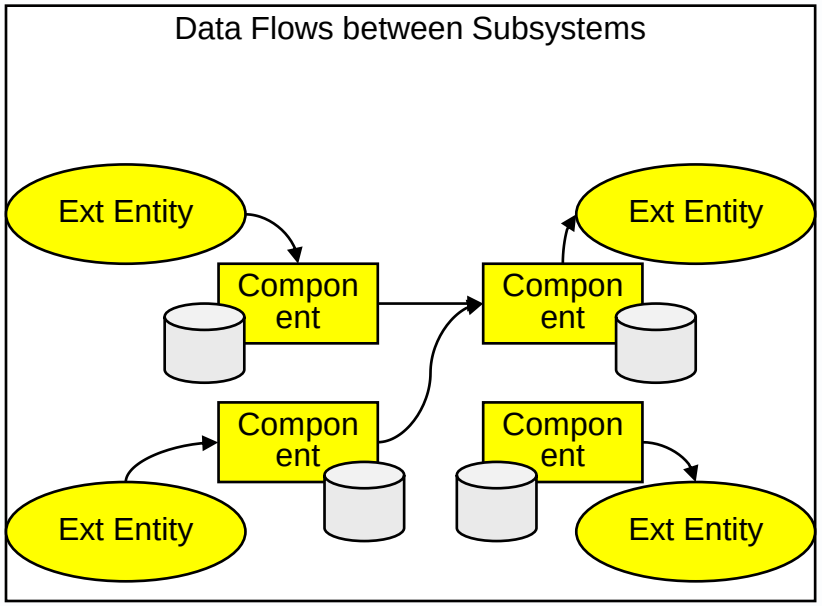


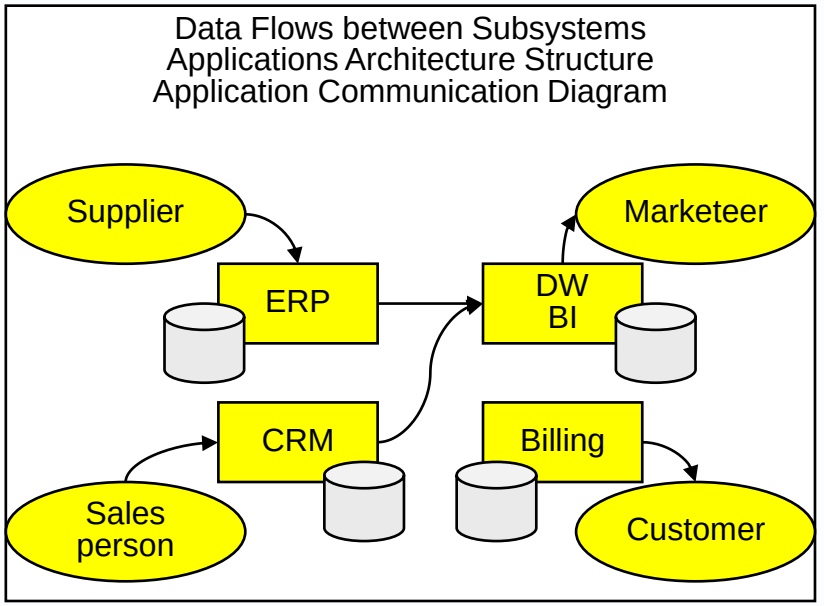
Other ways to classify applications

- Size
- Complexity
- User type: Public / Employee / Technical
- Generality: Universal <> Unique to business
- User base: Single-user - Dept - Enterprise
- Usage style: OLTP / Business Intelligence

Applications communication diagram

- [an artefact] that shows how applications are related by the exchange of data.
- Typically some kind of data flow diagram, or, where there are too many data flows, a dependency diagram.

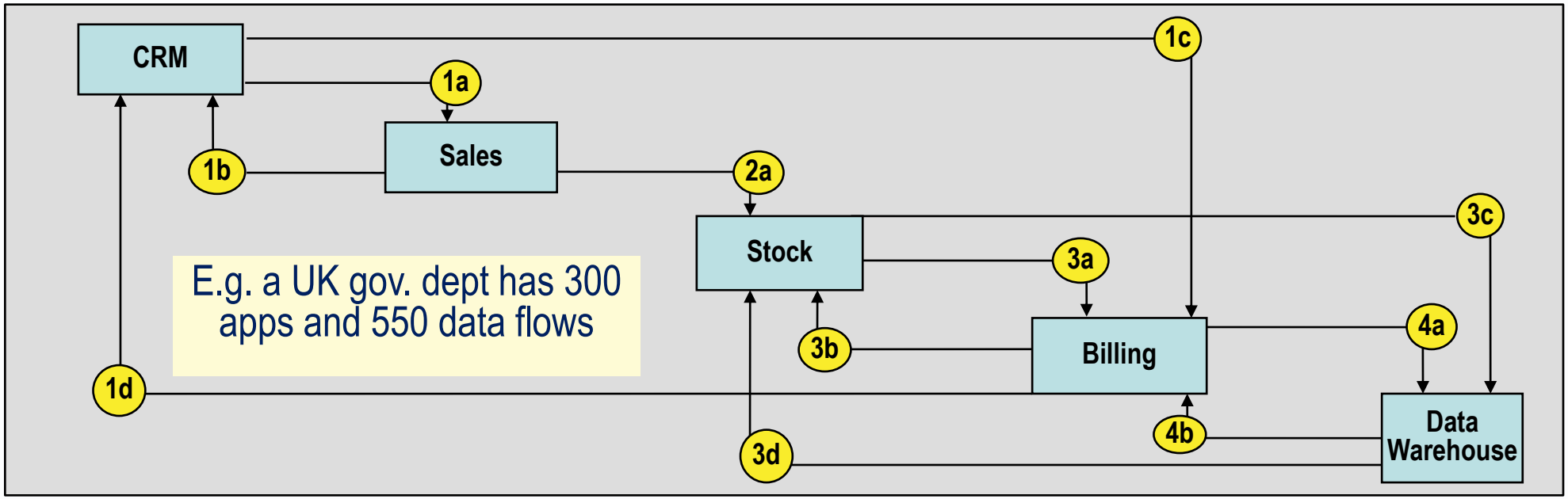




Applications communication diagram + data flow catalogue

Avancier

(cf. N2 model, or Node Connectivity diagram in FEAF)



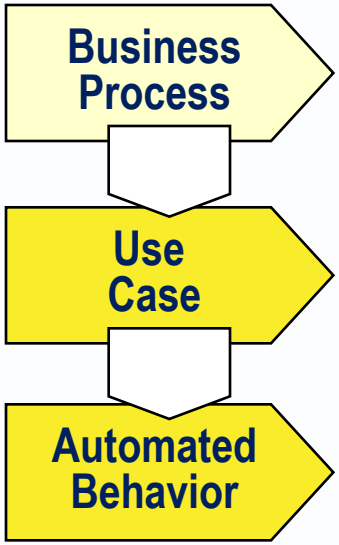
Data Flow id	Source App	Destination App	Data content	Trigger event
1a	CRM	Sales	Sales order request	New sales order
1b	Sales	CRM	Sales order confirmation	Order created in the Sales system
2a	Sales	Stock	Requisition	Subscribe/Publish timer

7.3 Application behaviour

- Use case diagram
- Use case description
- Automated behavior
- Application/data entity matrix
- Application interaction diagram

Business architecture

Applications architecture

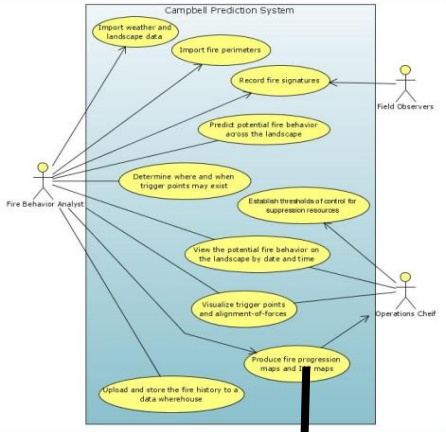


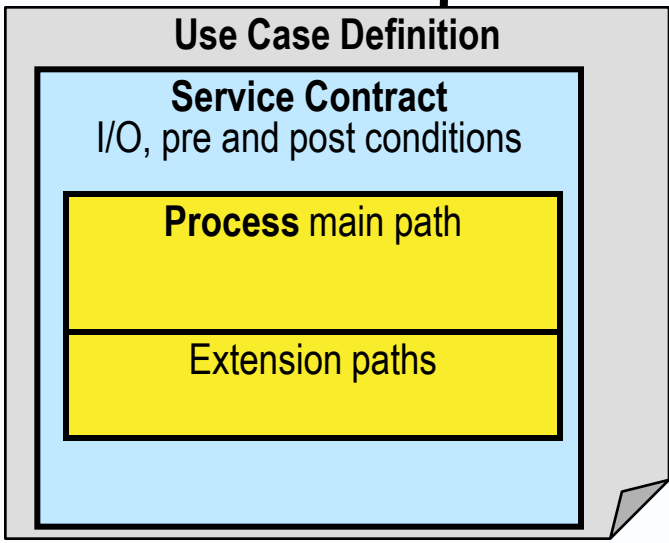
Use case

- **Use case diagram**
- [An artifact] that shows the uses cases supported by an application.
- An application is scoped and logically defined by the use cases it supports.

- **Use case description**
- [An application service] at the human-computer interface.
- A use case description defines a use of a system by an actor.
- It is normally named as a goal in verb-noun form (e.g. assess claim).
- It may be defined declaratively as a service.
- It may be detailed in terms of a process with main and alternative paths.

Use Case Diagram





Use case description – a simple example

Avancier

Service	Name: Book train seat Inputs: Journey facts Outputs or results: Seat reservation
Process flow	1. Identify journey start and end stations 2. Identify outbound start time 3. Identify inbound start time 4. Identify traveller numbers and ages 5. Review booking details 6. Enter payment details 7. Create personal account (optional) 8. Confirm payment details 9. Collect receipt
Non-functionals	Response time Throughput Availability etc

Service "signature"

Each step could be a user story

Don't forget the numbers

Use case – reference to automated service

Goal or name	Capture and confirm order
Scope	Kitchen sales system
Actor	Salesman
Stakeholder	Customer
Trigger event	Customer wants to place order Salesman presses order kitchen command
Precondition	Kitchen plan has been populated with kitchen items Kitchen order app is loaded onto the lap top
Post condition	Order is printed, signed and logged in order app
Main path	1 Salesman presses order kitchen command (OPEN ORDER APP) 2 Kitchen drawing app opens Kitchen Order app in new window 3 Salesman enters customer details 4 Salesman sends order to printer (PRINT ORDER) 5 Salesman presents order to customer 6 Customer signs order 7 Salesman enters confirmation of signature (CONFIRM ORDER) 8 Salesman close order app 9 Kitchen drawing app stores kitchen plan as version n+1
Extension paths	2 Kitchen drawing app displays “Kitchen order app not available” 3 Salesman closes order app before printing the order
NFRs	

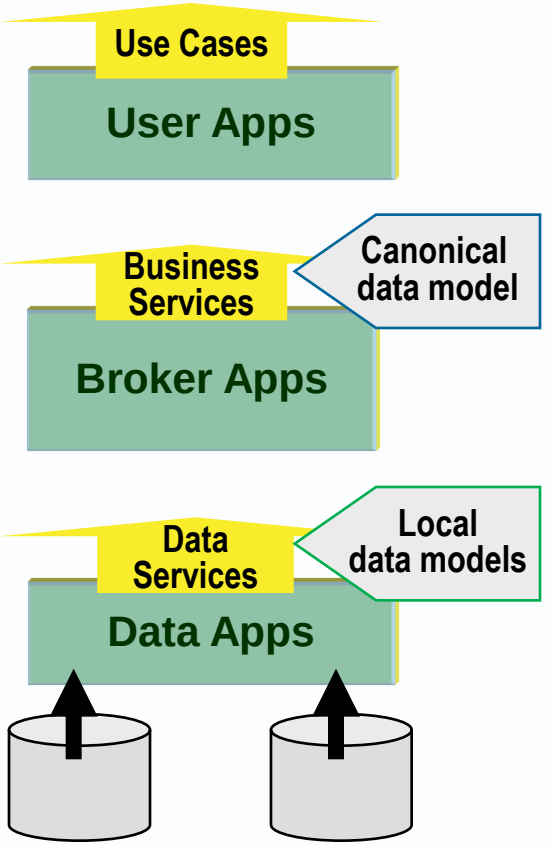
Reference to automated service

Automated behavior

- [An application service] that can be requested of a software component.
- It is sometimes an ACID transaction.

- **Automated business service**
- [An automated behavior] whose input and output data is defined in a canonical data model.

- **Automated data service**
- [An automated behavior] whose input and output data items are defined according to the parochial or physical data model of a specific data source.



Transaction

- [An automated behavior] a unit of work, a buy-sell or client-server interaction between parties, e.g. between a user and a computer, or an application and a database.
- **ACID transaction**
  - [A transaction] that is Atomic, Consistent, Isolated and Durable.
  - It can be rolled back if a specified precondition is violated.
  - Using a transaction manager to automate the roll back of a transaction preserves the integrity of stored data and simplifies design and development.
  - But is often impossible in loosely-coupled and distributed systems.
- **Compensating transaction**
  - [A transaction] to handle the side effects of a process (or workflow) that started but could not complete successfully.
  - It may undo updates committed to databases, remove messages placed in message queues, send follow-up correction messages, or report cases of data disintegrity.

ACID acronym says that database transactions should be:

- **A**tomic
  - Everything in a transaction succeeds or the entire transaction is rolled back.
- **C**onsistent
  - A transaction cannot leave the database in an inconsistent state.
- **I**solated
  - Transactions cannot interfere with each other.
- **D**urable
  - Completed transactions persist, even when servers restart etc.

- Transaction roll back simiplifies processing because means no need for **compensating transactions**

BASE (Basically Available, Soft State)

- The opposite of ACID.
- The principle used where interacting components are distributed and a process cannot be rolled back by a transaction manager.
- The unit of work is workflow started by one component (basically available) and completed by another component.
- If something goes wrong, since an update or output effect cannot be rolled back, compensating transactions may have to be designed.

Application / data entity matrix

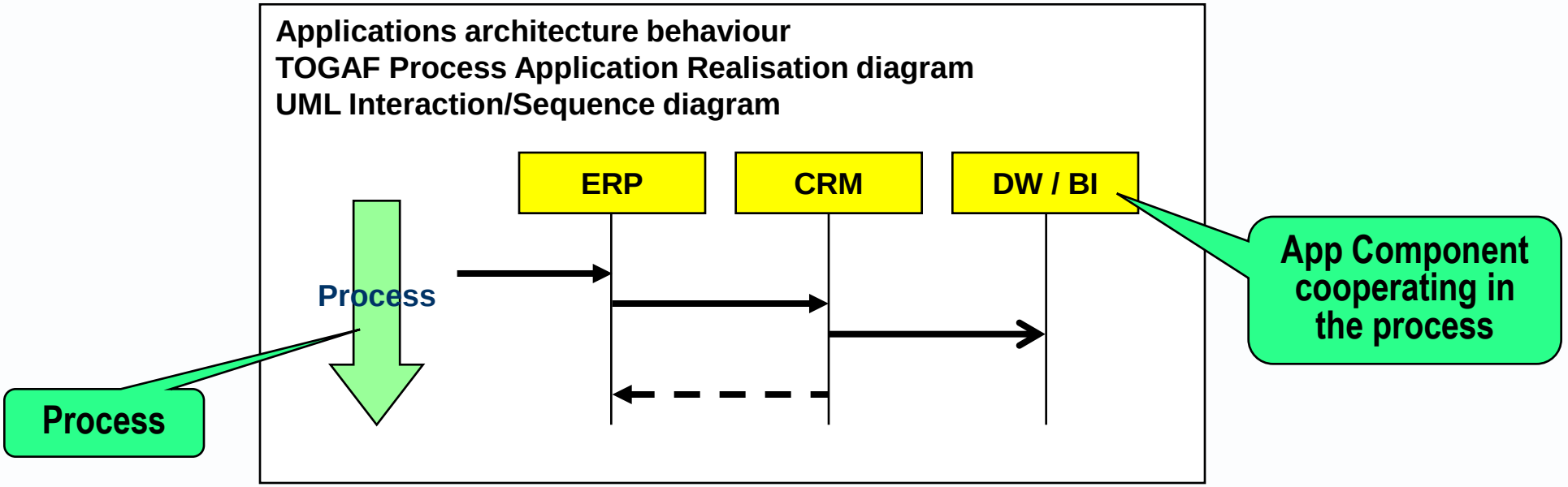
- [an artefact] that shows which applications create and use which data entities
- It may be completed with Create, Read, Update, and Delete entries.

App / Data entity	CRM	ERP	Billing
Customer	Create	Use	Use
Product		Create	
Accounts			Create

- (Look for overlaps between data maintained by different applications.)

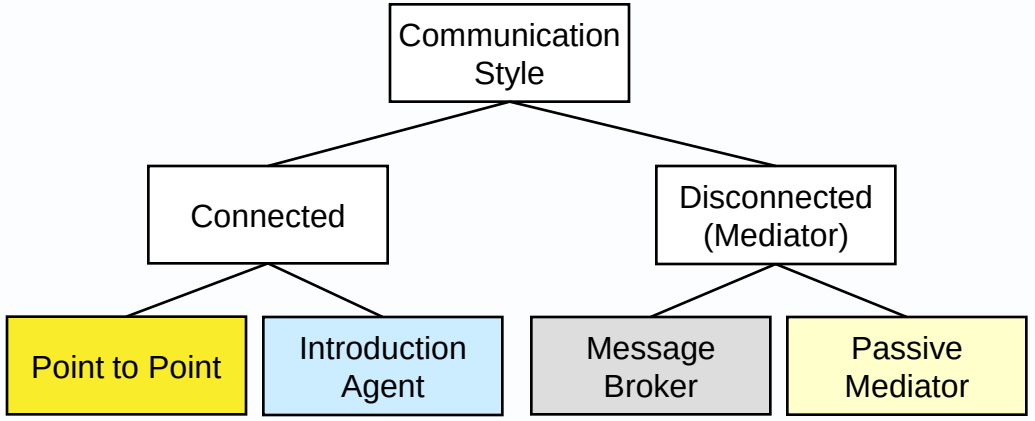
Application interaction diagram

- [an artefact] that shows how applications inter-communicate to enable a process.
- It is often used to examine where time is spent in or between application processing steps.
- Typically drawn as an interaction or sequence diagram.



7.4 Application Communication Patterns

- **Application communication style** [a pattern] in which a client/sender application (or other actor) connects to a server/receiver application (or other actor).
- Two broad communication styles, each subdivided into two narrower styles, are listed below.
- There are other subcategories, not listed here.

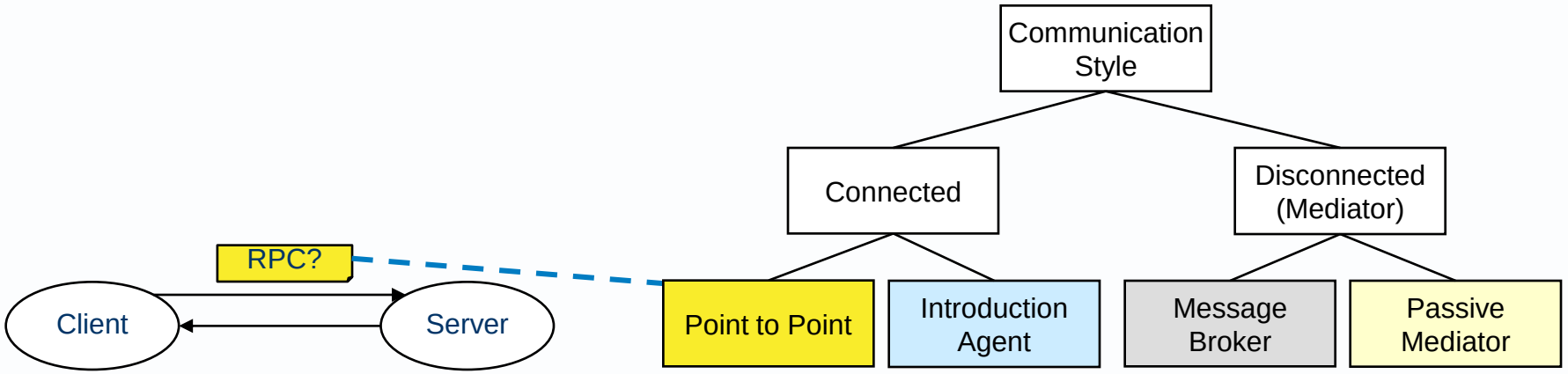


Tightly-coupled

Loosely-coupled

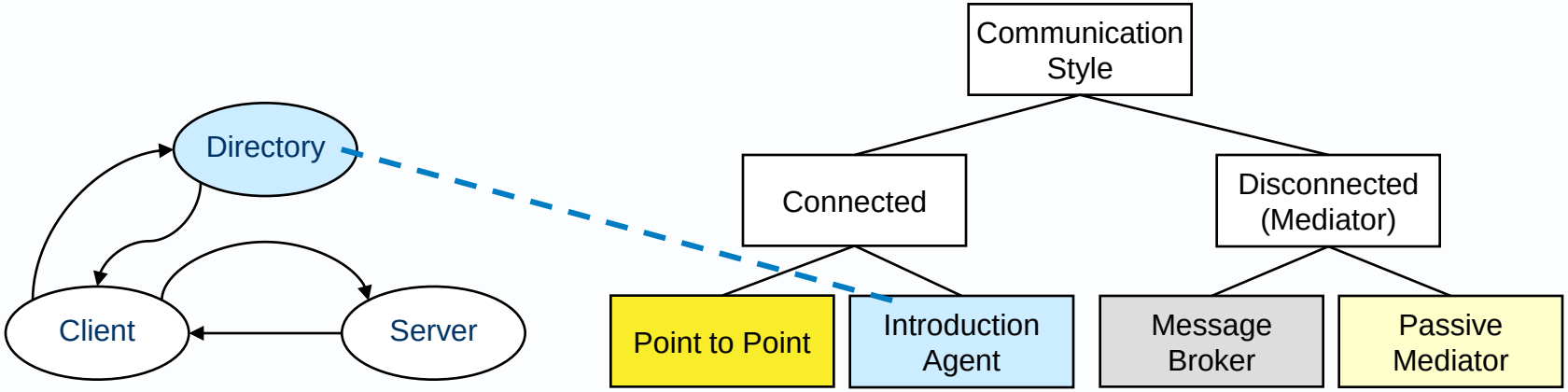
Direct connection

- a pattern] in which clients/senders talk directly to servers/receivers. There are two subcategories below.
  - **Point-to-point connection** [a pattern] in which a message sent by one client/sender is received by one server/receiver. The client/sender knows the location of the receiver. The client knows what protocols and data formats the server/receiver understands. Strengths: simple and fast. Weaknesses: potential duplication of data transformation and routing code, reconfiguration costs on receiver address changes.



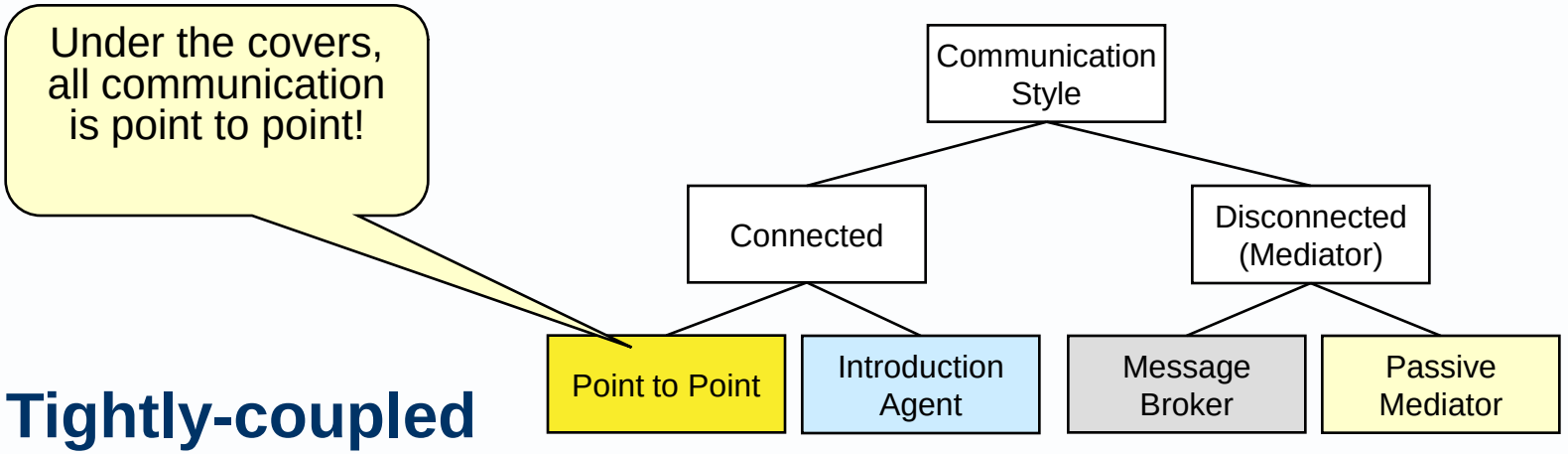
Direct connection

- a pattern] in which clients/senders talk directly to servers/receivers. There are two subcategories below.
  - **Direct broker connection** [a pattern] in which parties willing to communicate are registered (with end point locations) in a directory. When a client/sender wants to send a message to a server/receiver, the broker makes the introduction, and may establish client-side and server-side proxies. From then on, the parties talk directly or through proxies, as though using point-to-point connection. Not so simple and fast, but decouples clients/senders from server/receiver locations.



Indirect communication

- [a pattern] in which clients/senders never talk directly to servers/receivers, they talk only through a mediator or shared resource. There are two subcategories.

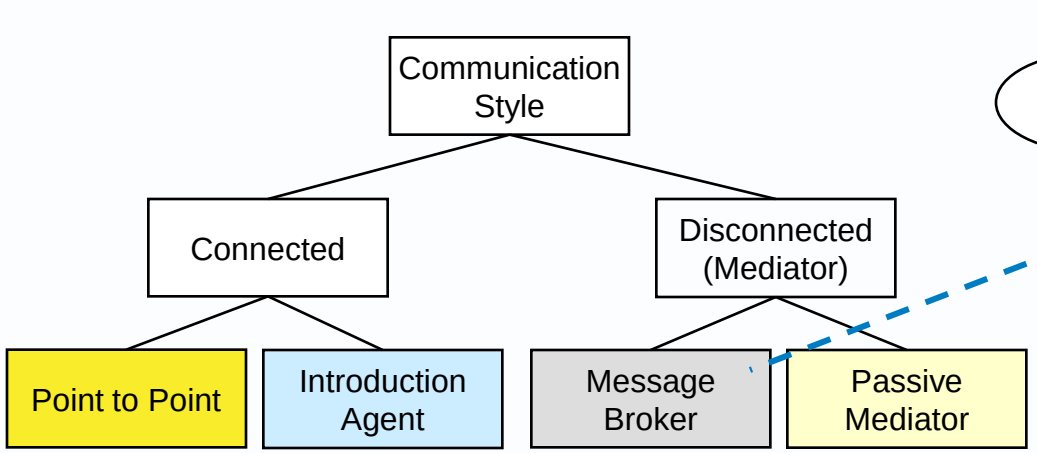


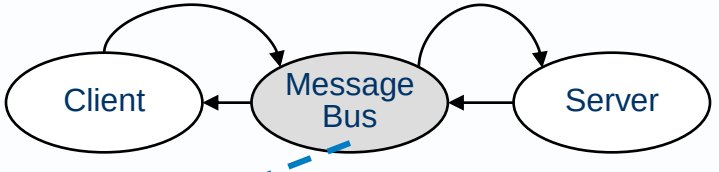
Tightly-coupled

Loosely-coupled

Mediated communication

- [a pattern] in which an **indirect broker** decouples communicating parties; it adds a layer of indirection between clients/senders and servers/receivers. This can enable communicating parties to work at different places and different times (asynchronously). It can shield one party from the effects of some changes to the other party. Mediator technologies include message brokers, message routers, message buses and publish-subscribe middleware.
- Aside: The technologies do what email infrastructure does for people, that is, enable them to communicate asynchronously via messages - rather than talk directly over an end-to-end network connection kept open for that conversation.

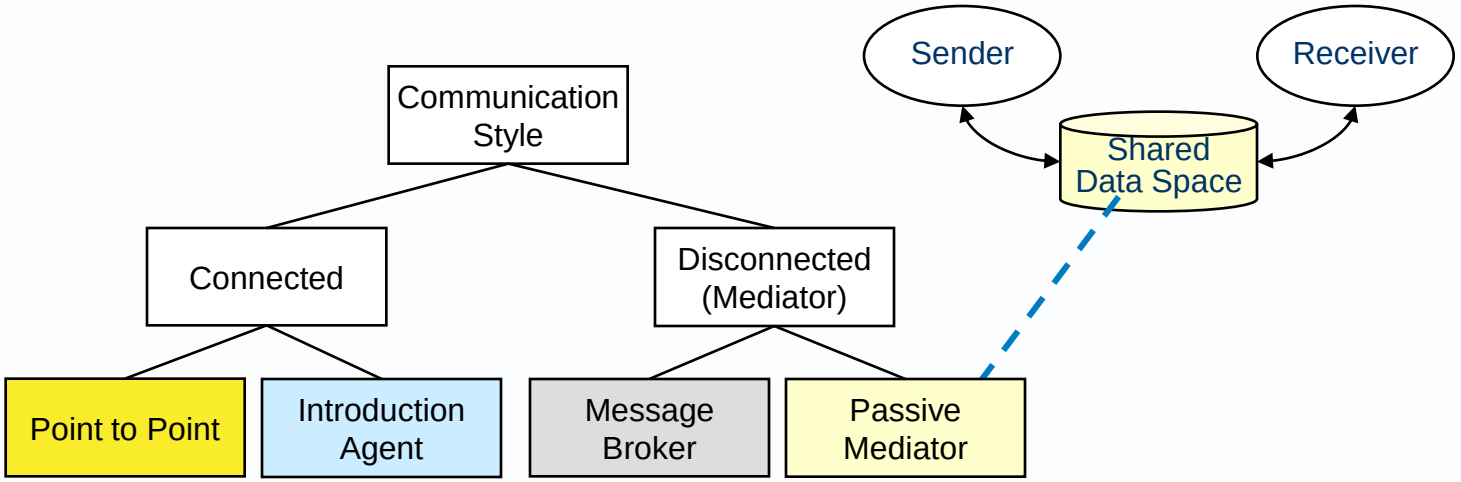




- Message bus = message broker + schema-based distribution

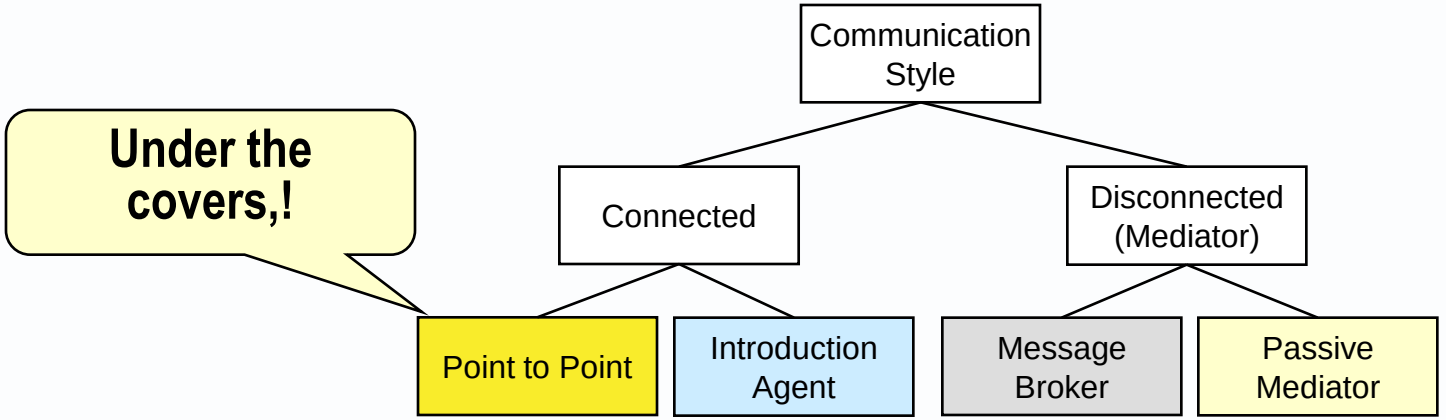
Shared data space communication

- [a pattern] in which parties communicate indirectly by reading and writing messages in a common data store, which might be shared memory, a message queue, a serial file or a database.
- Aka “shared memory”, “space-based architecture” or “blackboard design pattern” or “passive mediator”.

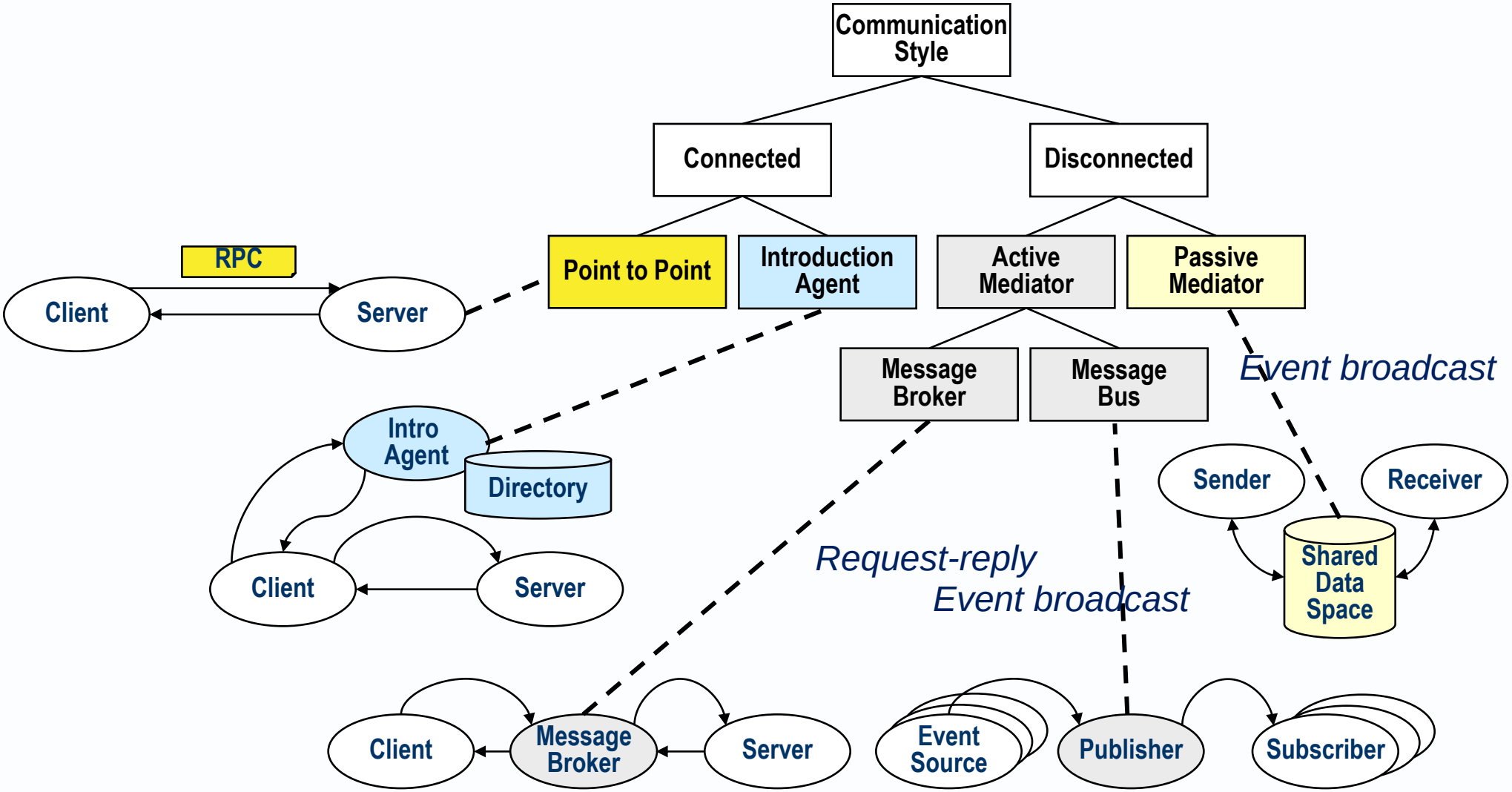


Aside

- Different communication styles may be used at different levels of a communication stack.
- Under the covers is always some point-to-point communication.

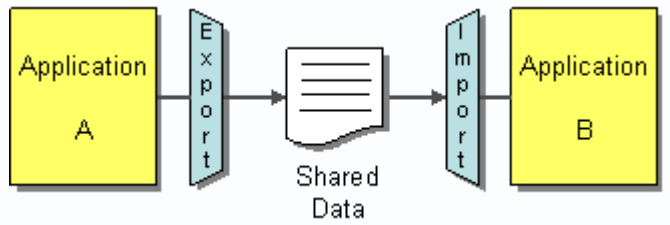


Communication styles – summary overview

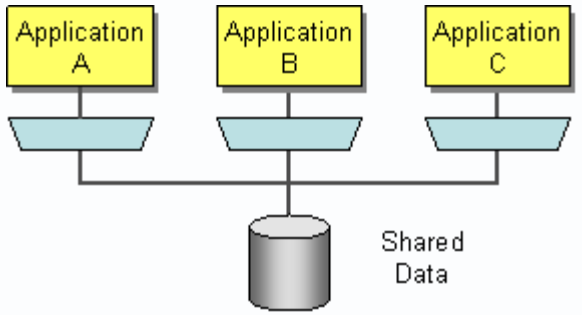


7.5 Applications Integration Tools (rarely examined)

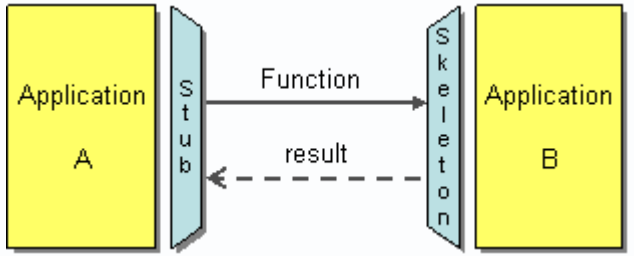
- ETL middleware
- Shared Data Space
- Point to Point (RPC/RMI/ORB)
- Web Service
- EAI/ESB Middleware



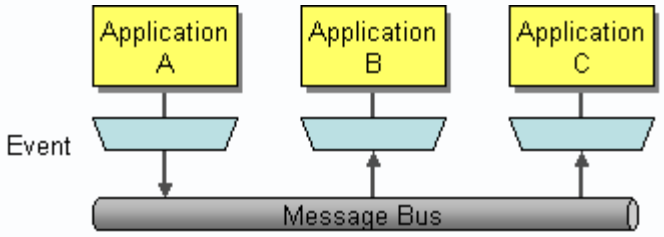
- File Transfer



- Shared Database



- Remote Procedure Invocation



- Messaging

- **ETL tool:** (a platform application component) that helps you to
  - Extract data from data sources/senders,
  - Transform data items from one format to another, and
  - Load the reformatted data into data stores.
- Useful for
  - loading a data warehouse on a regular basis,
  - loading a database during a one-off data migration,
  - moving bulk data between databases

- **Point to point:** See “RPC” and “ORB” in Software Architecture

- **Web Service:** See “Web Services” in Software Architecture.

Message/service bus tool

- (a technology component) that may:
  - manage message queues
  - store, route and forward messages between distributed components
  - transform messages between protocols
  - transform messages between data formats
  - use a canonical data model in data format transformation
  - manage federated/distributed transactions
  - host procedures/workflows that orchestrate distributed components.
  - support EDA using pub/sub mechanisms.

- Using middleware can be more complex and slower than point-to-point integration, but has advantages where inter-component communication is one to many or many to one, and where the components at either endpoint are volatile.

7.6 Applications Integration Patterns (rarely examined)

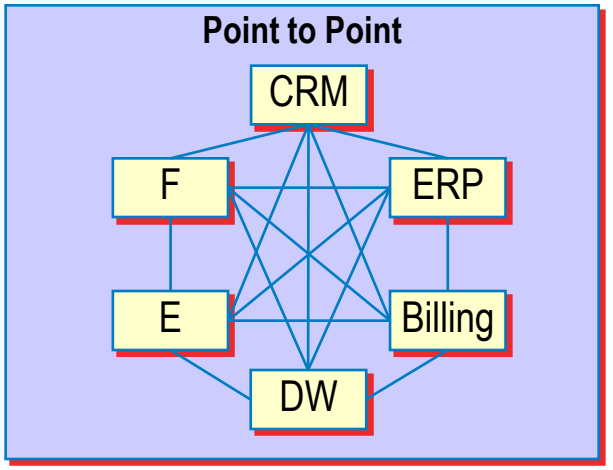
- Applications can be integrated at different layers from UI down to database.
- They can be integrated synchronously or asynchronously.
- Architects must be familiar with integration patterns, and trade off their pros and cons.

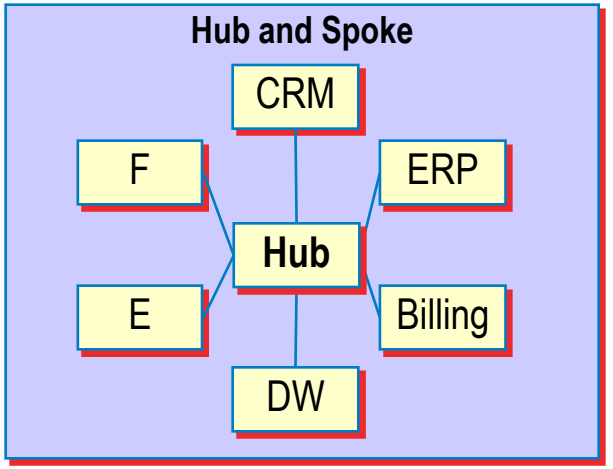
UI level integration

- **User interface integration pattern**
- [A pattern] in which applications are integrated by moving data from one user interface and user interface to another (perhaps using RPA tools).)
-
- **RPA (Robotic Process Automation) tool**
- [A technology component] used to trigger or integrate applications at the user interface level.
- In place of humans, robots complete tasks that involve entering data into applications.
- E.g. receive an email containing an invoice, extract data from it, and enter that data into a book-keeping system.
- It is a kin to a screen scraping for GUI testing tool, but sufficiently resilient, scalable and reliable for use in large enterprises.

On-line integration pattern

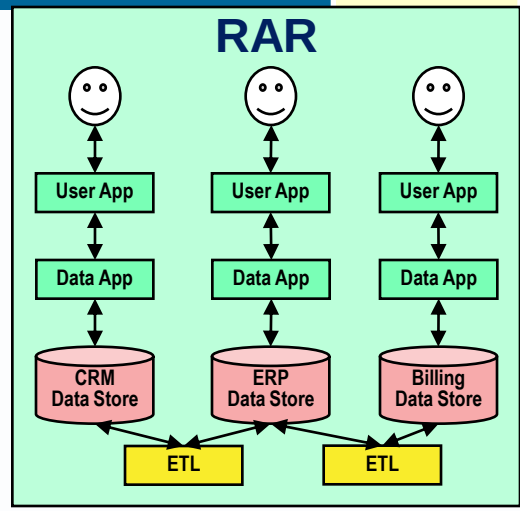
- [A pattern] in which discrete operations and data stores are synchronised on-line,
- using either federated ACID transactions or
- compensating transactions (often using message/service bus tools).





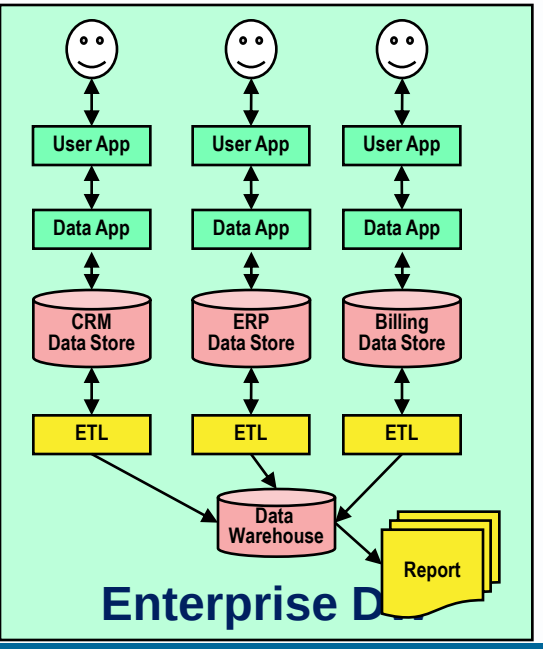
Off-line integration

- [A pattern] in which discrete operations or data stores are synchronised off-line, often by overnight batch processes (often using ETL tools).



Data warehousing

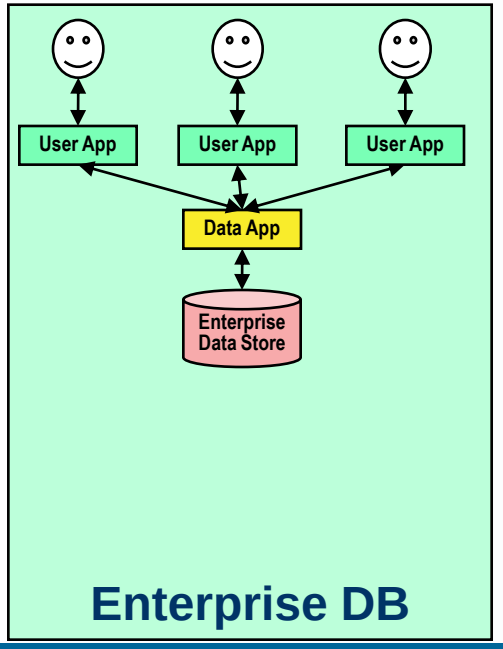
- [a pattern] in which business data is copied from on-line data stores into a central database for reporting, often using ETL tools.
- Data cleansing may be needed at any stage in the process.

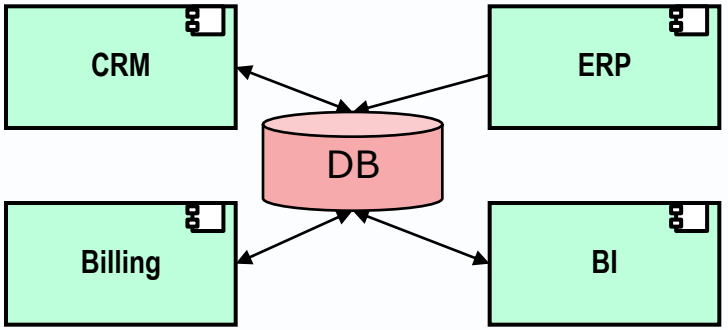


Database consolidation

- [a pattern] in which baseline applications become user application components accessing one shared database.

- ACID transactions ensure consistency





Physical master data

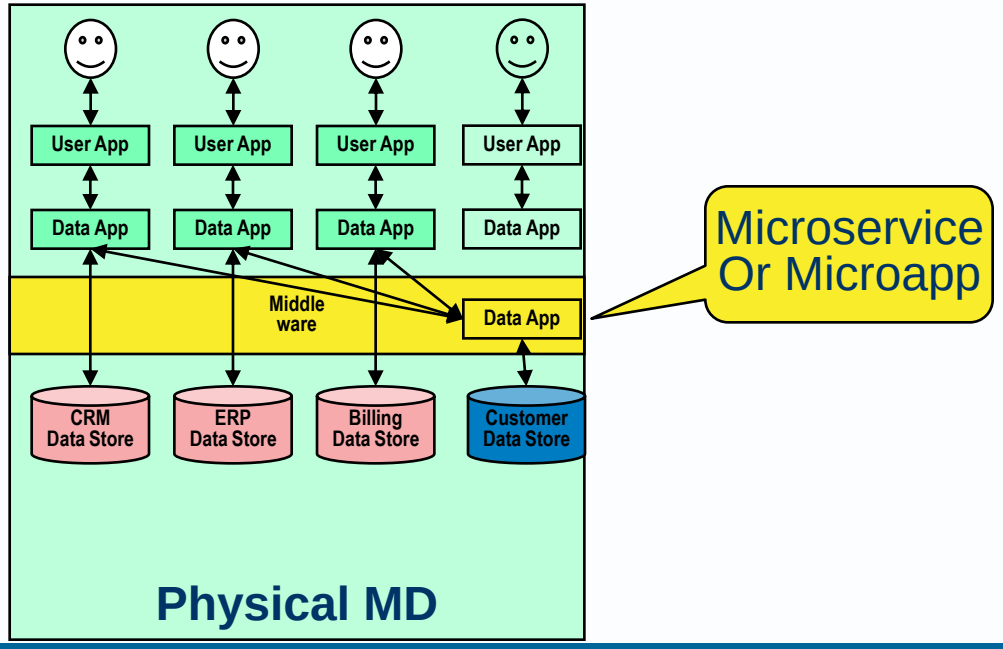
- [a pattern] in which a common data entity is stored in a discrete database, where it can be accessed by any application with a pointer to the common data.

Commonly duplicated data

- Customer?
- Employee?
- Person?
- Product?
- Asset?

Options

- Leave only pointers to new data
- Maintain copies



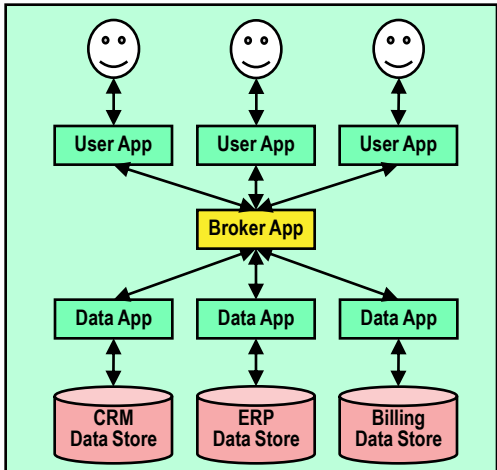
Virtual Master Data

- [a pattern] in which required data can be integrated at run time from several data stores or sources by some kind of broker application. It features three layers of software components.

- **User apps:** present user interfaces, capture events from them and invoke broker apps.

- **Broker apps:** decouple by providing automated business services to user apps, and invoking data services from data app(s)

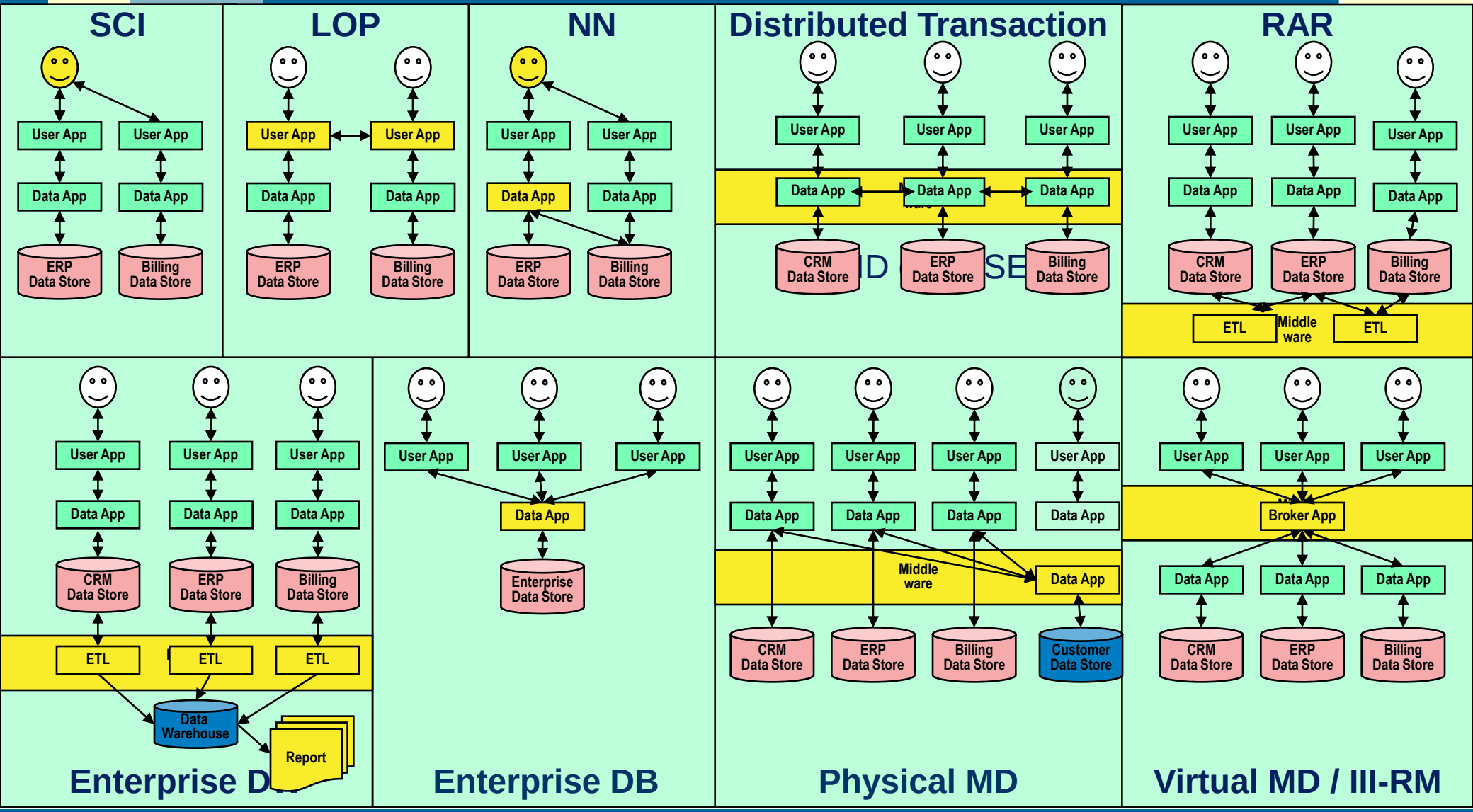
- **Data apps:** provide automated data services to put/get data to/from a particular database or other data source.



Virtual MD / III-RM

Application Integration Patterns

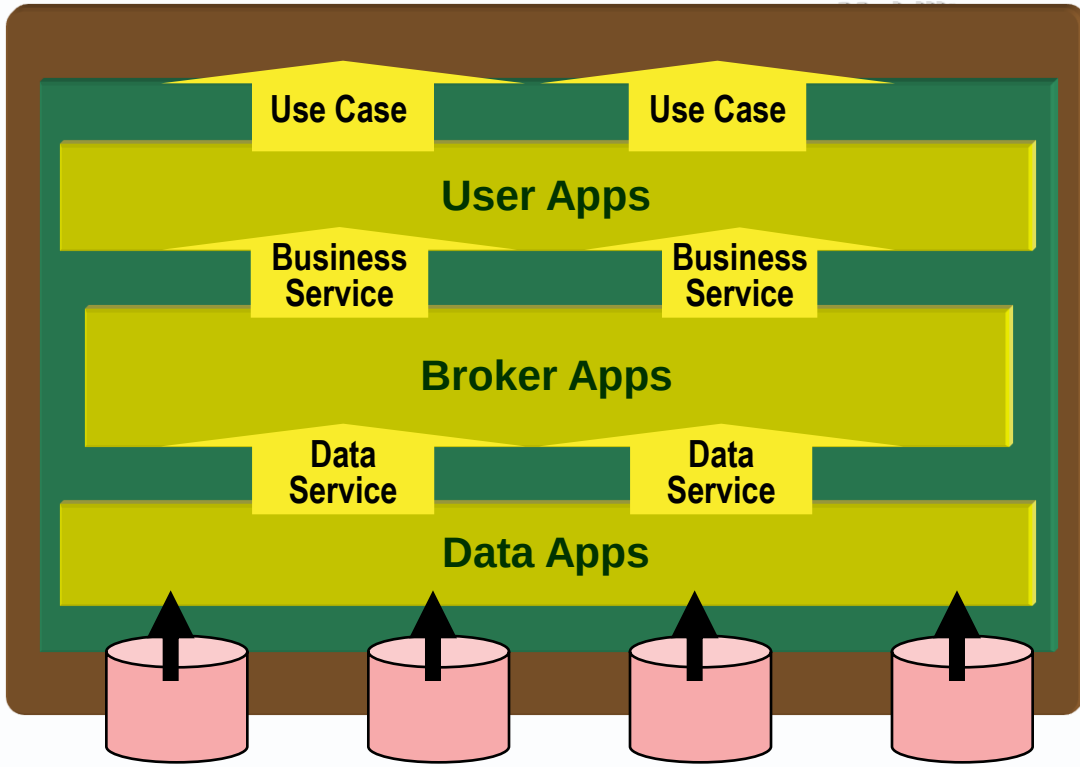
Avancier



App/IS Services in the BCS reference model

Avancier

- **Use Cases**
  - Uses made by users
- **Business services**
  - Automated IS services that are invoked using data types in a canonical data model
- **Data services**
  - Automated IS services that need to understand data types in a local data sources



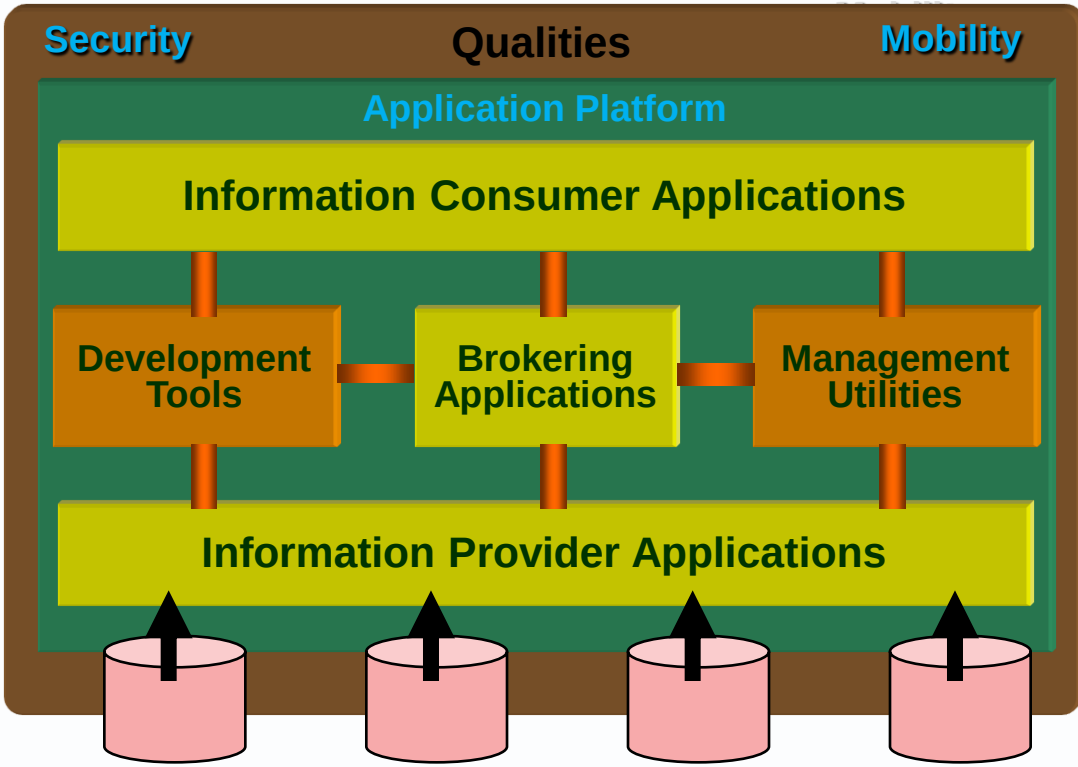
The III-RM in TOGAF (Integrated Information Infrastructure RM)

Avancier

- **Information Consumer Applications**
  - deliver content to the user of the system,
  - provide services to request access to information in the system on the user's behalf
- **Brokering Applications**
  - manage the requests from any number of clients
  - to and across any number of Information Provider Applications
- **Information Provider Applications**
  - provide responses to client requests
  - and rudimentary access
  - to data managed by a particular server

- The overall set creates an environment that provides a rich set of end-user services for transparently accessing heterogeneous systems, databases, and file systems.

- **TOGAF v9**



How to choose between app integration patterns?

- There are always trade offs

Quality goals	
Confidentiality	Low
Integrity	Medium
Availability	Medium
Change requirements	
Budget	Low
Deadline	High
Resources needed	Low

- Evaluate options against the criteria that matter to you