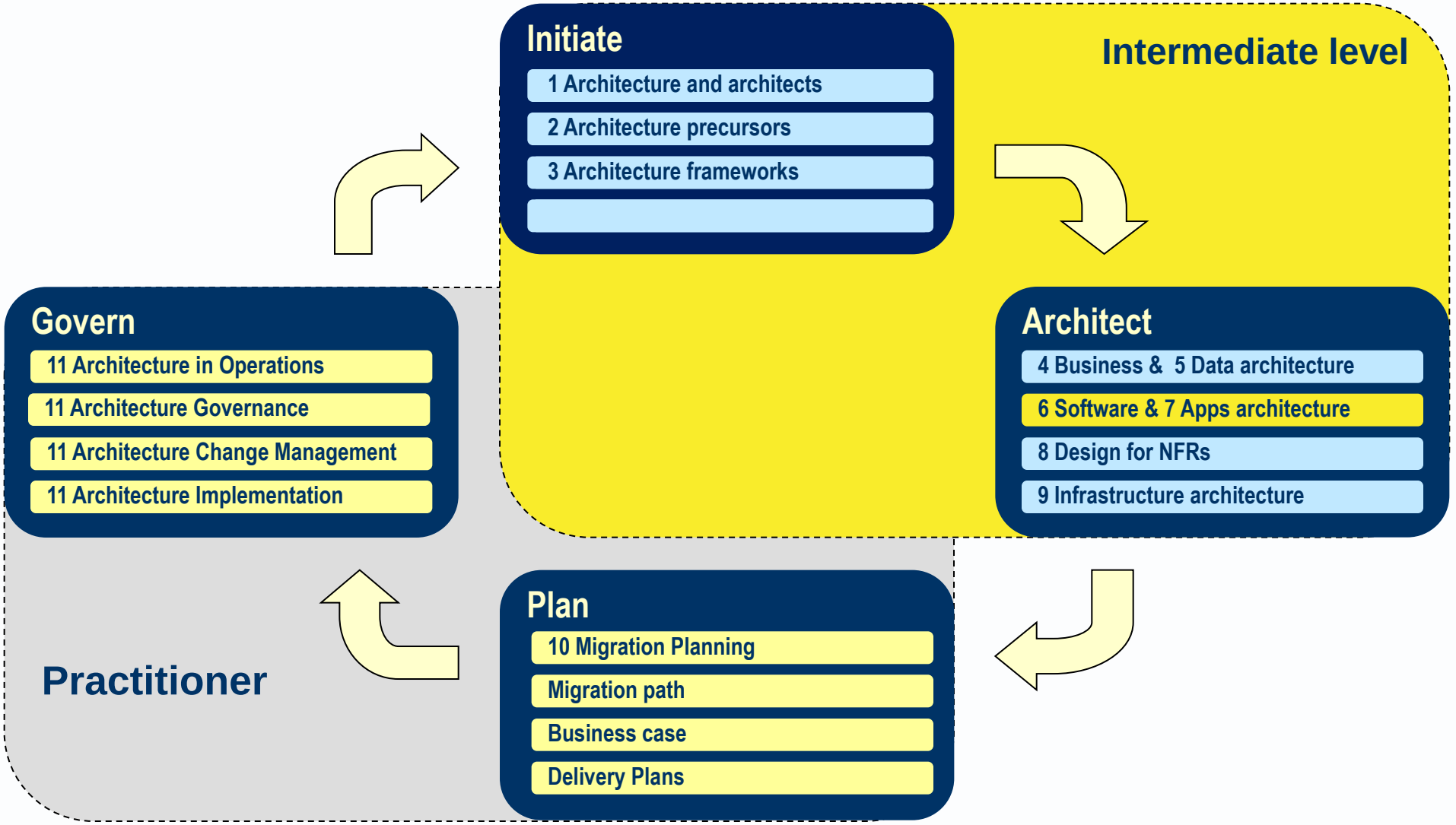
Avancier Reference Model

Applications Architecture (ESA 7)

7. Applications architecture

Avancier



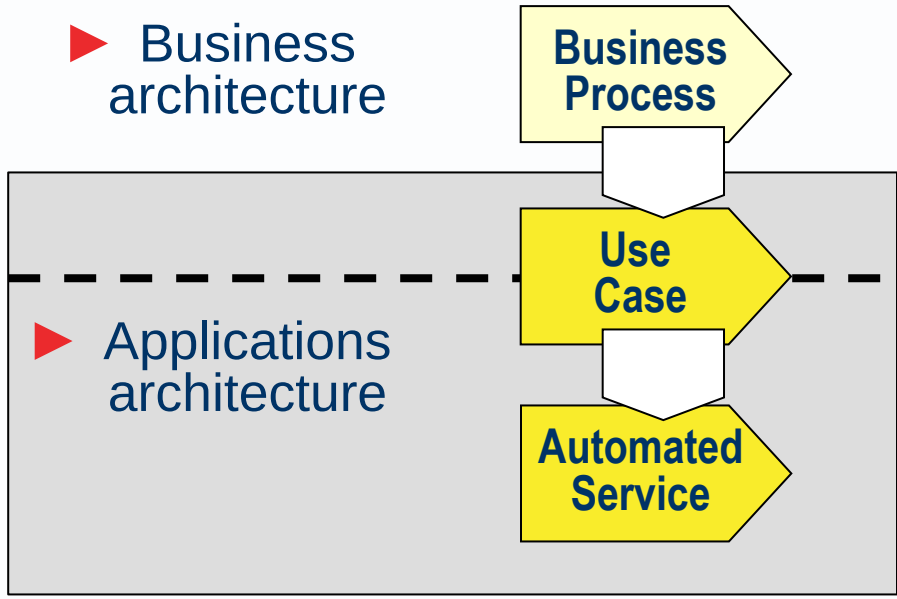
7.1 Foundation

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

IS/App service

- [a service] that can be requested of a business-oriented application component, by a human actor or another application component.
- It could be
  - a use case provided by one application to an end user, or
  - a fully automated service provided by one application to another application.

- IS/App service examples include:
  - check customer credit, provide weather data, consolidate drilling reports,
  - create policy, pay premium, register claim.



Application interface

- [an interface] a collection of application services accessible by application clients
- It identifies services, may provide access to them, and hides what performs them.
- It may be defined in a user interface or API.

Application

- [a component] of business-oriented software (e.g. CRM, Billing).
- It is specified
  - logically by the IS services it provides, and sometimes also by the data entities it maintains, and/or
  - physically as a vendor/technology specific product that can be hired, bought or built.
- It is divisible into three categories: platform, generic and business application.

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.

- Two application classifications follow.

- (The BCS reference model is thin here
- It classifies apps only in these two ways)

Classification by architecture domain

- [a pattern] dividing applications into one of three kinds:
  - **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. See section 9 for more detail.

Compare with classification of applications by type in TOGAF

- Business
- Infrastructure
  - (= generic business apps)
- Platform
  - (= technical infrastructure apps)

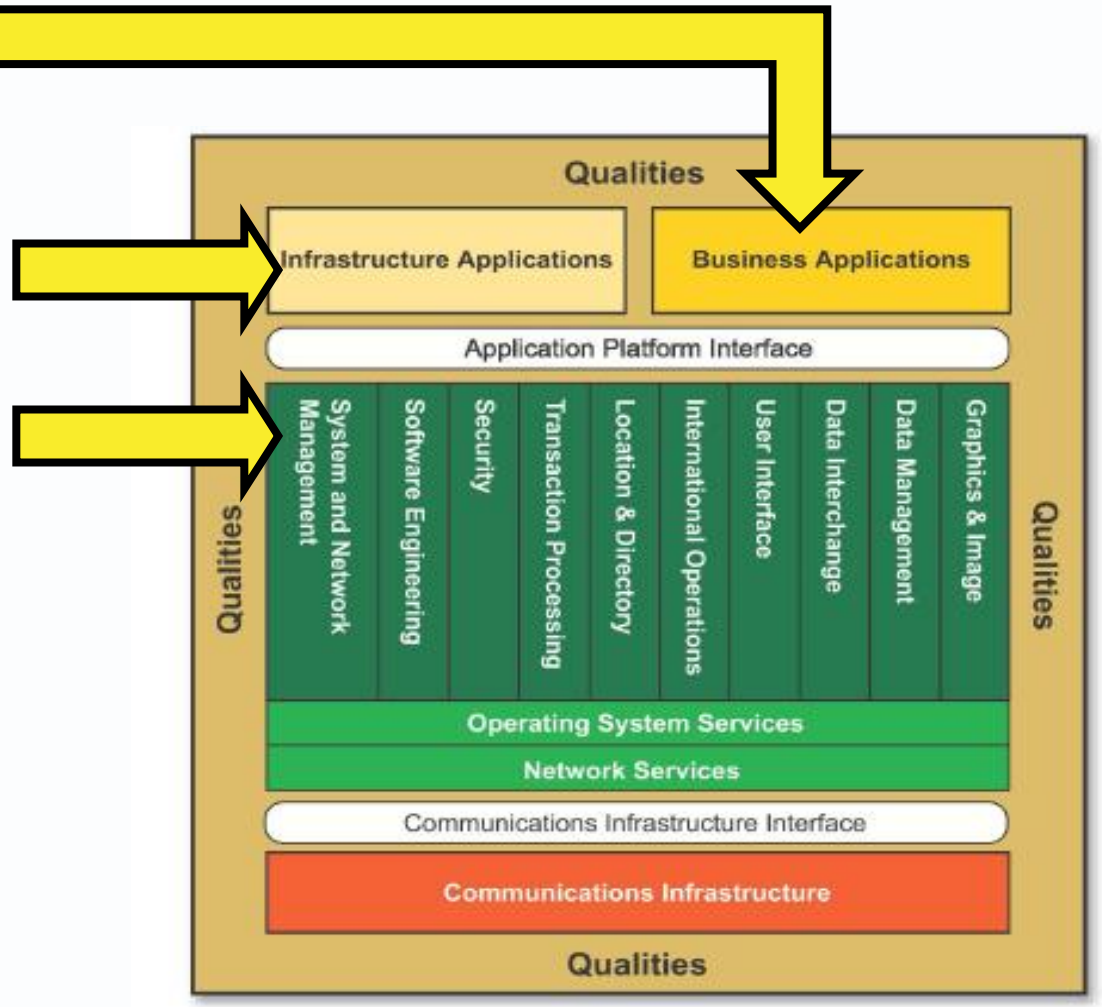


Figure 43-2 Detailed Technical Reference Model (Showing Service Categories)

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 the CRM class of apps

Avancier

- 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 business function categories

- **Billing** [a business application] that sends bills and matches payments received to bills sent.

- **Business intelligence** [a business application] that extracts management information from large quantities of data.

More business function categories - after the NCC

Avancier

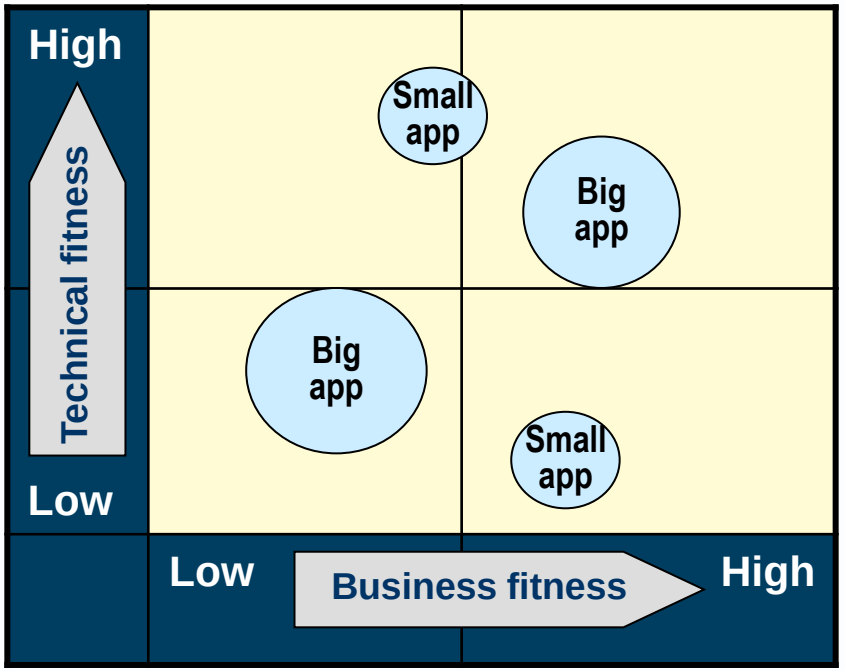
- NCC: Conspectus & Evaluation Centre

  - ERP
  - CRM, Call Centre & Marketing
  - Accounting & Financial Reporting
  - Data Warehousing, Business Intelligence & CPM.
  - Document Management, Content Management & BPM.
  - HR & Payroll
  - Project Management & PSA

Other ways to classify apps in the app portfolio

- Cost, Value (mission critical <>unimportant),
- Business fitness, Technical fitness
- Size, Complexity
- User type: Public / Employee / Technical
- Generality: Universal <> Unique to business
- User base: Single-user - Dept - Enterprise
- Usage style: OLTP / Business Intelligence

- Score or rank apps

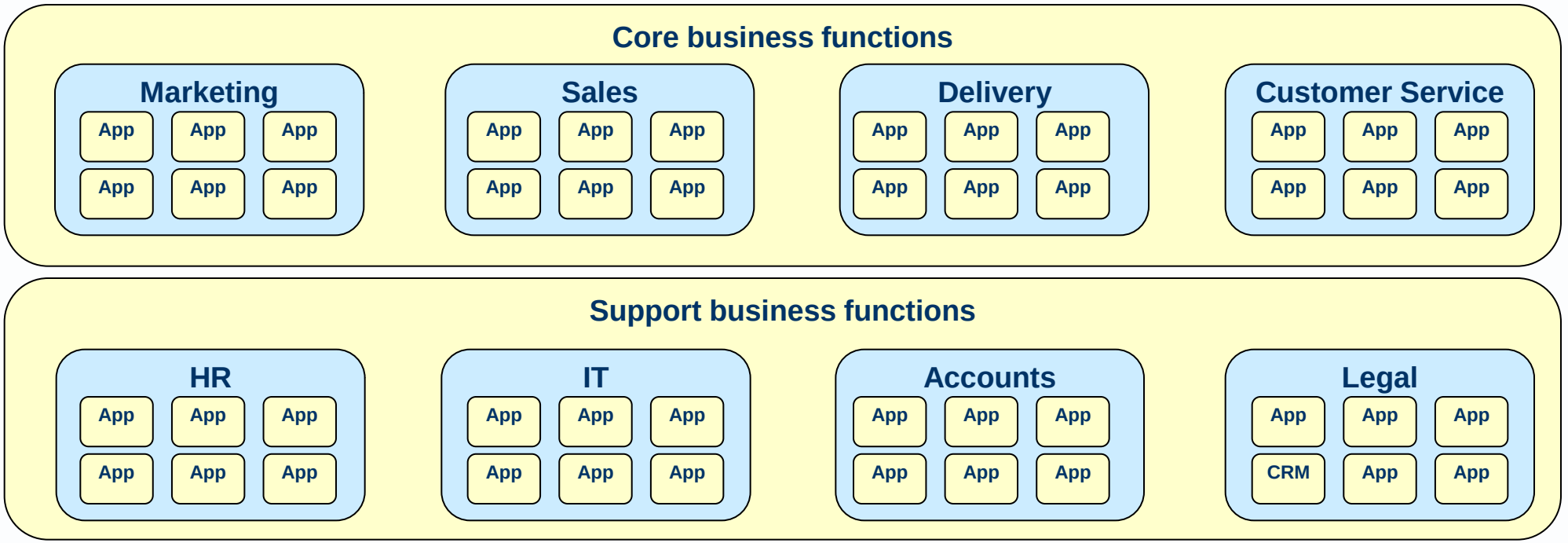


7.3: Application structure

- Application portfolio catalogue
- Applications communication diagram
- Application / data entity matrix

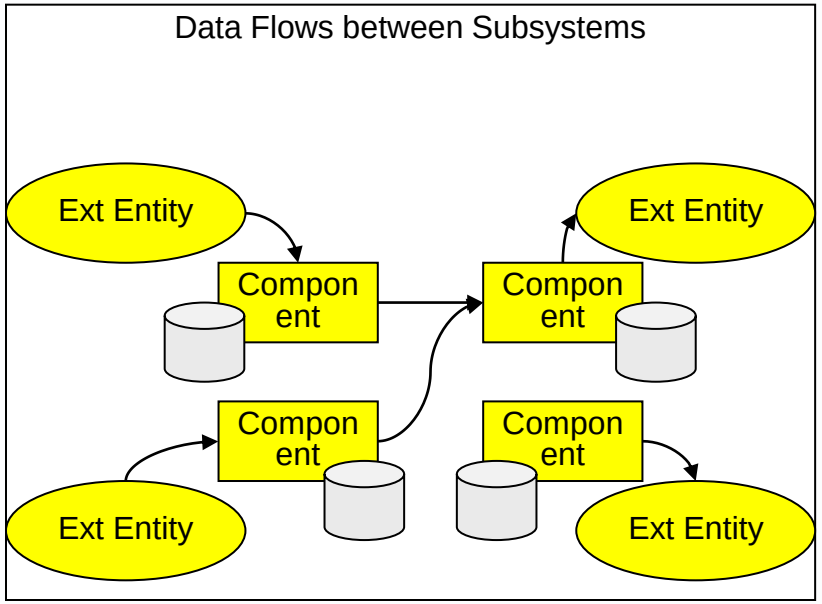
Application portfolio catalogue

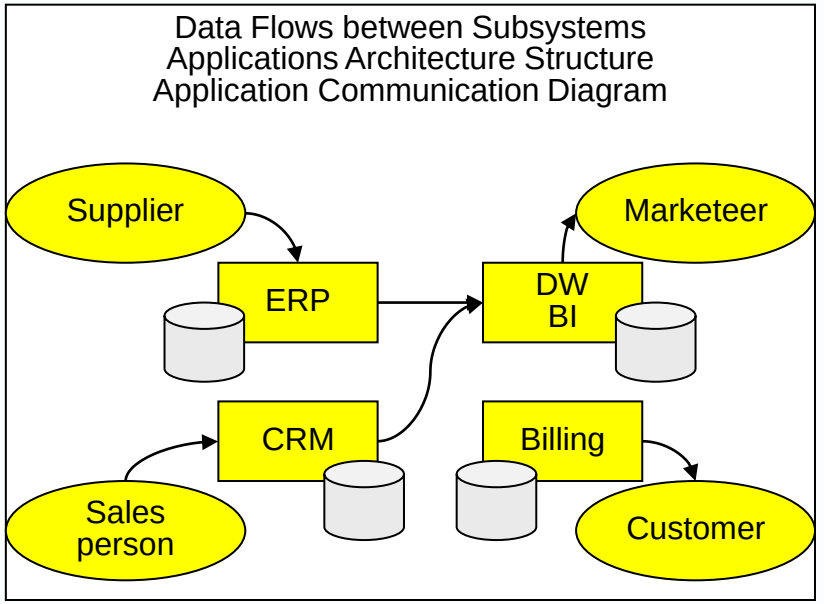
- [an artefact] listing business applications and recording their properties.
- Usually structured so as to reflect the business function hierarchy.



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.

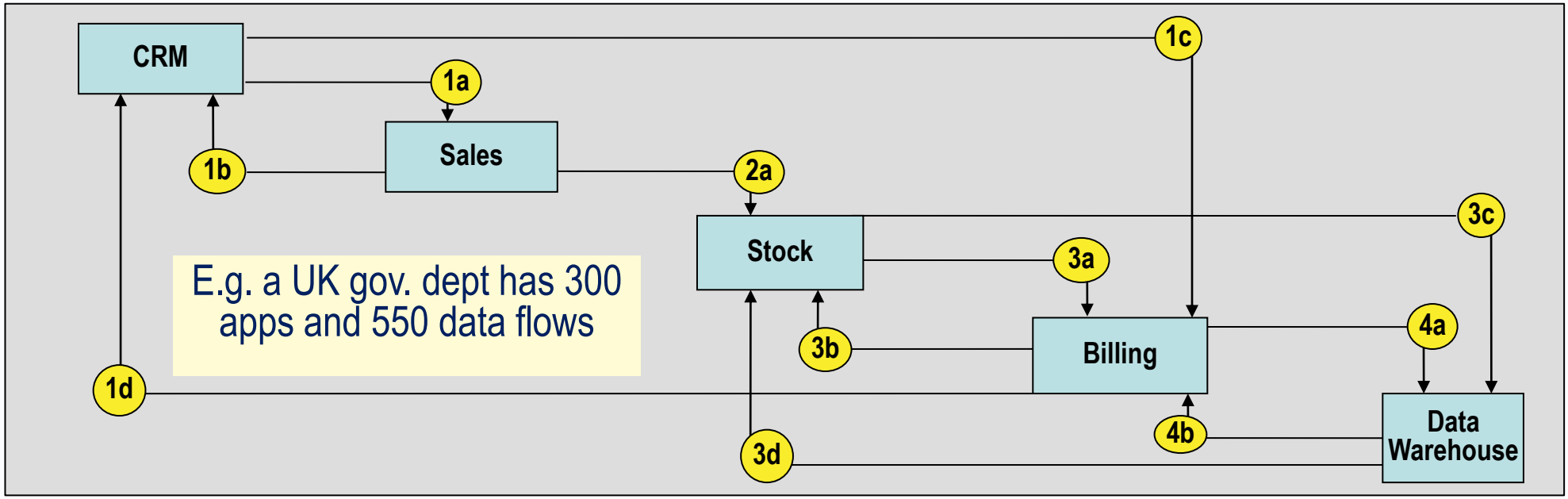




Applications communication diagram - example

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

Application / data entity matrix

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

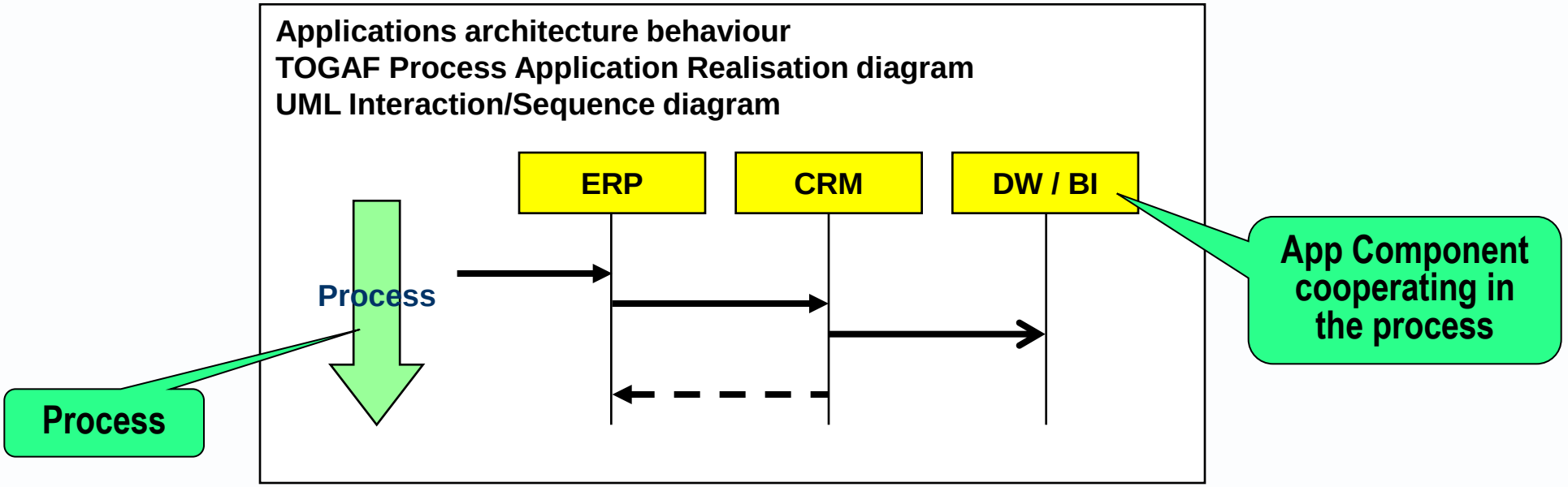
App / Data entity	CRM	ERP	Billing
Customer	Create	Use	Use
Product		Create	
Accounts			Create

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

7.4: Application behaviour

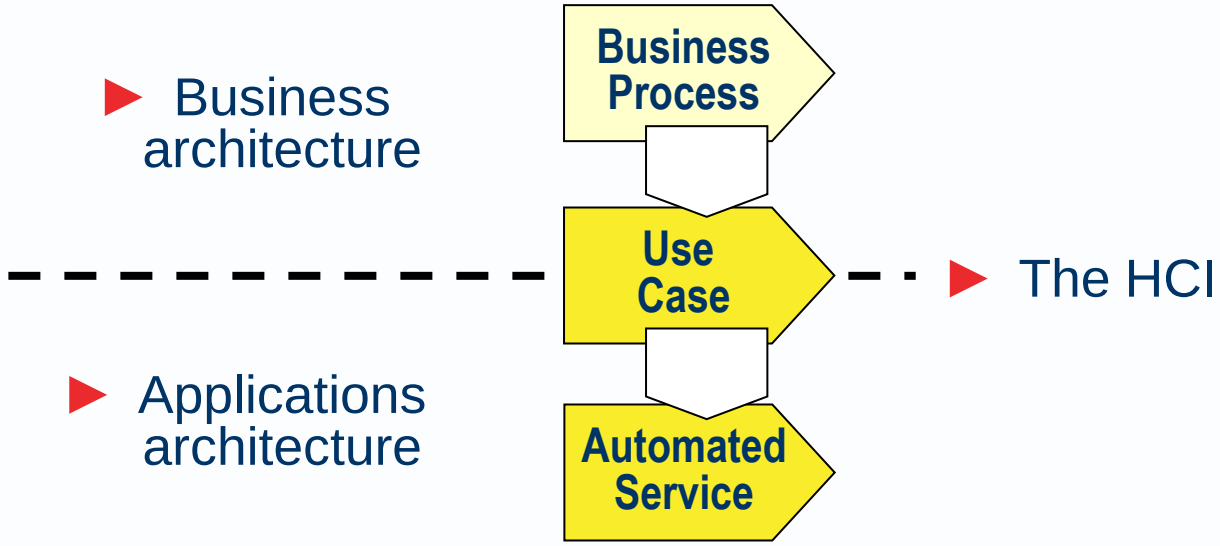
Process realisation diagram

- [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.



IS/App service (REPEAT)

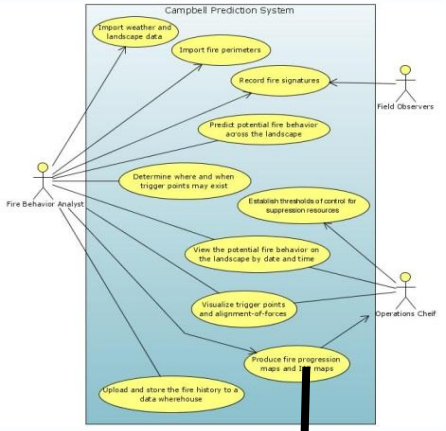
- [a service] that can be requested of a business application;
- a use case provided by one application to an end user, or
- a fully automated service provided by one application to another application.



Use case

- [an IS/App service] at the human-computer interface.
- A use case description defines a use of a system by an actor, typically in the course of an OPOPOT business process or role.
- It is normally named as a goal in verb-noun form (e.g. assess claim).
- It is defined
  - primarily by user role, trigger event, inputs and outputs preconditions and post conditions, and non-functional requirements and
  - secondarily by its process flow in the form of the main path and alternative or exception paths.
- The details of each process step (including automated services or transactions invoked) may be documented separately from the use case description.

Use Case Diagram



Use Case Definition

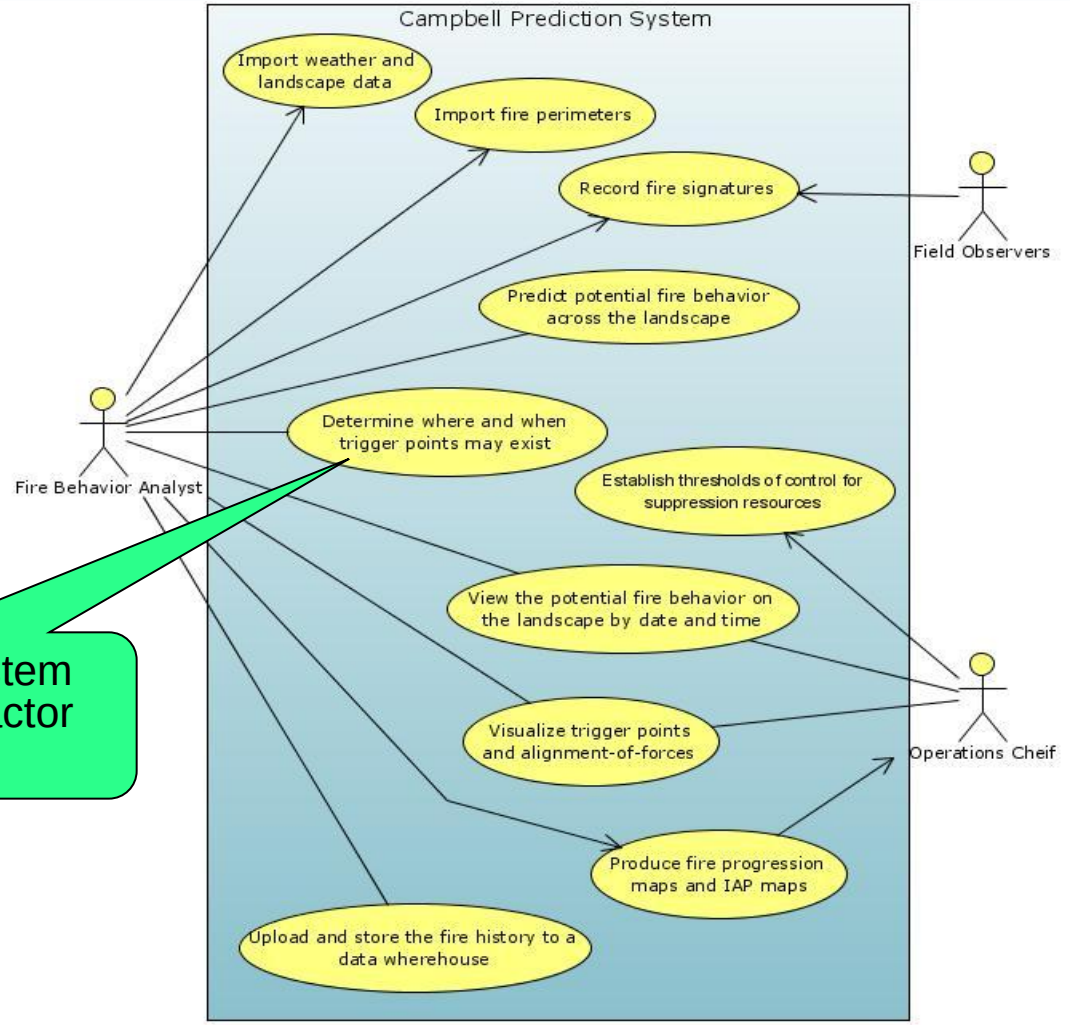
Service Contract
I/O, pre and post conditions

Process main path

Extension paths

Use case diagram

- [an artefact] that shows the uses a actor can make of an application.
- An application is scoped and logically defined by the use cases supports.



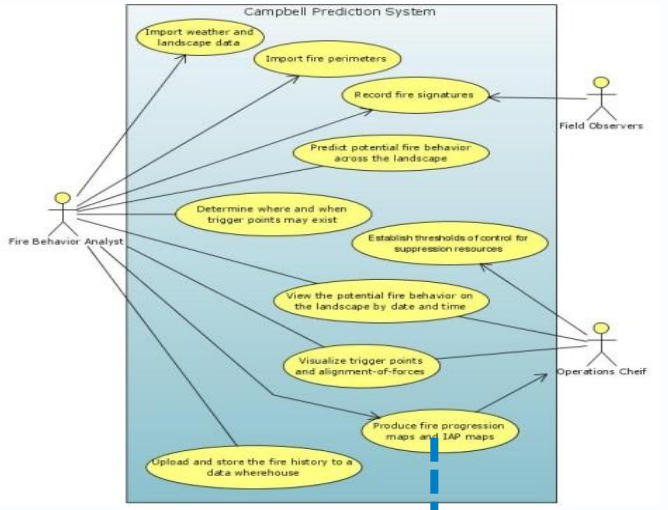
Use case = a system process that an actor engages with

Plenty more such diagrams on the web!

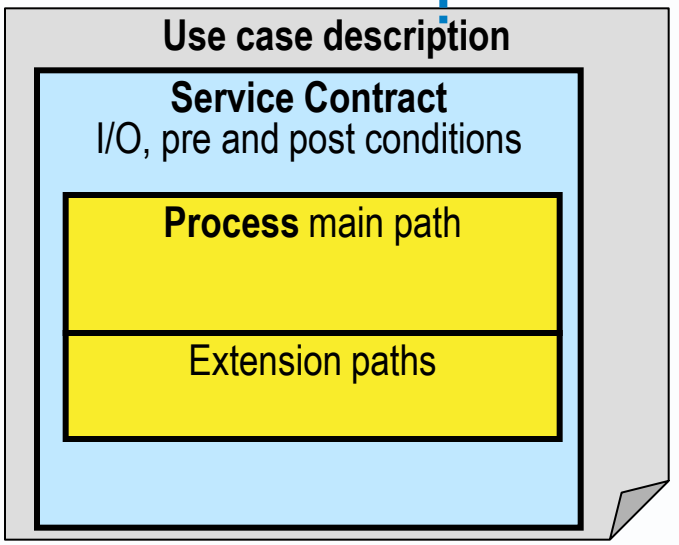
https://kenai.com/attachments/wiki_images/cps/CPS_Main_Use_Case_Diagram_v2.jpg

Describe use cases

- **Application use case diagram**
  - scopes the application system
  - *identifies and names* use cases



- **Use case description**
  - *describes* use cases
  - **Service** offered by the use case
  - **Process** executed during use case
    - Main path
    - Other paths



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

Avancier

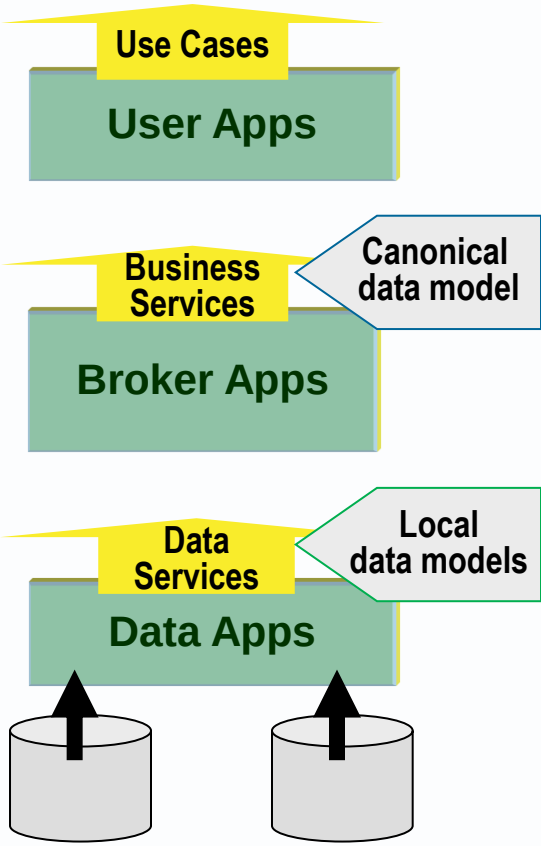
Goal or name	Capture and confirm order
Scope	Kitchen sales system
Actor	Salesman
Stakeholder	Customer
Trigger event	Customer wants to place order Salesman presses order kitchen command
Precondition	Kitchen plan has been populated with kitchen items Kitchen order app is loaded onto the lap top
Post condition	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 service subtypes

- **Automated business service** [an automated service] a kind of automated service whose input and output data is defined in a canonical data model.
- Aside: It is the interface that matters, not the deployment location. So it can be provided by a broker application or an application that encapsulates a data source.

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



Transaction

- [a process] 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] a unit of work that is Atomic, Consistent, Isolated and Durable. It can be rolled back if a specified precondition is violated. Using a transaction manager to automate transaction roll back preserves the integrity of stored data and saves considerable design and development effort, but is impossible in loosely-coupled designs. Where a process has an update or output effect that cannot be rolled back, then compensating transactions may have to be designed.

  - **Compensating transaction** [a transaction] a process 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, report cases of data disintegrity.

Automated Service

- Usually,
  - a service of a back-end business component,
  - invoked from a user interface or data flow consuming process,
  - supports and progresses a use case,
  - applies a message to stored business data.
- The back-end component might be code
  - on an app server
  - on a database under our control,
  - on a database under a somebody else control,
  - a 3rd party component of any kind - a web service perhaps
- Usually atomic
  - Transaction management can be applied.
  - So it can be rolled back if any precondition is violated.

- Typically an **ACID** transaction
- So not requiring **compensating transactions**

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**

Remember

- Business processes steps and use cases may be scoped as
  - OPOPOT: One Person, One Place, One Time
- Use cases may be supported by automated services that are
  - Ideally ACID (that is, roll-backable)

7.5: 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.

- **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 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.
-

- **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.
-

- **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, all communication requires point-to-point connection at some level.

7.6: Applications Integration Tools (rarely examined)

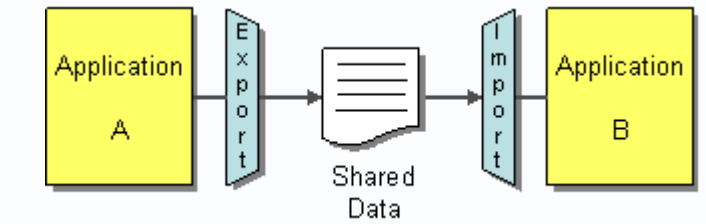
ETL middleware

Shared Data Space

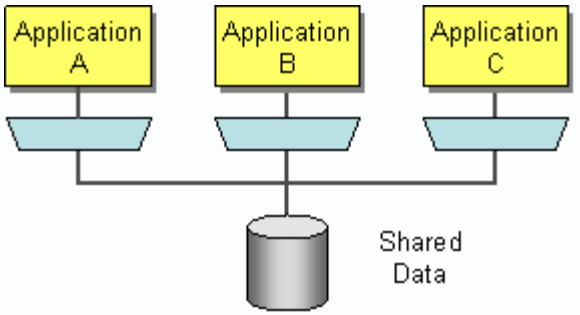
Point to Point (RPC/RMI/ORB)

Web Service

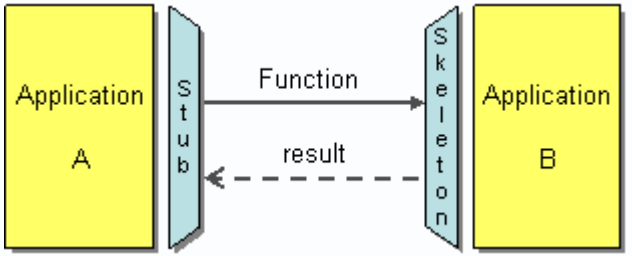
EAI/ESB Middleware



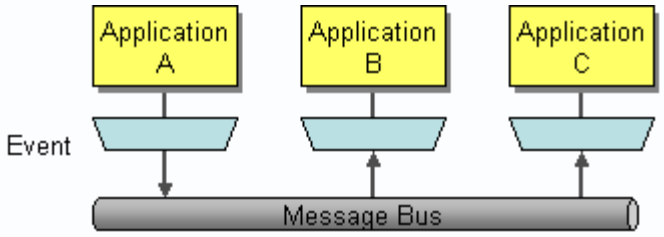
- File Transfer



- 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.

ESB middleware:

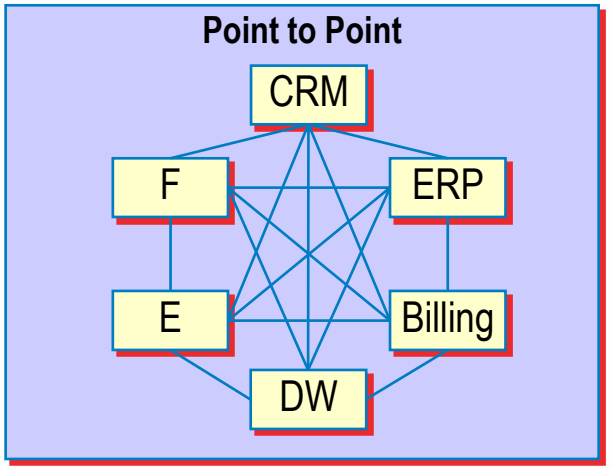
- (a platform application component) that may:
  - store, route and forward messages between distributed components - using message queues
  - transform messages between protocols
  - transform messages between data formats
  - use a canonical data model in data format transformation
  - manage federated/distributed 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.

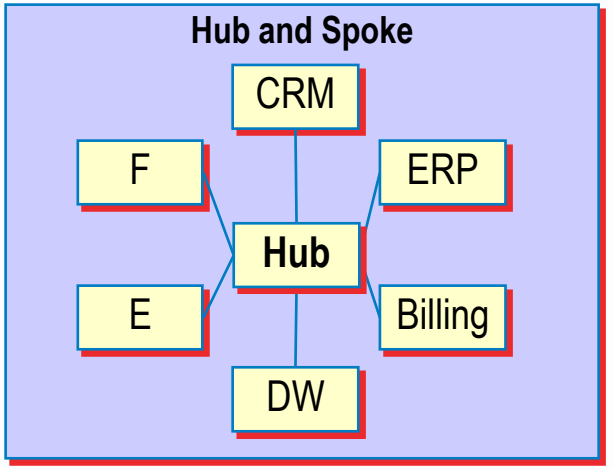
7.7: Applications Integration Patterns (rarely examined)

- **Application integration pattern** [a pattern] for sharing data between applications or their database, including those listed below.

- **Point-to-point integration** [a pattern] in which subsystems talk to each other directly.

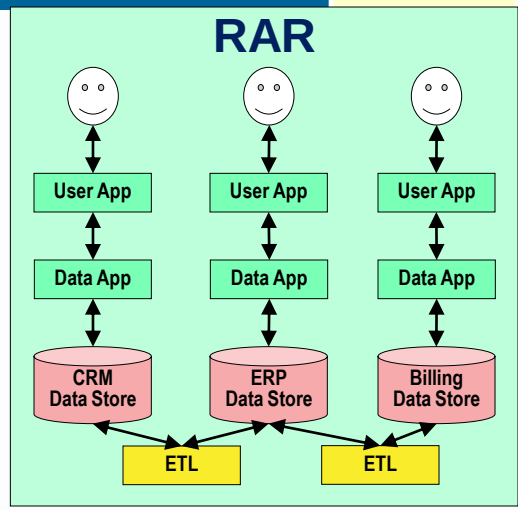


- **Hub and spoke integration** [a pattern] in which subsystems communicate via a mediator.



Off-line integration

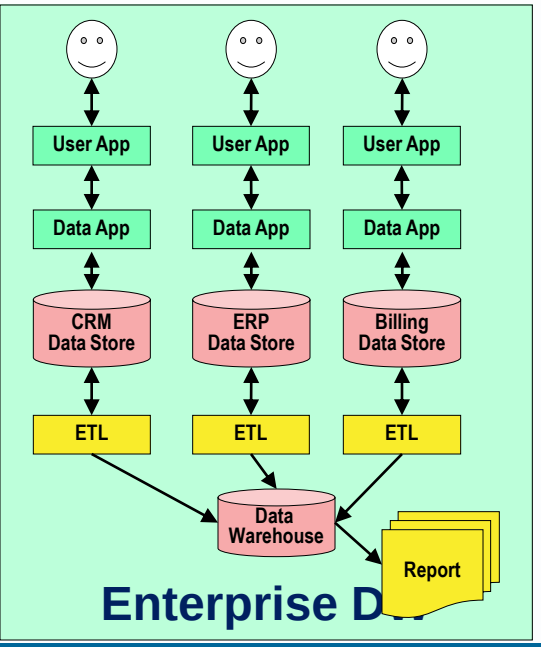
- [a pattern] in which discrete data stores are synchronised off-line, often by overnight batch processes, often using ETL tools.



- BASE – eventual consistency
  - Asynchronous updates
  - Compensation transactions needed!

Data warehousing

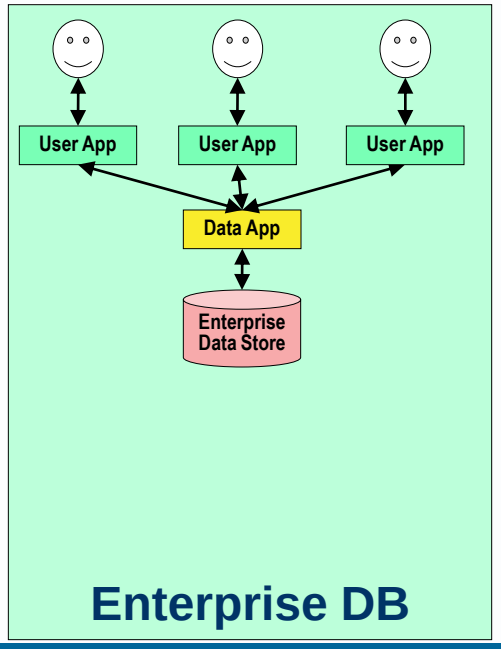
- [a pattern] in which business data is copied from on-line data stores into a central database for reporting, often using ETL tools.
- Data cleansing may be needed at any stage in the process.

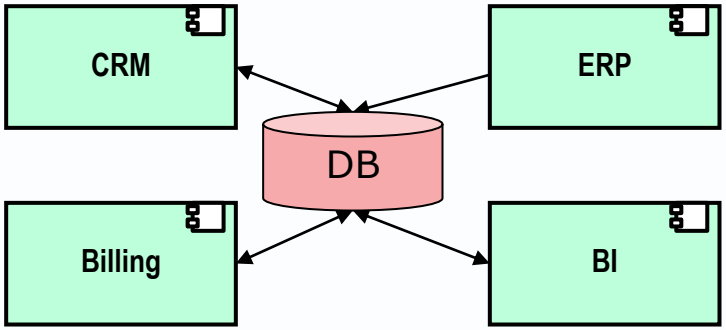


Database/app consolidation

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

- ACID – transaction
  - Consistency assured





Physical master data

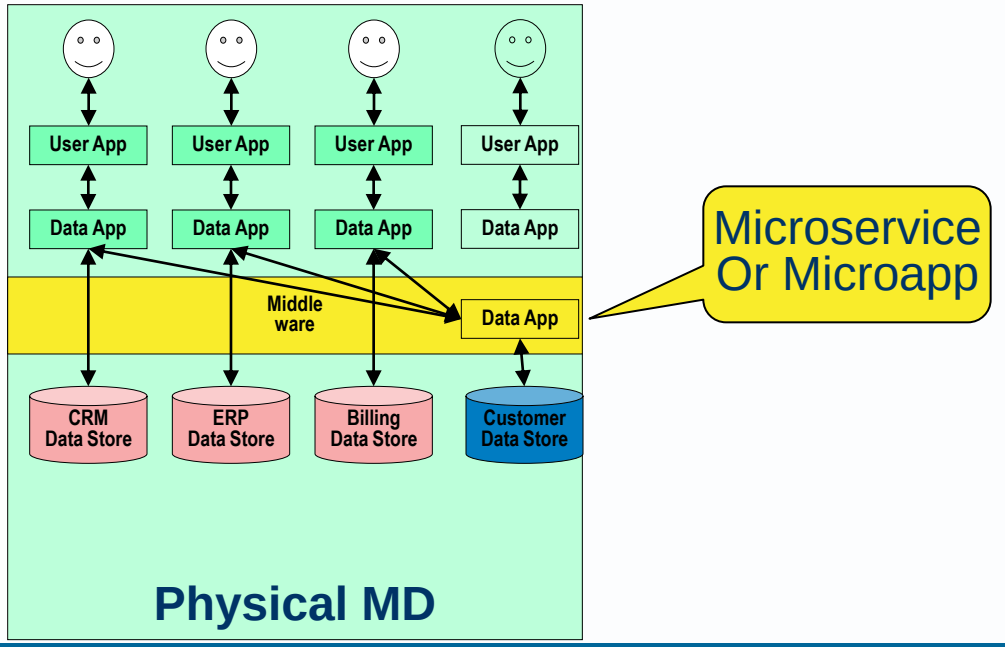
- [a pattern] in which a common data entity is stored in a discrete database, where it can be accessed by any application with a pointer to the common data.

Commonly duplicated data

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

Options

- Leave only pointers to new data
- Maintain copies



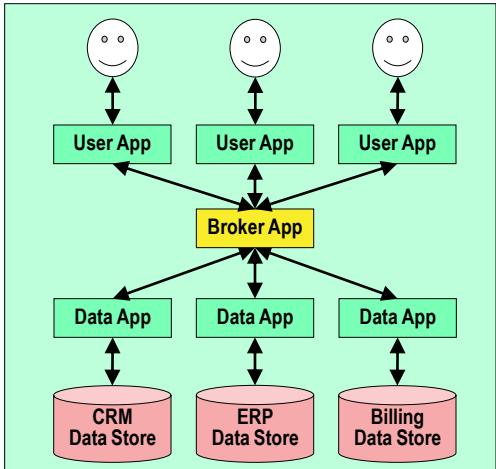
Virtual Master Data

- [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)

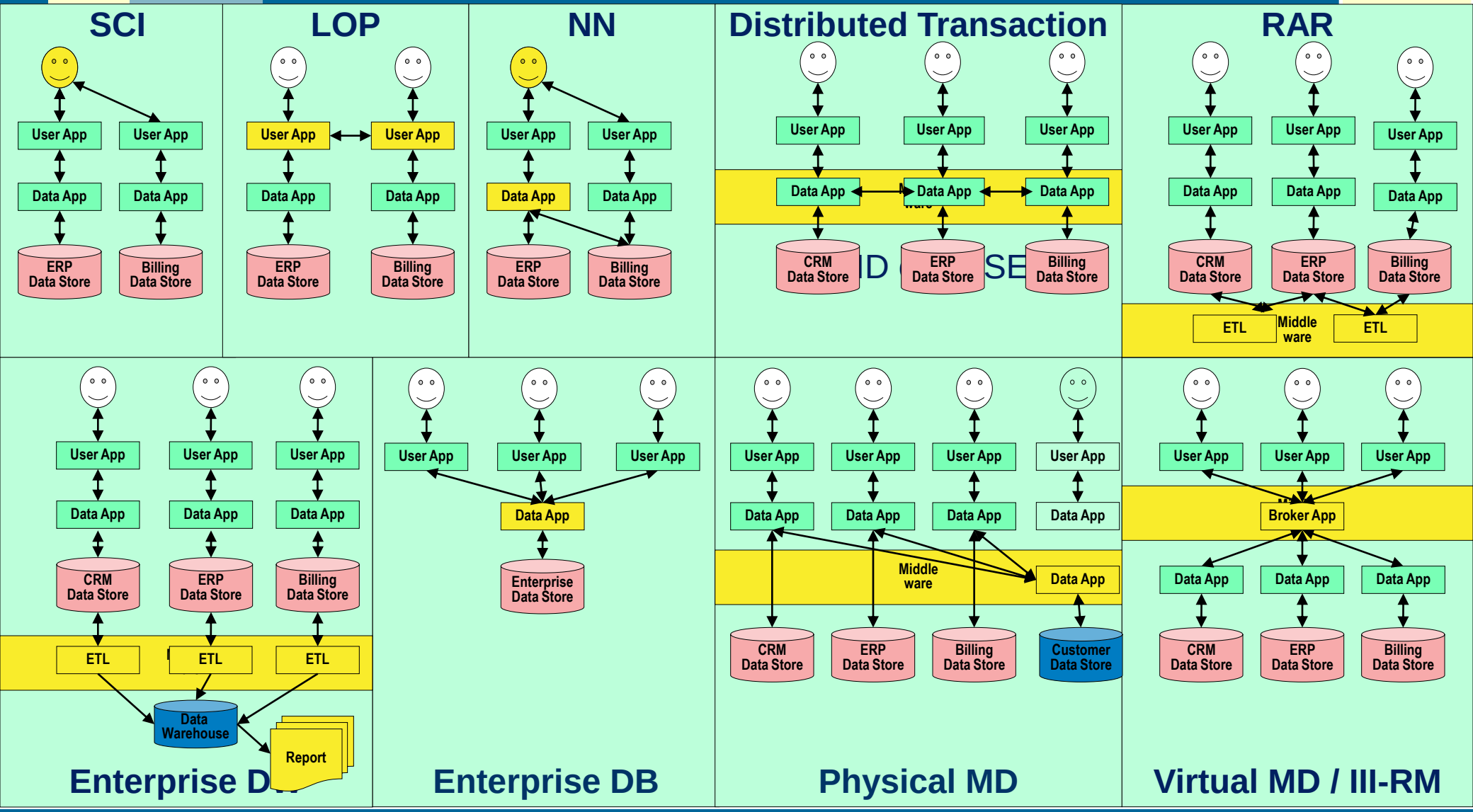
- **Data apps:** provide automated data services to put/get data to/from a particular database or other data source.



Virtual MD / III-RM

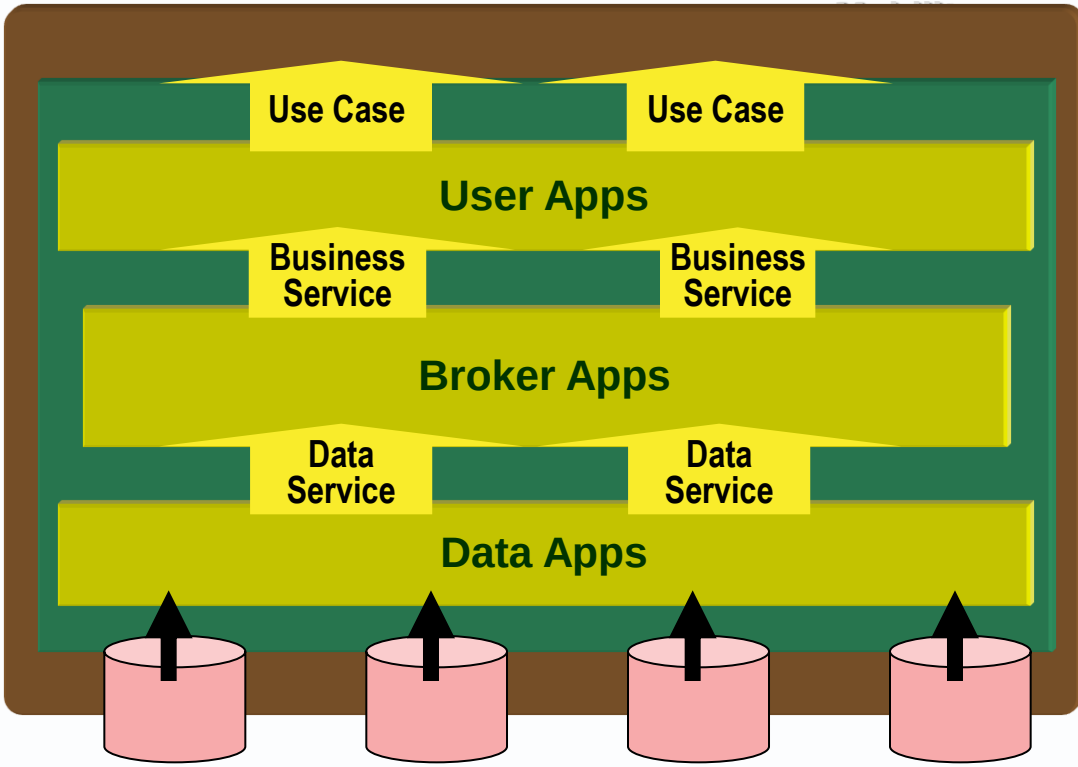
Application Integration Patterns

Avancier



App/IS Services in the BCS reference model

- **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

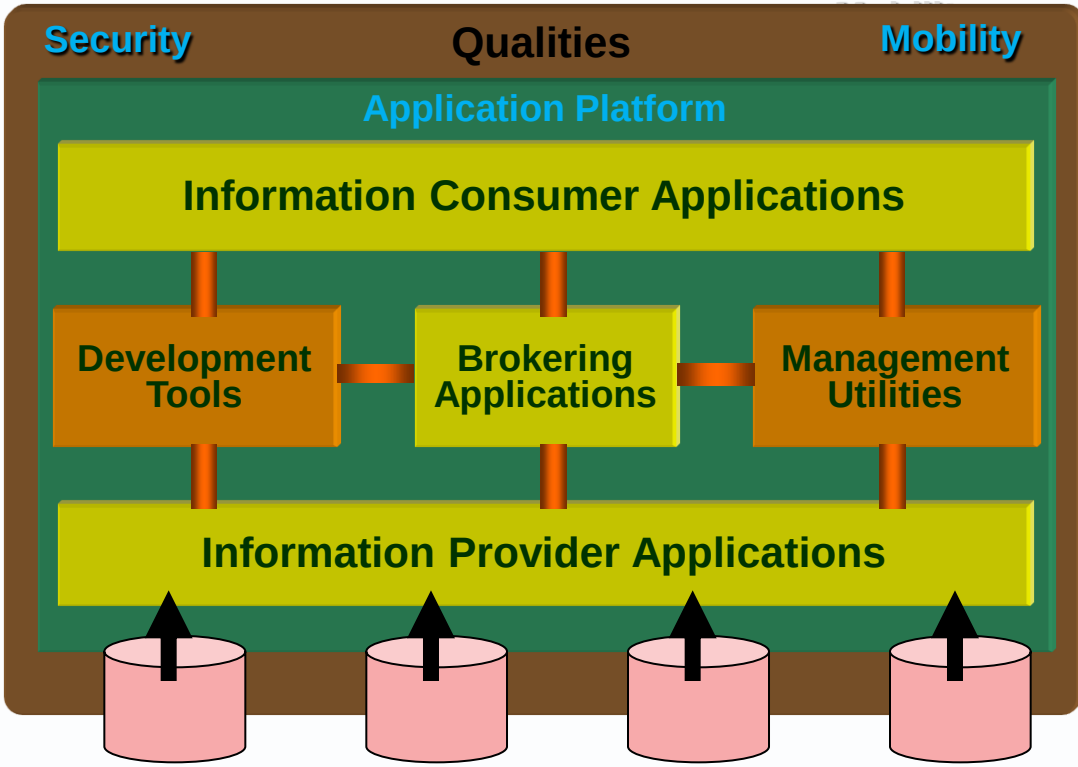


The III-RM in TOGAF (Integrated Information Infrastructure RM)

- **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