e-Business Technological Infrastructure

406.306 Management Information Systems

Jonghun Park

jonghun@snu.ac.kr

Dept. of Industrial Engineering

Seoul National University

9/20/2007

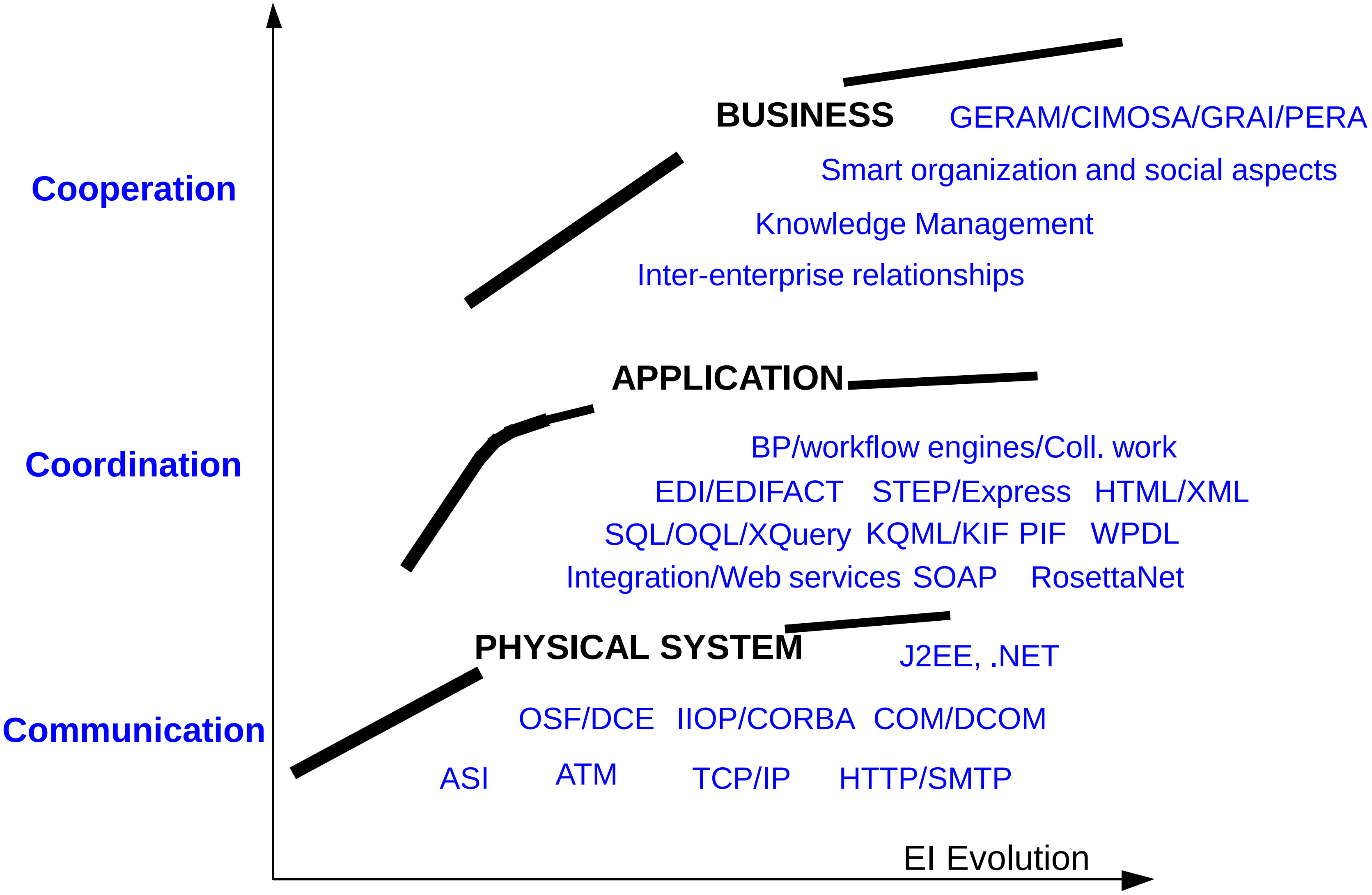
technical e-business challenges

- key mgmt issues in e-business
  - coordination
  - collaboration
  - integration
- ultimate objective of e-business: business process integration
  - interactions between enterprises are achived electronically according to some predefined process which ensures that the business transaction is both meaningful and reliable
  - if all systems are integrated end-to-end, any issues would become immediately visible and appropriate action could be taken
  - however, provision of end-to-end process integration is not an easy undertaking

integration vs. interoperation

- often, the terms integration and interoperation are used interchangeably
- integration: the given components or resources are pulled together into one logical resource with a single schema
- interoperation: the given components work together
- earlier work on addressing the challenges of heterogeneity involved integrating different information resources
    - EII (Enterprise Information Integration)
    - EAI (Enterprise Application Integration)
    - EI (Enterprise Integration): EII + EAI

interoperation levels



BUSINESS GERAM/CIMOSA/GRAI/PERA

Smart organization and social aspects

Knowledge Management

Inter-enterprise relationships

Cooperation

APPLICATION

BP/workflow engines/Coll. work

EDI/EDIFACT STEP/Express HTML/XML

SQL/OQL/XQuery KQML/KIF PIF WPDL

Integration/Web services SOAP RosettaNet

Coordination

PHYSICAL SYSTEM J2EE, .NET

OSF/DCE IIOP/CORBA COM/DCOM

ASI ATM TCP/IP HTTP/SMTP

Communication

EI Evolution

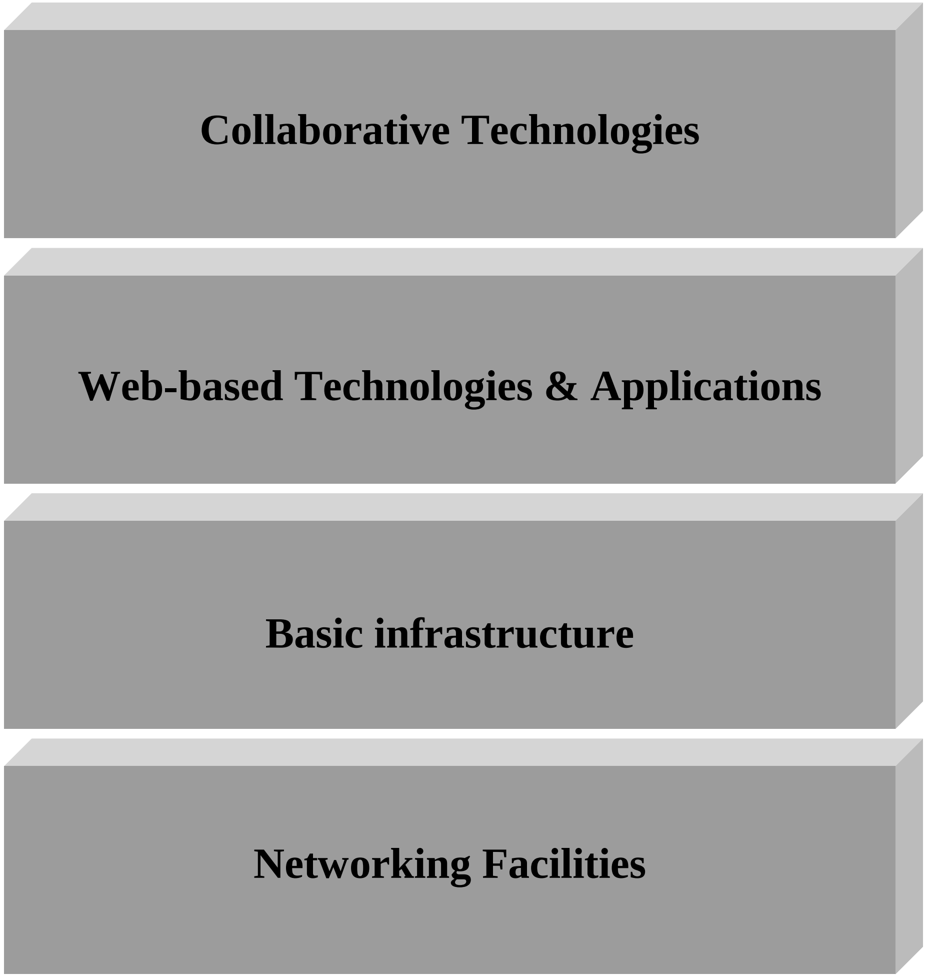
layering

- advantages
  - complexity reduction
  - independence from change
  - component sharing
  - component reuse
  - possibilities for distribution
  - flexibility
- disadvantages
  - performance problem due to the communication overhead

e-Business technical infrastructure

- infrastructure for e-Business comprises technologies that can be seen as different layers that built upon each other:
  - the bottom layer includes networking topologies, the internet, and protocols such as TCP/IP
  - the layer above is the basic infrastructure layer that contains such as client/server and tiered architectures
  - the layer above this contains the technologies that are required to develop web-based applications
  - the top layer contains collaborative technologies such as workflow systems and EDI

technology stack for e-business



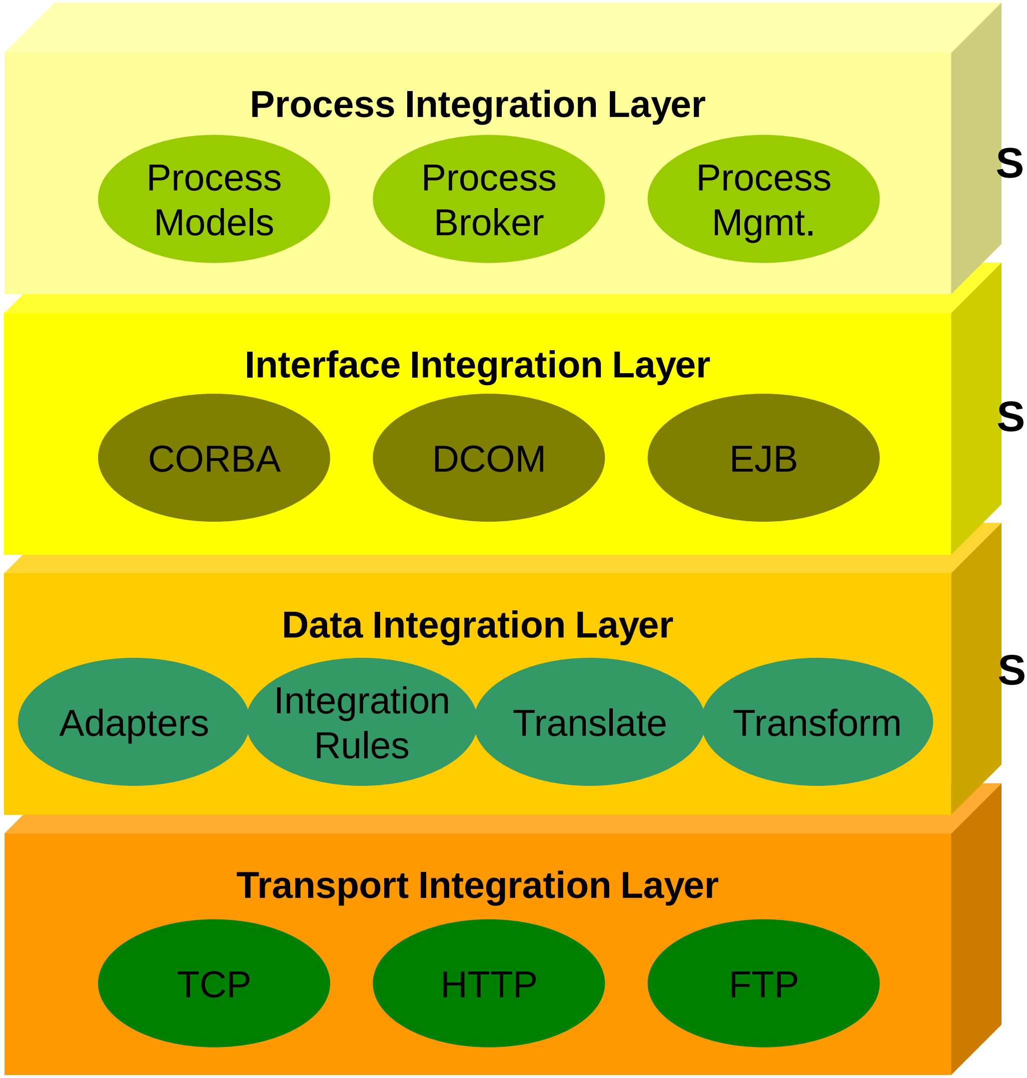
Collaborative Technologies

Web-based Technologies & Applications

Basic infrastructure

Networking Facilities

another technology stack



Sharing of business process

Sharing of business logic

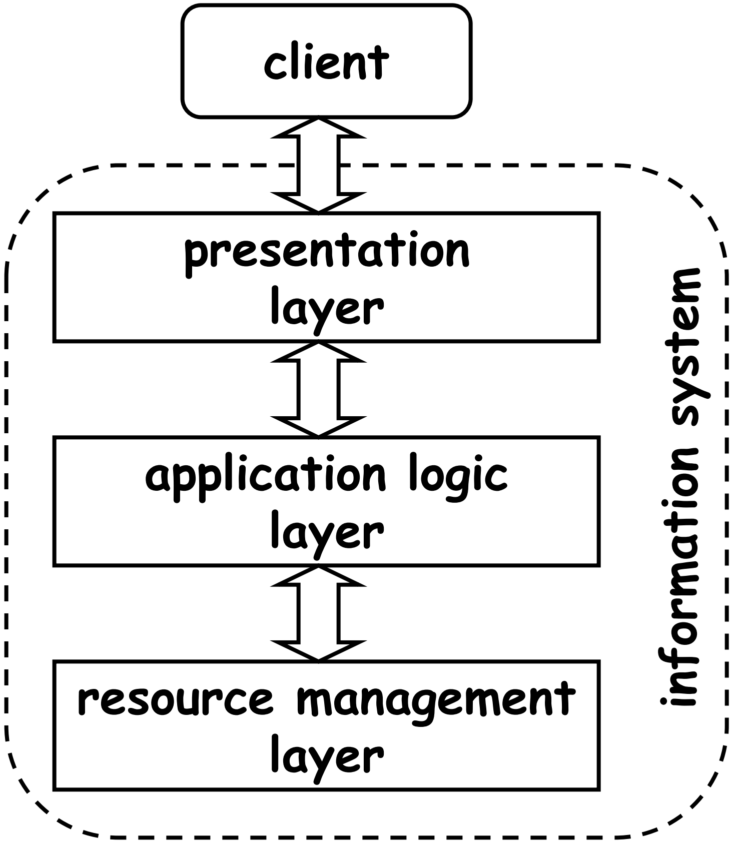
Sharing of application data

integration challenge

- heterogeneous platforms
- heterogeneous communication protocols
- corporate islands of automation (COTS)
- various data syntax
- diverse application semantics
- middleware revolution: various technologies available

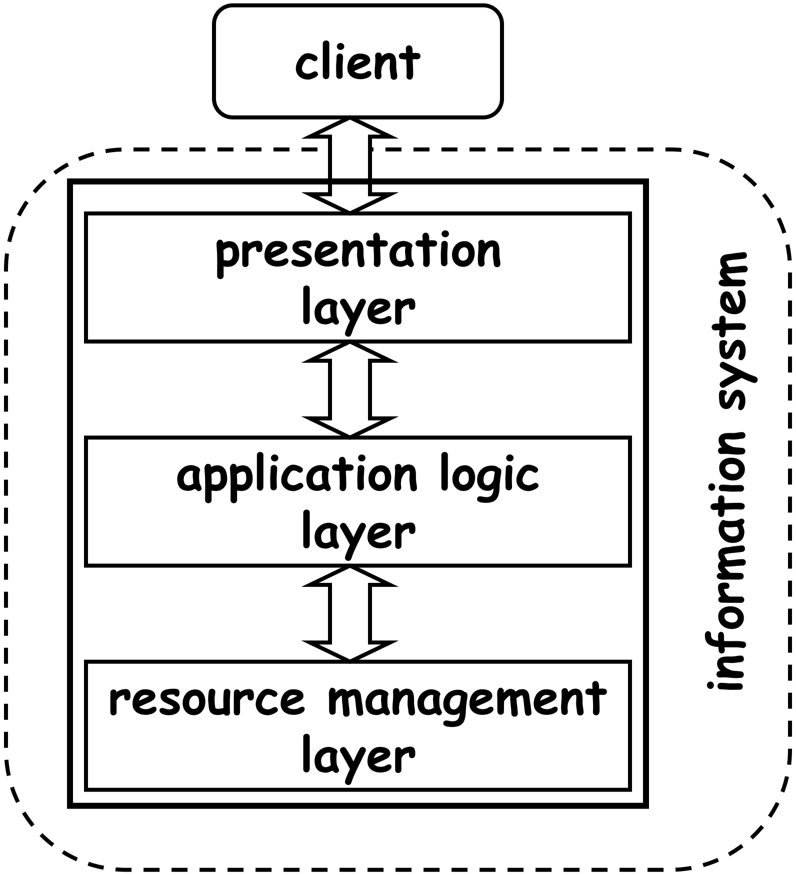
modern enterprise information systems

- Information systems are now designed around 3 layers: presentation, application logic, and resource management
- **presentation layer**: presenting information to the external entities and allowing those to interact with the system by submitting operations and getting responses
  - e.g., Web server modules for HTML creation
- **application logic layer**: services that implements actual operation requested by the client
  - business logic, business rules
- **resource management layer**: deals with and implements the different data sources of an information system
  - e.g., RDBMS, ERP, Legacy systems



one-tier architectures

- direct result of the computer architectures used several decades ago (i.e., mainframe-based)
- presentation, AL, and RM layers were merged into a single tier (i.e., in the single mainframe)
- many such systems are still in use today
- 1-tier systems do not provide any entry point from the outside except the channel to the dumb terminals (i.e., no APIs) -> needs screen scraping
- advantages: extremely efficient
- drawback: monolithic piece of code -> hard to maintain

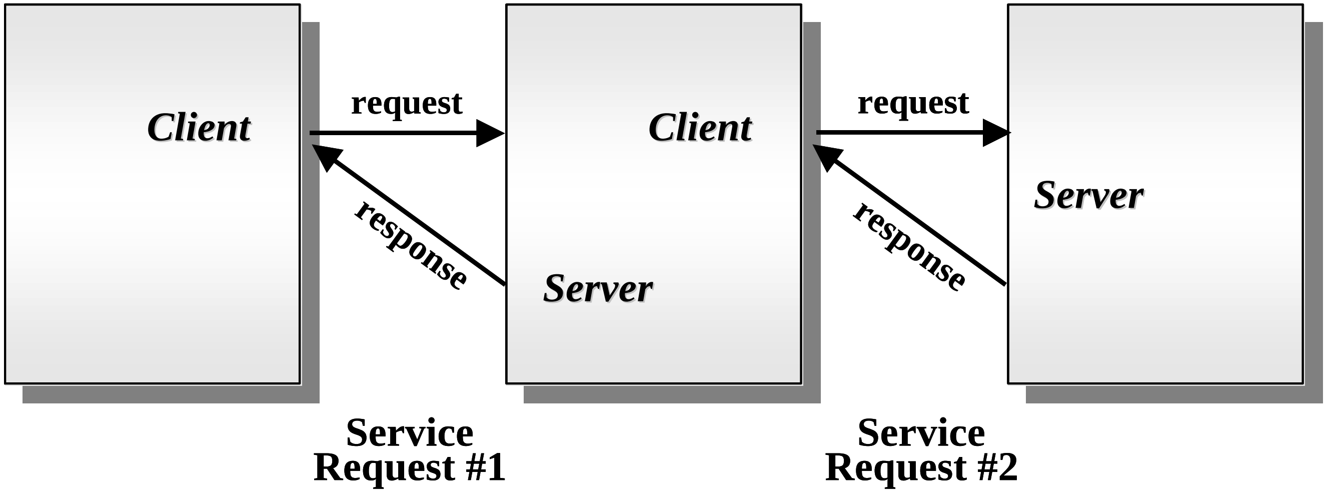


emergence of client-server computing

- distributed computing is the classical paradigm in support of e-business processes & applications.
- client server computing
  - an architecture that involves client processes (service consumers) requesting service from server processes (service providers)
  - handles the need for both centralized data control and widespread data accessibility
  - does not emphasize hardware distinctions; it rather focuses on the applications themselves
  - provides a typical way to interconnect programs that are distributed across different locations

configurations of CS computing

- there are different ways in which processing tasks can be divided between the client and the server
  - thin clients, with heavy servers
  - servers that only contain common data with all the processing executed at the level of the client
- solutions chosen depend on specific application requirements, e.g., local vs. central control, # of users, processing needs etc.



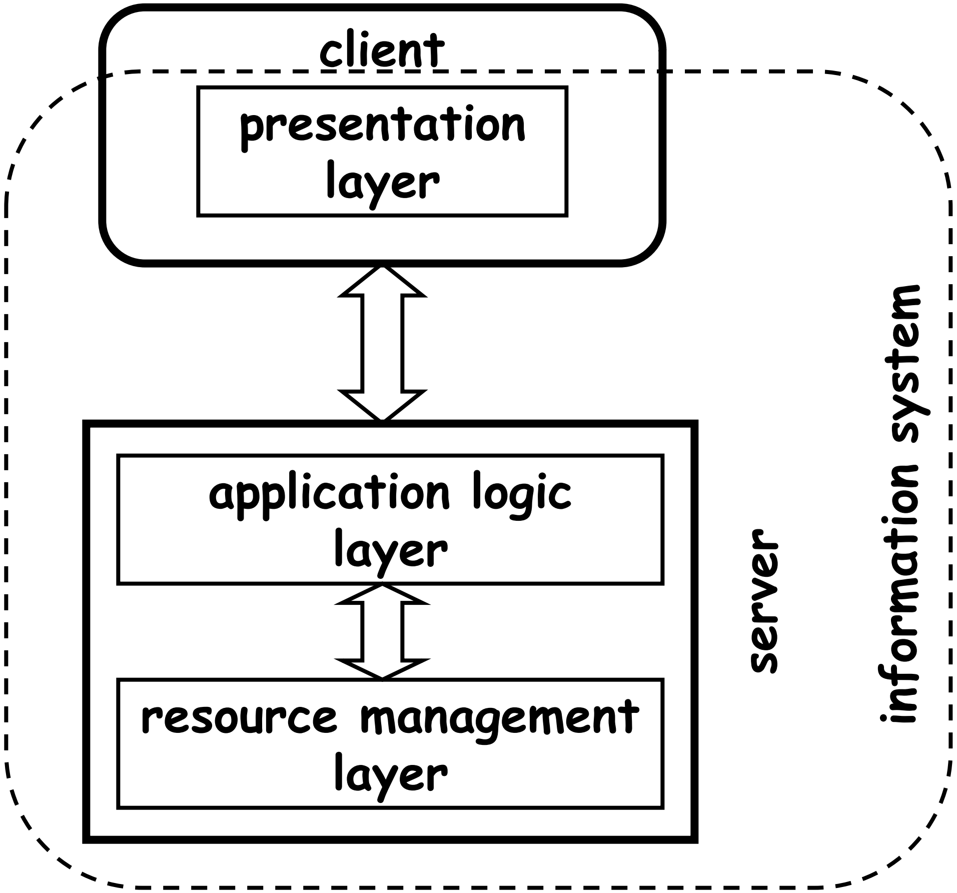
Client/Server architecture

features of client/server Architecture

- basic features of the client/server model
  - clients and servers are functional modules with well defined interfaces
  - functions of a client and a server can be implemented by a set of software modules, hardware components, or any combination thereof
  - each client/server relationship is established between two functional modules, where one module, the client, initiates service requests and the other module, the server, responds to these requests
  - information exchange between clients and servers, i.e., requests and responses, are strictly through messages.
  - message exchange is typically interactive.
  - clients and servers may run on separate dedicated machines connected through a network

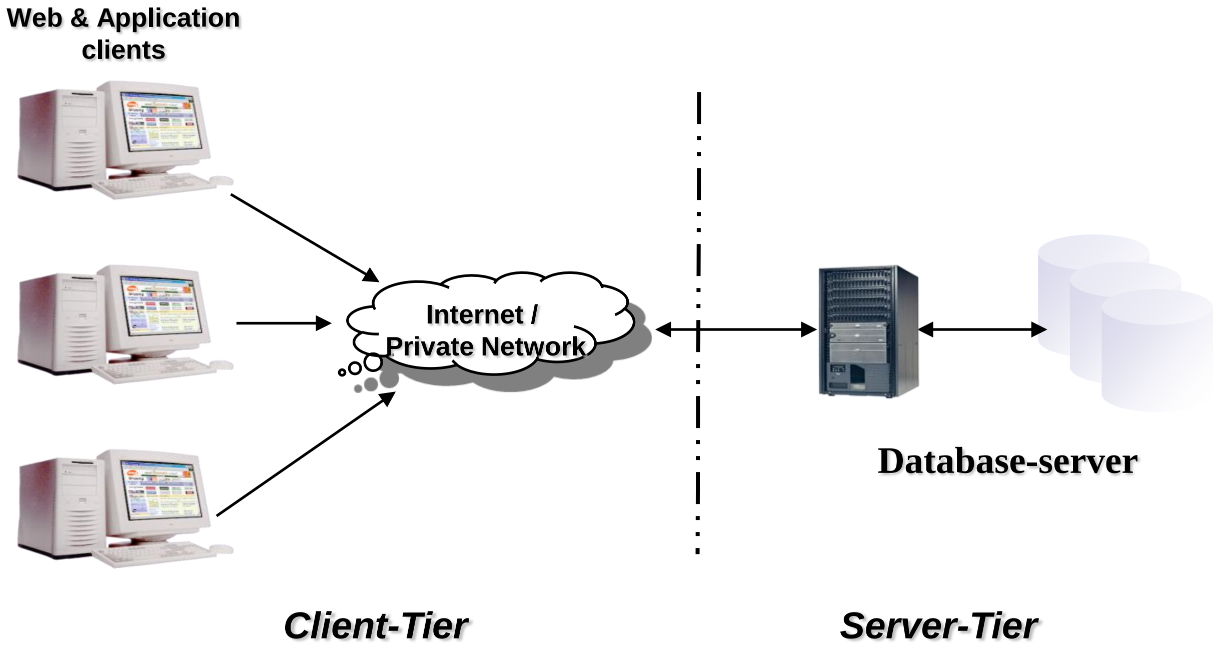
two-tier architectures

- driven by the emergence of the PC
- presentation layer can utilize the computational power of a PC
- it becomes possible to tailor the presentation layer for different purposes
- resulted in the development of API that specifies how to invoke a service, the responses that can be expected, and what effects the invocation will have on the internal state of the server



two-tier client/server architecture

- tiers into which an application is partitioned is known as the **logical partitioning** of an application as opposed to **physical partitioning** (# of platforms where the application executables are deployed)
- client: GUI, business application logic, and rules
- server: DB access

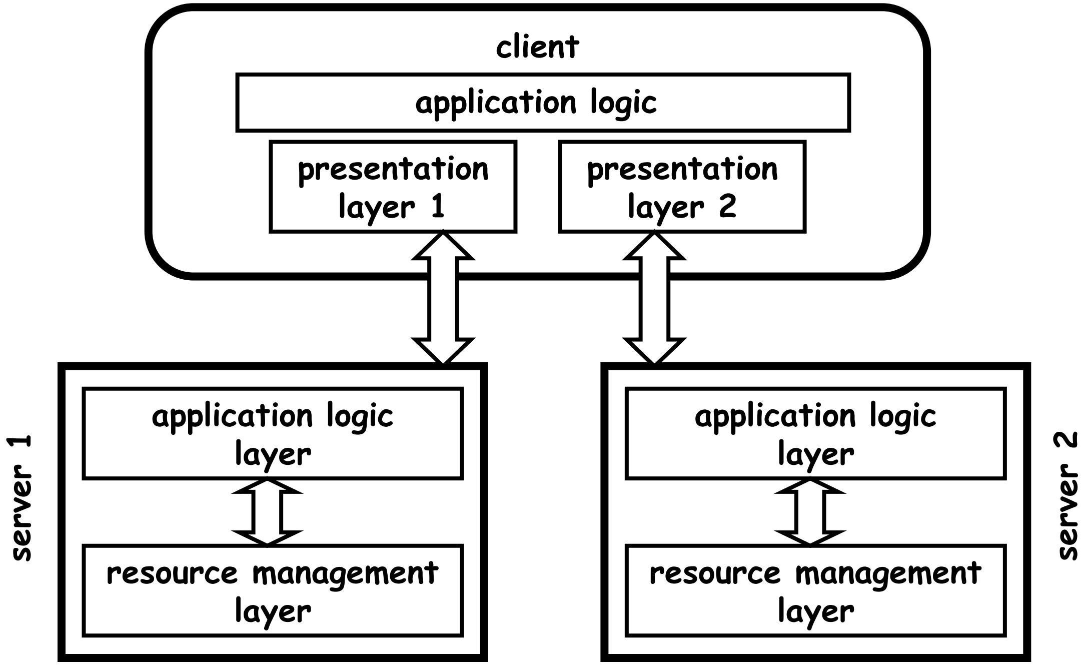


drawbacks of the two-tier CS architecture

- client in such a two-tier system is known as “fat client”
- conversations occur at the level of the server’s database language.
- two-tier architecture has several drawbacks, which are especially problematic for large and distributed applications
  - scalability problems of server
  - poor business logic sharing: logic is embedded in the UI code
  - client reliance on the database structure: hard to change the DB
  - limited interoperability: difficulty of integrating with another DBMS
  - high-maintenance costs: update of client applications

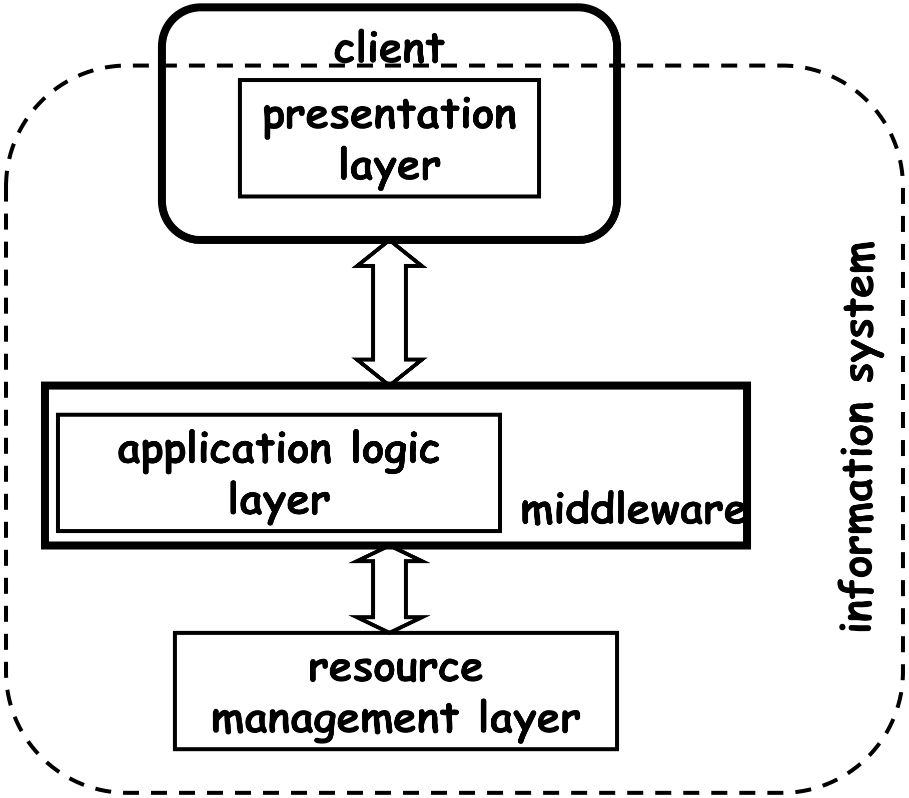
legacy problem in two-tier architecture

- individual programs responsible for the AL became services running on a server
- advantages: fast execution of key operations, portability across different platforms
- problems:
  - a single server can only support a limited # of clients
  - **legacy problem**: for the client that need to connect to multiple servers, any changes made to a server requires a change to the client



three-tier architectures

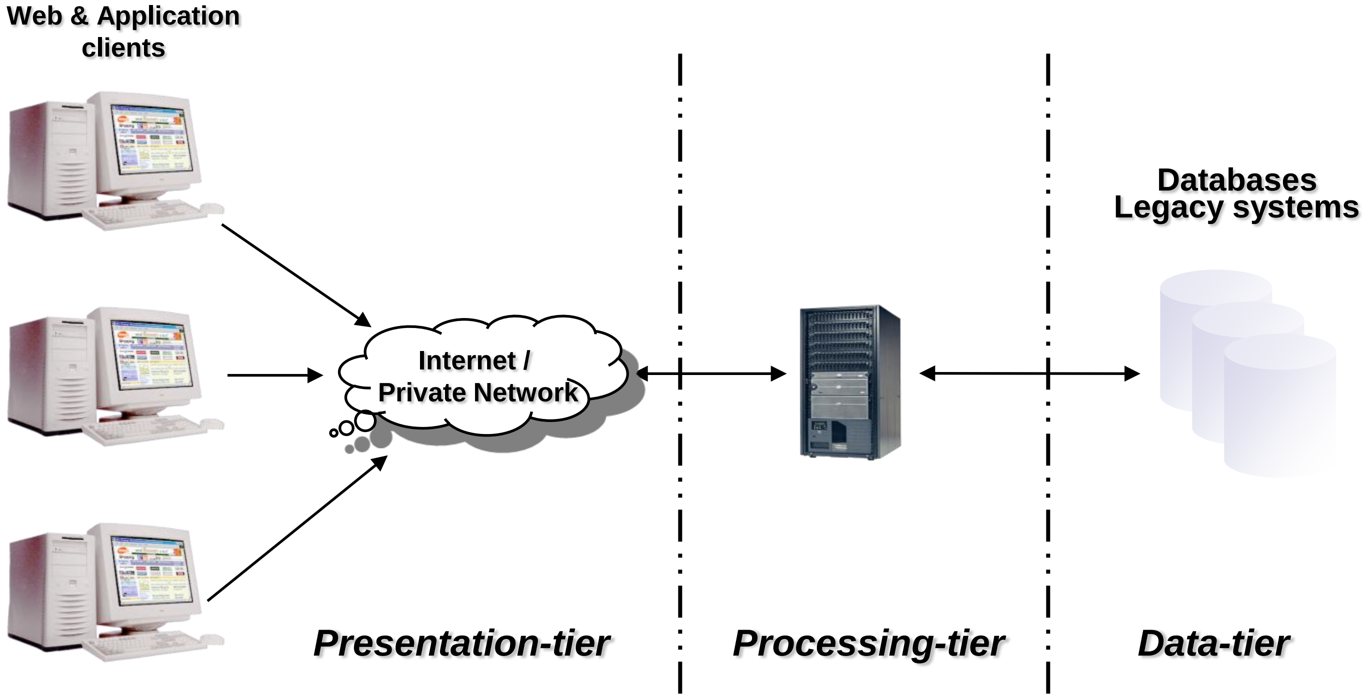
- proliferation of 2-tier systems created islands of information where a set of clients could communicate with a server but **could not communicate with other servers**
- introduces an additional layer between the clients and the servers to support the integration of the underlying systems
- abstractions and infrastructure that support the development of the AL are collectively known as middleware



3 logical tiers

- presentation tier
  - responsible for the graphical user interface (GUI) layer usually in the form of a web-browser
- processing tier (or middle-tier)
  - contains the business logic & is responsible for the processing associated applications supported
  - enables developers to isolate the main part of an application that can change over time
  - has the effect of logically and physically decoupling business logic from the presentation and database functions
  - a variety of ways of implementation: transaction processing monitors, message servers, or application servers
- data tier
  - holds the permanent data associated with the applications supported
  - interprets requests from a client and routes them to a suitable data resource
  - e.g., modern and legacy application databases, and transaction management applications

Three-tier client/server architecture

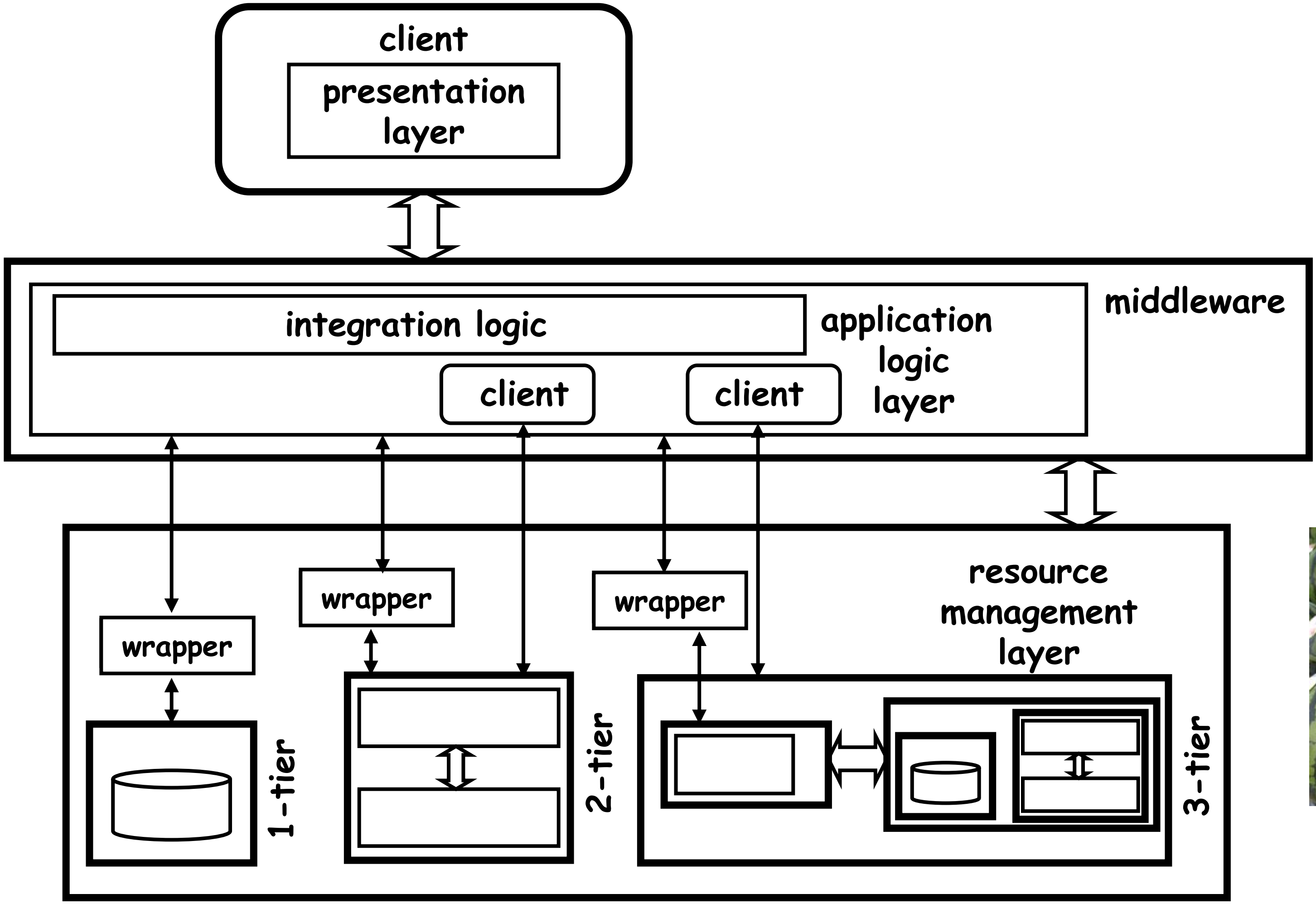


more on 3-tier architectures

- scalability over 2-tier system can be accomplished by running each layer in a different server
- also increased interoperability (w.r.t. new applications), flexibility, maintainability, reusability, security
- disadvantage is that the **communication between the RM layer and the AL layer** becomes much more expensive
- resource managers were forced to provide clear interfaces so that they could be accessed by AL running at the middleware layer -> introduction of ODBC, JDBC
- that is, **2-tier architectures forced the definition of AL layer APIs, and 3-tier architectures forced the creation of RM layer APIs**
- the legacy problem still exists

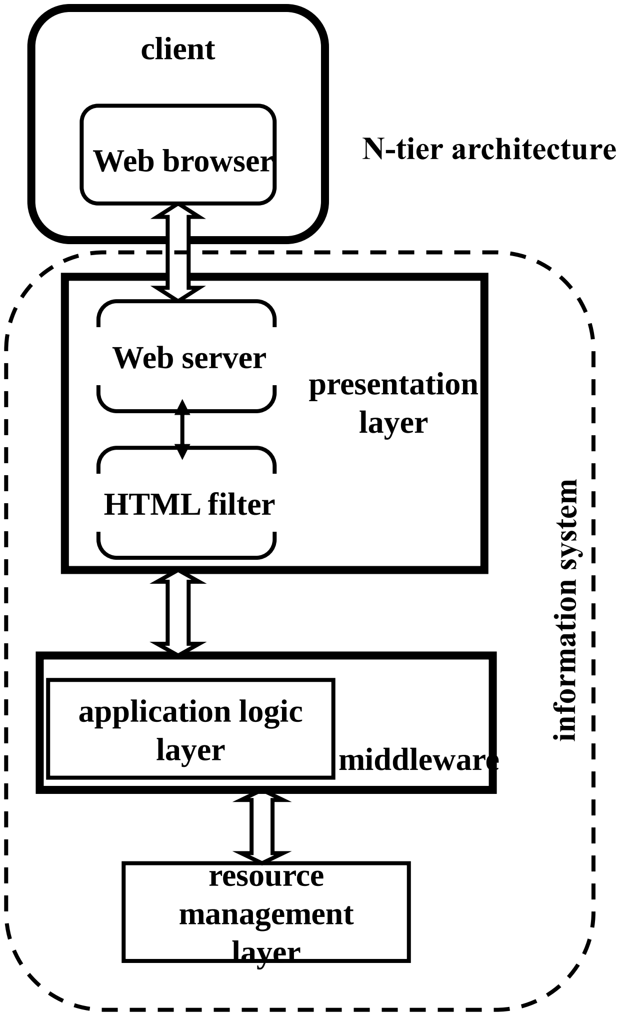
deployment of 3-tier architectures

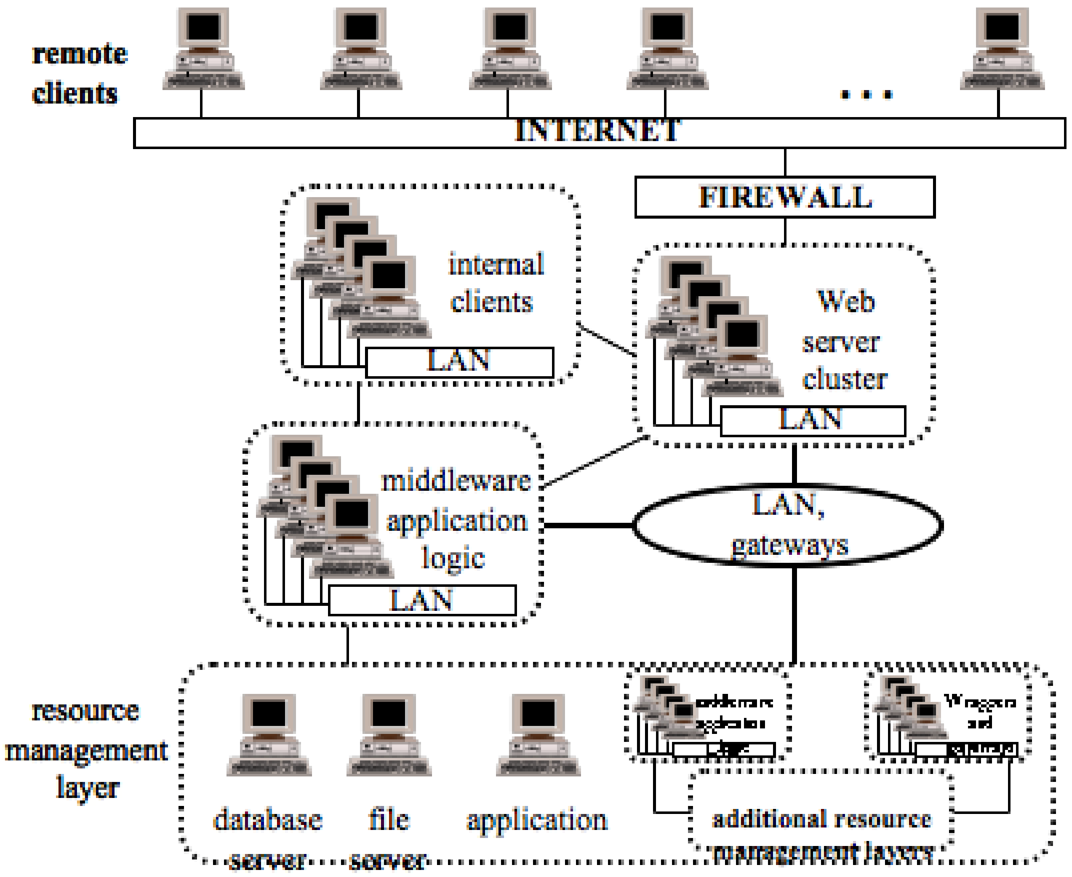
- integration of systems with different architectures using a 3-tier approach



N-tier architectures

- appears in two generic settings: linking of different systems and adding connectivity to the Internet





web technologies and applications

- WWW: a vast information system consisting of software applications or processes that exchange info & that act on behalf of a user or another application
- WWW architecture consists of
  - identifiers
    - a single specification to identify objects in the system
    - the Uniform Resource Identifier (URI)
  - formats
    - a nonexclusive set of data format specifications designed for interchange between agents in the system
    - e.g., HTML, XML schemas, etc.
  - protocols
    - a small and non-exclusive set of protocol specifications for interchanging information between agents
    - e.g., HTTP, SMTP

resources and URLs

- web is a **universe of resources** (anything that has identity)
  - examples: documents, files, menu items, machines, and services, as well as people, organizations, & concepts
- URI (uniform resource identifier) helps locate WWW resources
  - a URI consists of a string of characters that uniquely identifies a network resources
  - URIs include URLs, which use traditional addressing schemes such as http and ftp, and Uniform Resource Names (URNs)
- HTML (hypertext markup language): comprises elements and tags
  - organises inter-linked pages of information residing on sites throughout the world
  - web pages rely on markup languages to tag text files for display at web browsers
- HTTP (hypertext transfer protocol)
  - uses an simple request/response model that establishes connection with web server specified in the URL, requests a specific web page, retrieves the needed content, and closes the connection

web-based applications

- web sites provide the content that is accessed by web users
- a web site consists of
  - web server: an application (technically a server process) that receives calls from web clients and retrieves web pages and/or receives information from gateways
  - content files (web pages)
  - gateways (programs that access non-Web content, e.g., databases).
- web browsers
  - clients that typically use graphical user interfaces to wander through the web sites
  - IE, firefox, ...

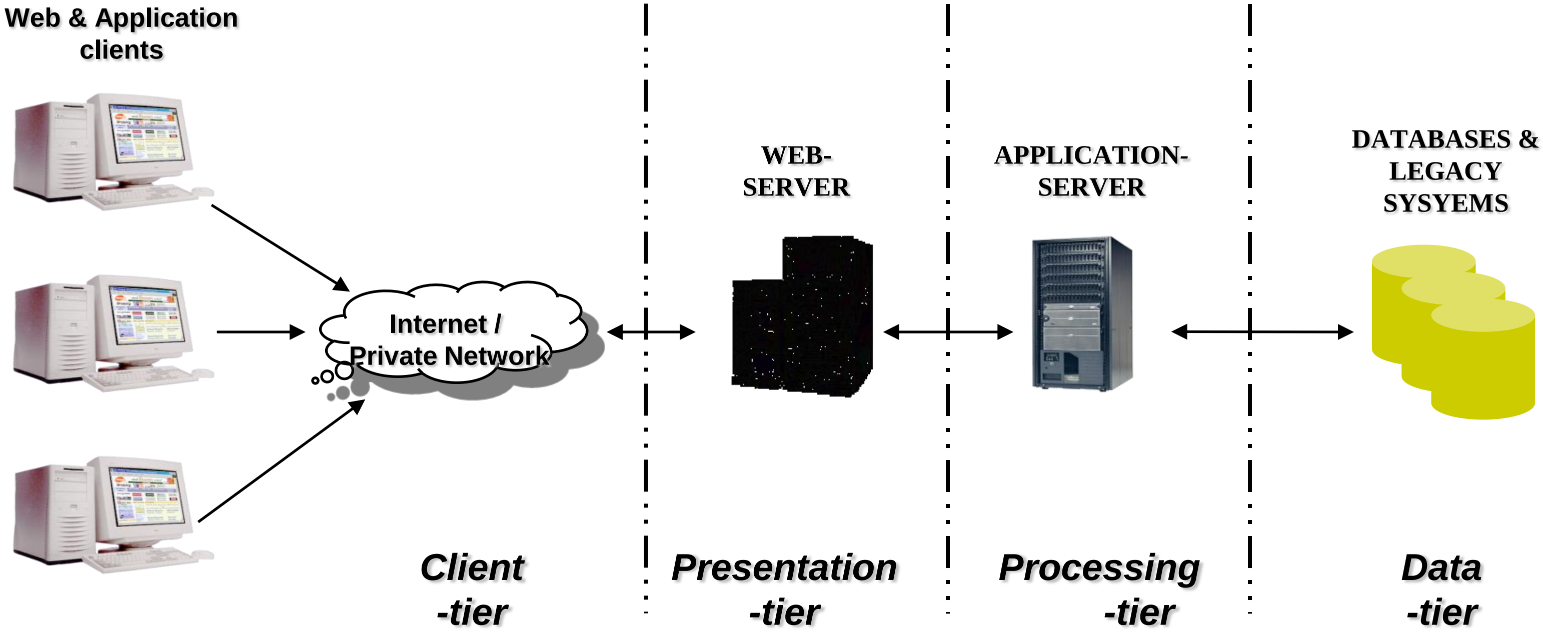
major elements in web application

- **web clients** through which users communicate with web application servers
- **web application servers** (WAS)
  - administer the entire info content intended for publication on the Web
  - dispense files that contain web pages, images, sound and video clips and other media
- **infrastructure services**
  - e.g., caching, directory & security
- **external services** (non-web)
  - mission critical applications & data internal to an enterprise
  - external partner services, e.g., financial, payment, information services, etc.

types of web applications

- static documents
  - delivered from the file system of the web server
  - can be read from an existing file
  - e.g., HTML
- dynamic documents
  - have an interactive and usually time-sensitive nature (e.g., “hello, jessica” in amazon.com)
  - require the server to generate the document on the fly
  - may not exist on a disk at all: e.g., can be generated from databases, video capture systems and from scientific instruments such as whether monitoring systems
  - e.g., JSP, PHP
- cf: active documents

multi-tiered architecture for web-based applications



architectural features of web-based Applications

- web-enabled applications: a special case of client-server applications where the client is a **web browser**
- web-enabled applications leverage the three-tier architecture
  - AS just became too big -> presentation tier of the three-tier architecture is subdivided into a client-tier and a new presentation-tier
  - needs of this new presentation-tier is addressed by an infrastructure known as a web server (e.g., tomcat)
- presentation-tier (web server)
  - receives requests from client apps & generates HTML using the services provided by the business (processing) tier
  - provides further isolation between the application layout and the application logic

more on features of web-based applications

- **client tier**
  - implemented as a web browser running on the user's client machine
  - displays data & lets users & client applications enter/update data
- **presentation tier**
  - generates web pages in which it includes dynamic content
  - a web-server finds the client application or user-entered data in web pages coming back from the client & forwards it to the business logic-tier
  - e.g., CGI, ASP.NET, JSP, ...
- **processing tier**
  - includes application logic for performing all required calculations and validations, managing workflow & all data access for the presentation-tier
  - AS provides middleware services for security, state maintenance, transaction management, DB connection pooling, ... (ususally following J2EE)
  - examples of ASs: IBM WebSphere, MS MTS, Oracle AS, JBoss, ...
- **data tier**
  - provides the business logic-tier with required data when needed & store data when requested
  - includes access procedures to ERP, legacy systems, ...

client-side programming

- applets
  - snippets of java code that are firmly anchored in a web document and run on the client side
  - can be sent with a web page to a user
- JavaScript
  - can be directly embedded in the HTML code
  - interacts with browser
  - uses DHTML to interact with elements that make up a web page
  - cf. AJAX

server-side programming

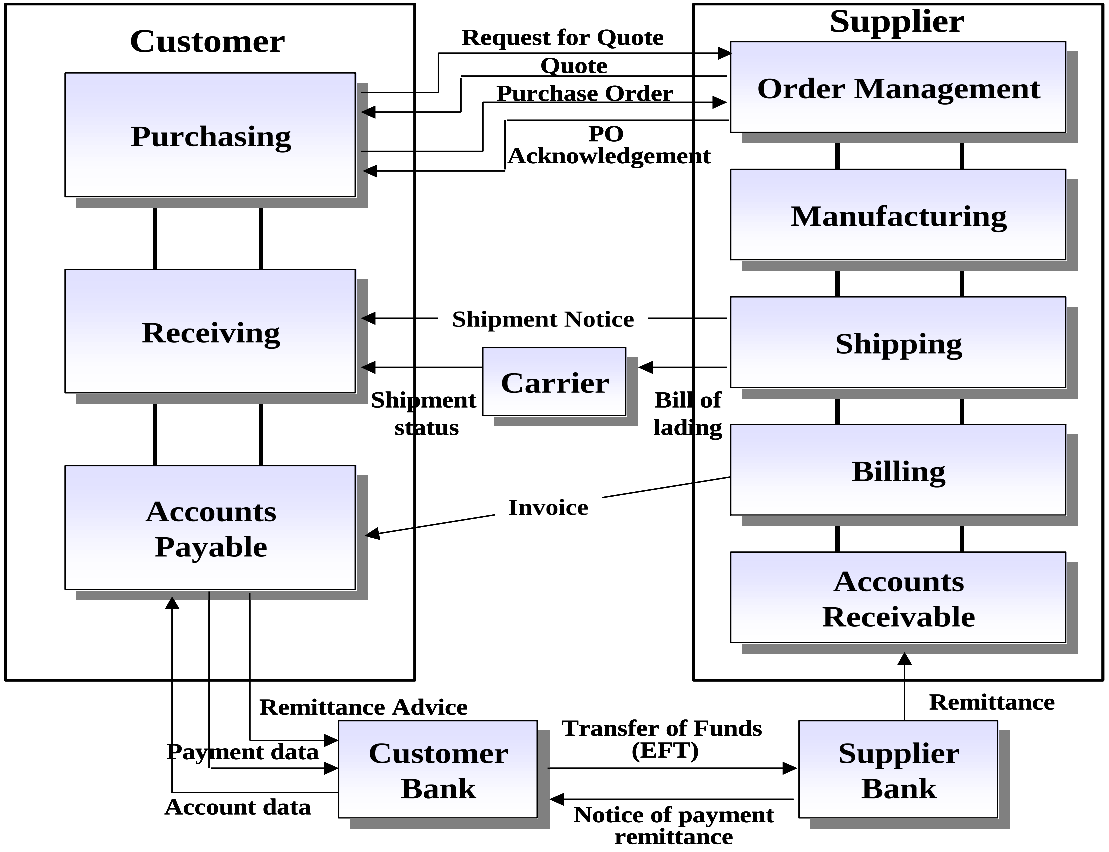
- servlets
  - a java component that can be plugged into a java-enabled web server to provide server extensions in the form of customized services
  - **java code that runs in a server application** to answer client requests
  - developers have access to the **full range of java APIs**
  - typical uses
    - **processing and storing data** submitted by an HTML form
    - providing **dynamic content** (e.g., DB query results)
    - **managing state** information on top of HTTP (e.g., shopping cart)
  - servlet containers (usually included in WAS): apache tomcat, sun one, ...
- JSP
  - an extension of servlet technology to support authoring of HTML and XML pages
  - allows **java code to be embedded** into web pages to carry out the display of information dynamically on as and when needed basis
  - can be viewed as a **template for the HTML result page**, with “holes” for the dynamic data that may vary on each request

EDI: infrastructure for e-Business

- objective of collaborative technologies: to **eliminate the manual trading processes** by allowing internal applications of different companies to directly exchange information and documents
- electronic data interchange (EDI)
  - transfer of **structured data** by **agreed message standards** between computer applications
  - **transaction sets** define a set of fields, the order and length of fields along with business rules that accompany fields
  - a fast, cost-effective, and safe method of sending POs, invoices, shipping notices, ...
  - each participant needs to **map its docs to EDI doc standards**
  - in use since the 1970s on private VANs
  - still accounts for the bulk of e-business transactions
  - standards maintained by UN/EDIFACT, ANSI X12

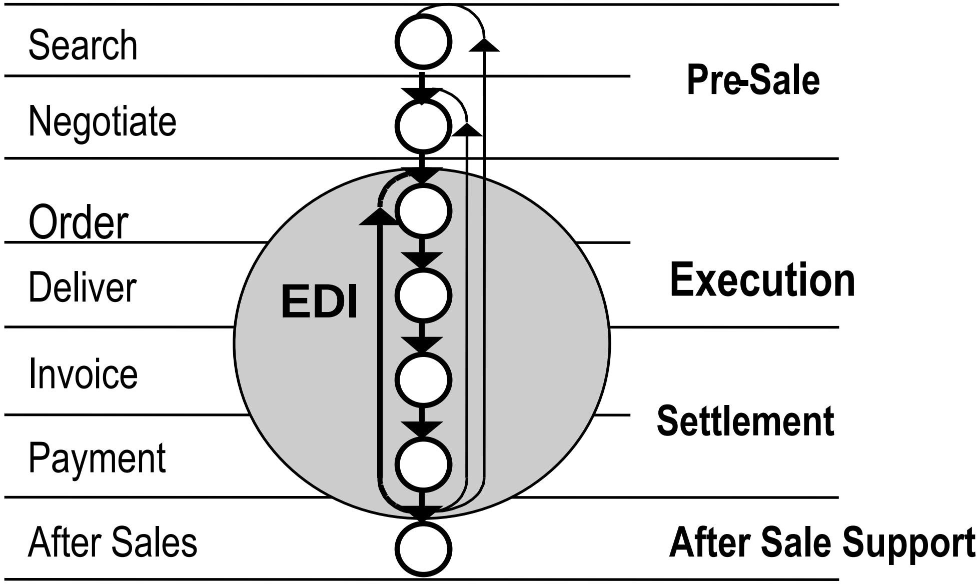
EDI

- two key elements
  - electronic documents replace their paper counterparts
  - exchange of documents takes place in a standardized format



EDI trade cycle

- regular, repeat transactions between commercial trading partners
- examples:
  - supermarkets replenishing stocks
  - vehicle assemblers purchasing components



problems with EDI

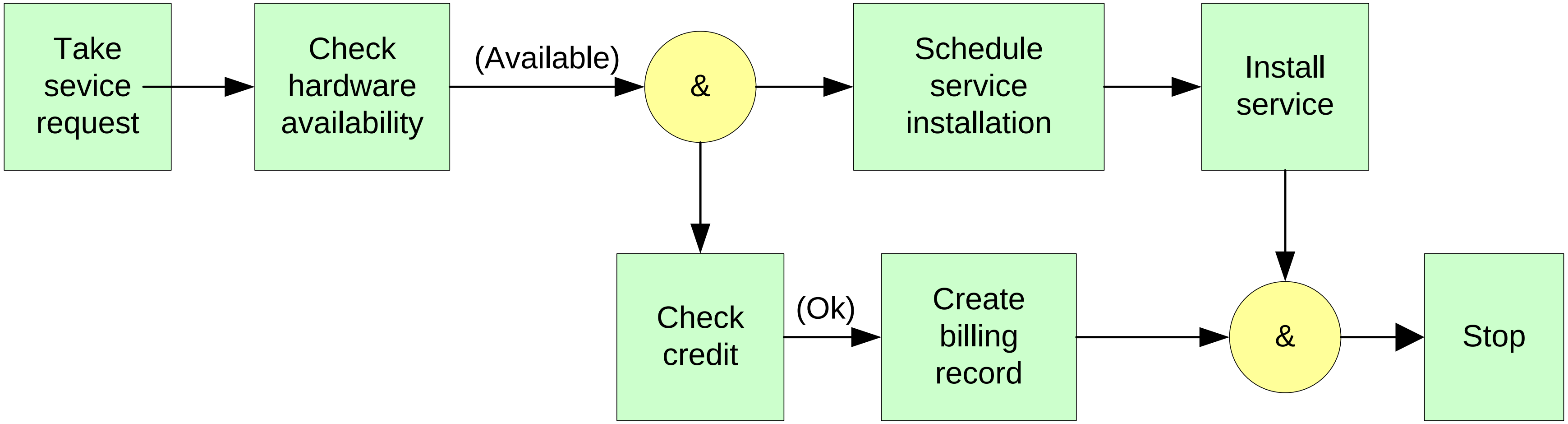
- fixed transaction sets
  - difficult to deal with the normal **evolution** necessary for companies introducing new products, services, or evolving computing infrastructures, or improving BPs
- resilience to change
  - standards bodies are slow
- reliance on **proprietary communications networks**
  - expensive, incompatible
- encapsulation of business rules in transaction sets
  - business processes and rules are mixed with documents

collaborative technologies: workflows

- a workflow system
  - **automates** a business process, in whole or in part, during which **documents**, **information**, or **tasks** are passed from one participant to another for action, according to a set of rules
  - responsible for definition, creation, and management of the workflow execution
  - increasingly migrating from departmental server-based architectures to EAI, espousing the business process modeling paradigm of BPM
  - generally supports well-defined, static, clerical processes
  - provides transparency
- a **workflow**
  - normally comprises a number of logical steps (activities)
  - an **activity** may involve manual interaction with a **user**, or might be executed using diverse resources such as **application programs** or **DBs**
  - a **work item** is created, processed, and changed in stages at a number of processing or decision points
  - depict various aspects of a business process including automated and manual activities, decision points and business rules, parallel and sequential work routes, and how to manage exceptions

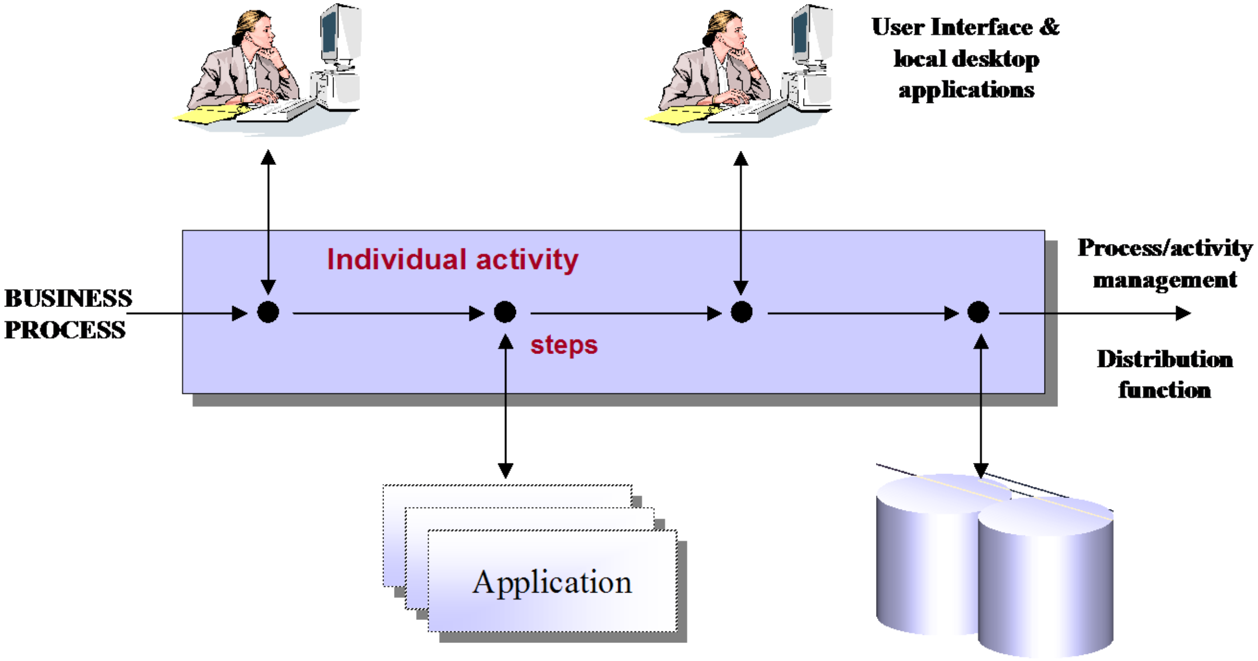
an example workflow

- control and data flows are directly specified from a logically **central** perspective



process flow across applications

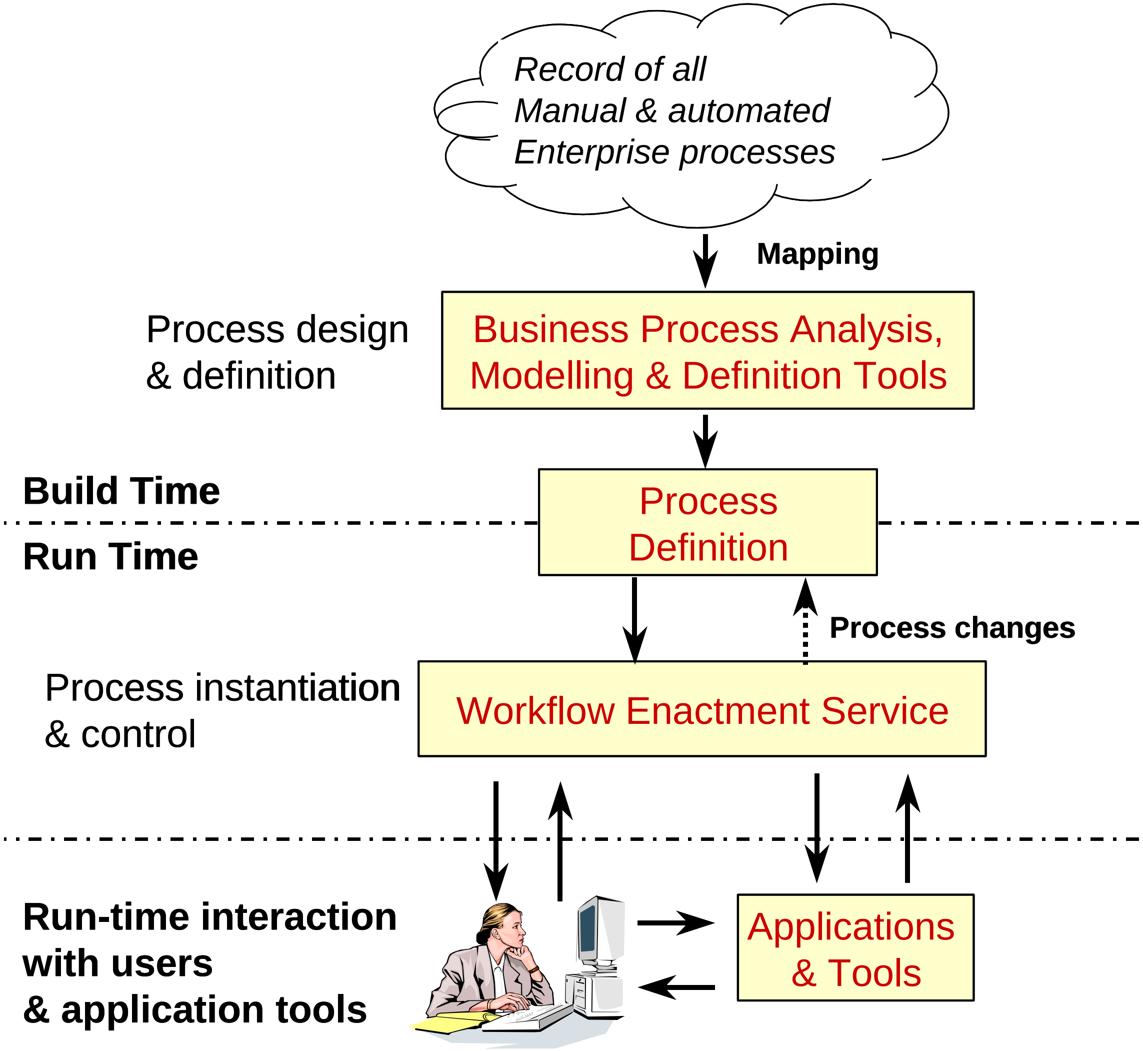
- workflow technology enables developers to describe full intra- or inter-organisational business processes with **dependencies**, **sequencing selection** and **iteration**
- it enables the developers to describe the complex **rules** for processing in a business process



workflow system characteristics

- each workflow management system can be viewed at the highest level as providing support for the three functional areas
    - **build time** functions: these are concerned with defining, modeling and analyzing workflow processes and related activities
    - **run-time** process **control** functions: these are concerned with managing the sequencing and execution of workflow processes
    - **run-time interactions**: these are concerned with supporting interactions with human users and applications and tools for processing activity steps in workflow processes

architectural characteristics of WfMS


Record of all
Manual & automated
Enterprise processes

Mapping

Process design
& definition

Business Process Analysis,
Modelling & Definition Tools

Build Time

Run Time

Process
Definition

Process changes

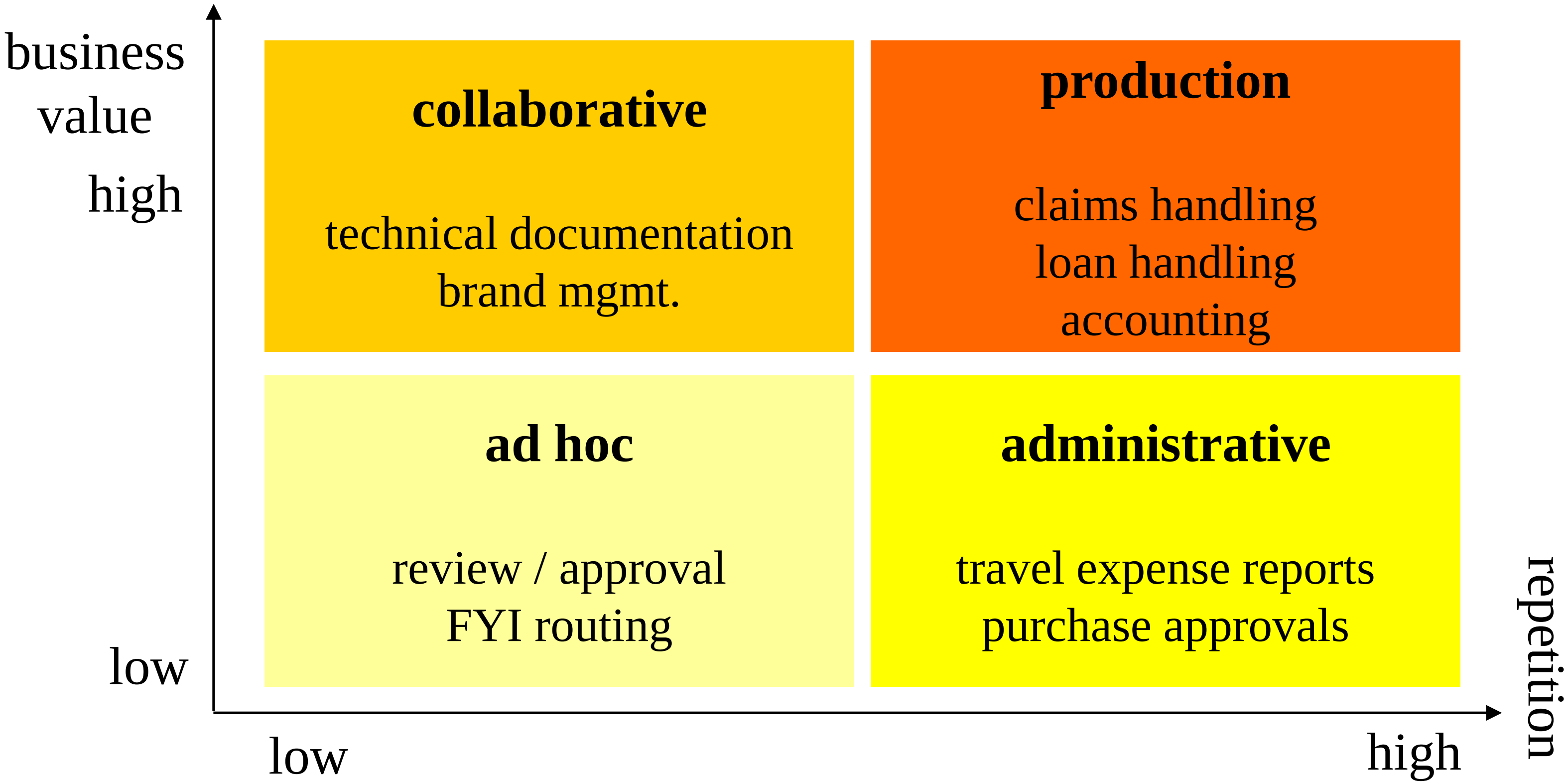
Process instantiation
& control

Workflow Enactment Service

Run-time interaction
with users
& application tools

Applications
& Tools

types of workflows



business value

high

collaborative

technical documentation
brand mgmt.

production

claims handling
loan handling
accounting

low

ad hoc

review / approval
FYI routing

administrative

travel expense reports
purchase approvals

low

high

repetition

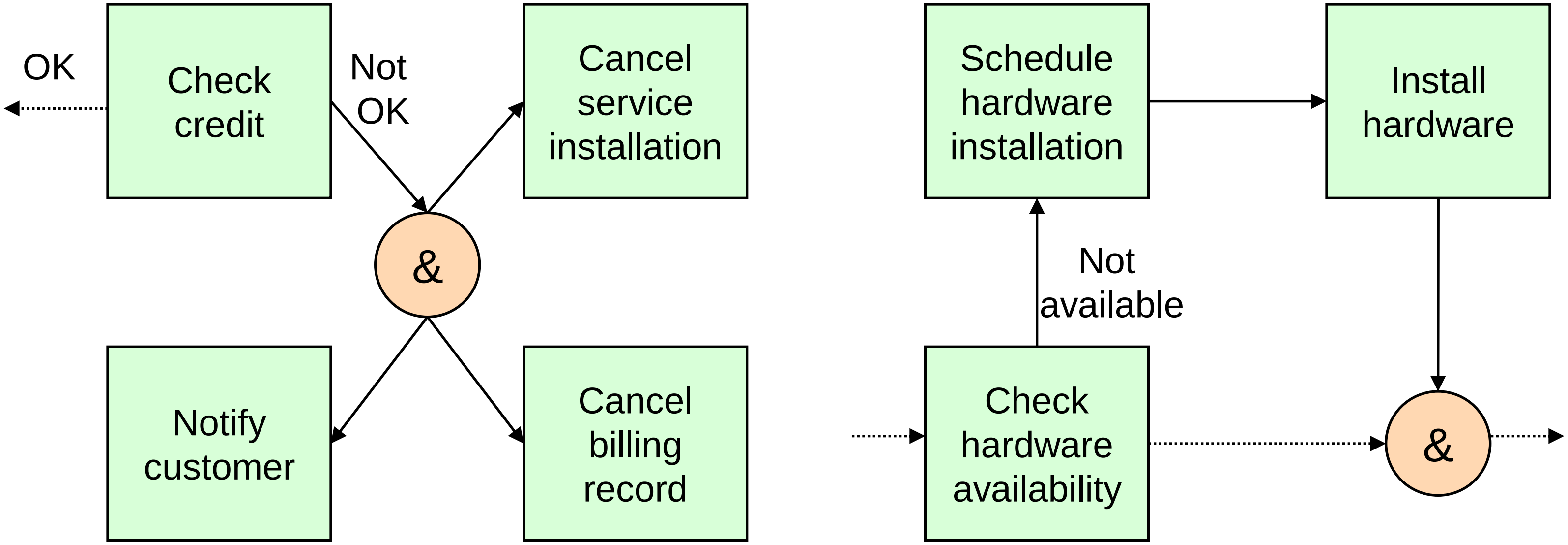
- business value defines the importance of a workflow to the company’s business
- repetition measures how often a particular process is performed in the same manner

types of workflows (cont.)

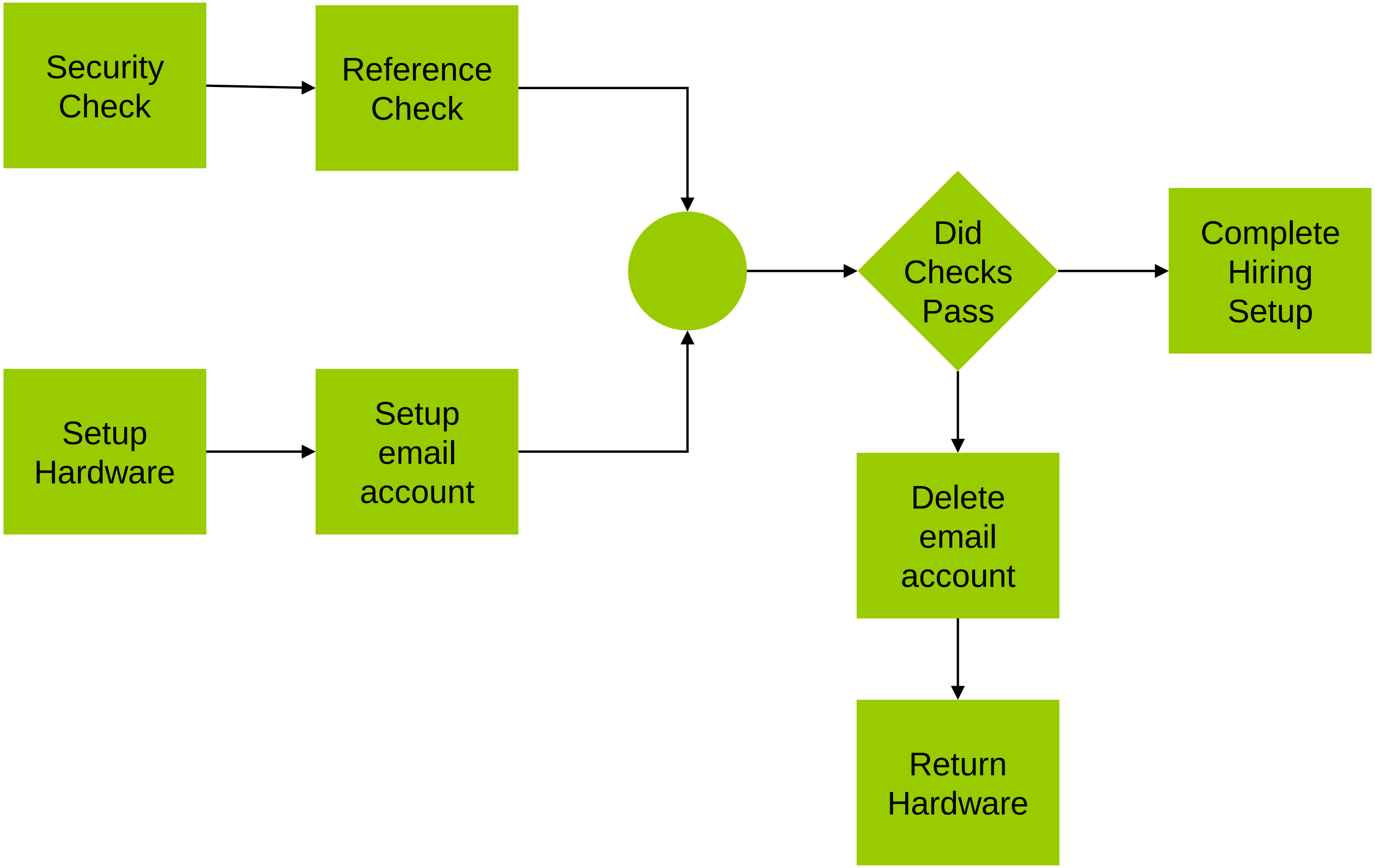
- production workflow
  - make up the traditional part of the workflow market
  - goal: manage large numbers of similar core tasks and optimize productivity
  - can be completely pre-defined and prioritized
- administrative workflow
  - should able to ease the process definition
  - process definitions are ususally created using forms
- ad-hoc workflow
  - allow users to create and amend processes
  - have no predefined structure, each BP is constructed individually
  - usually built on top of an e-mail platform
- collaborative workflow
  - focuses on teams of people working together towards common goals
  - e.g., designing and building an automobile, creating technical docs for a software product
  - usually supported by groupware

exceptions in a workflow

- possible exceptions in the previous workflow example
  - customer has an unpaid and overdue balance
  - the service installer for your area calls in sick
  - the available hardware is unusable
- practically **impossible to specify all exceptions statically and in advance**
  - exceptions are not just alternative flows of control!
  - if some exceptions occur often enough to become almost routine, they will be incorporated as explicit alternatives within the workflow



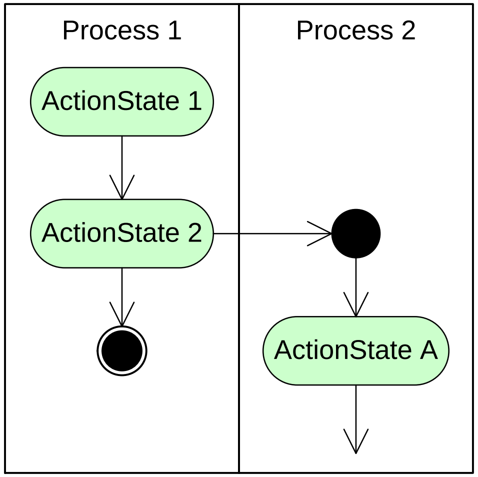
an example of compensating transactions



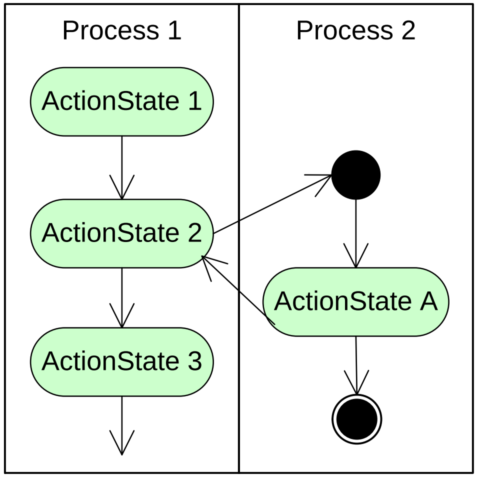
workflow interoperability

- multiple workflows can arise and interact with each other
- workflows interact by sharing data or functionality
- an **interaction** can occur
  - directly
  - via message passing
  - through a gateway that translates protocols
  - by mutual use of a common repository
- 3 primitive patterns for the interoperability

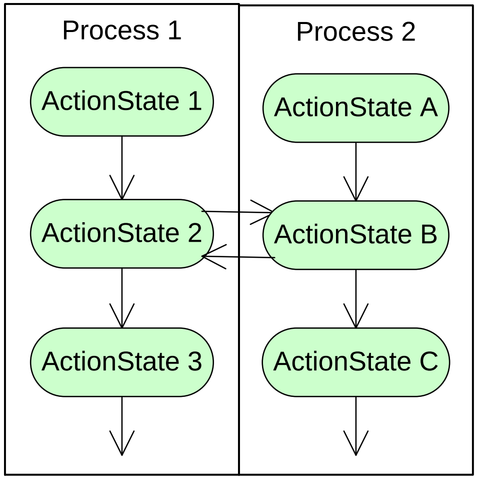
Chained Interoperability



Nested Interoperability



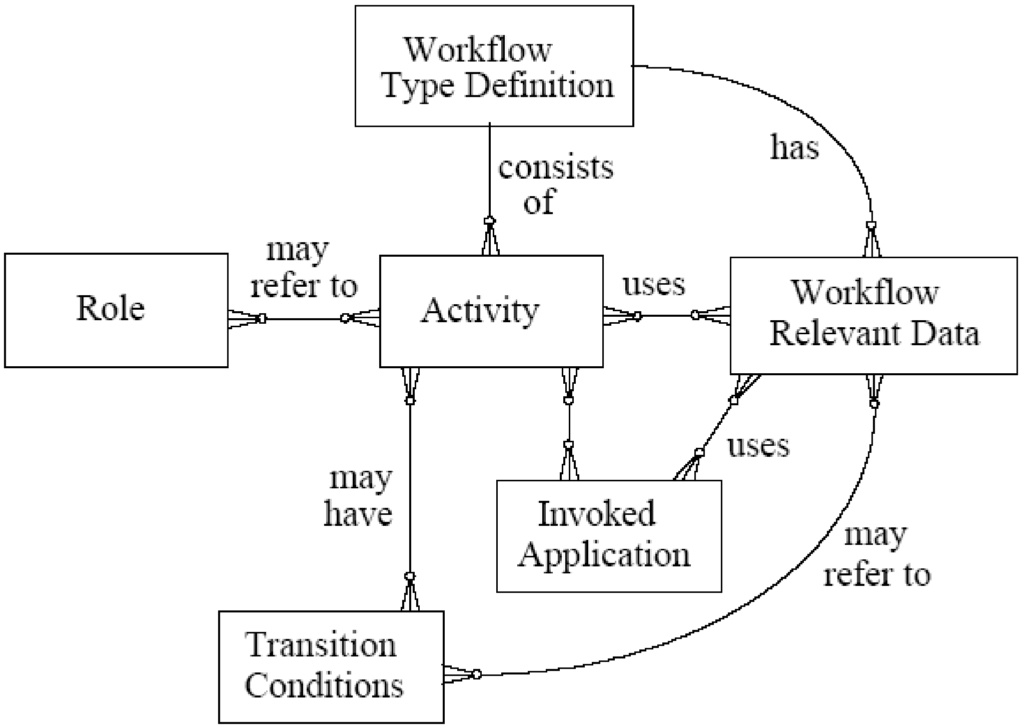
Synchronized Interoperability



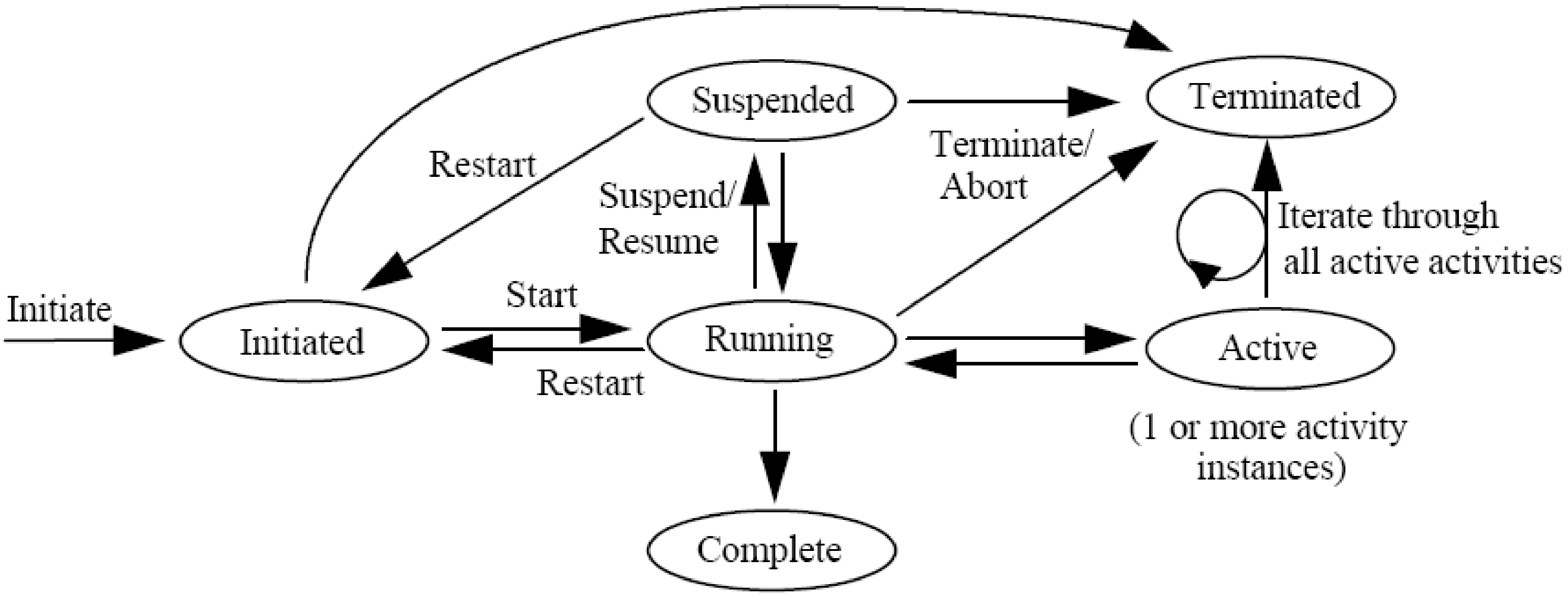
a metamodel for workflow (WfMC)

- process
  - a collection of tasks organized into a graph
- task
  - an atomic work item
- service
  - implements a task and may be implemented
- actor
  - a human or machine that performs a task by fulfilling a service
- role
  - abstracts a set of tasks
- workflow
  - an instance of a process that binds and consumes resources in fulfilling the tasks of a process

Workflow Reference Model



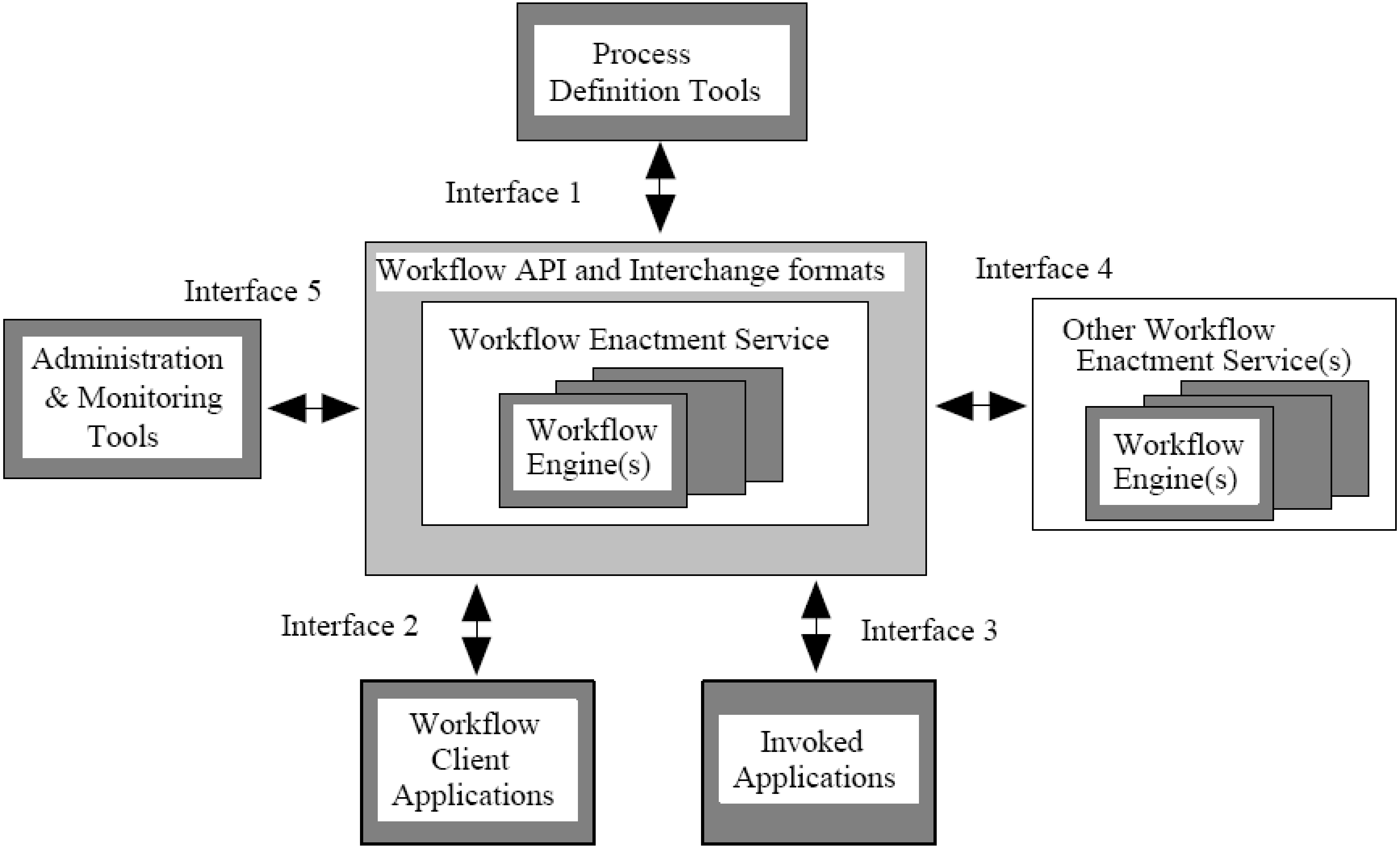
a state-transition diagram for a workflow



workflow interoperation

- a workflow represents the interoperation of several applications and databases
- **multiple workflow units interact with each other**, because some people participate in more than one, and the units inevitably share resources
- a reference model for workflow management (WfMC)
    - describes how workflow engines ought to be connected to applications, databases, development tools, runtime tools, and each other
- SWAP (Simple Workflow Access Protocol)
    - enables workflows modeled and managed by tools from different vendors to be related
    - governs both the control and monitoring of workflows
    - protocol for basic interaction
        - the client invokes createProcessInstance command on the workflow server
        - the server returns the URI of the workflow instance
        - the client sends its own URI to the instance
        - when it is done, the workflow instance invokes the completed command on the client

a reference model for WFMS (WfMC)


Process Definition Tools
Interface 1
Workflow API and Interchange formats
Workflow Enactment Service
Workflow Engine(s)
Interface 5
Administration & Monitoring Tools
Interface 4
Other Workflow Enactment Service(s)
Workflow Engine(s)
Interface 2
Interface 3
Workflow Client Applications
Invoked Applications

workflow management systems

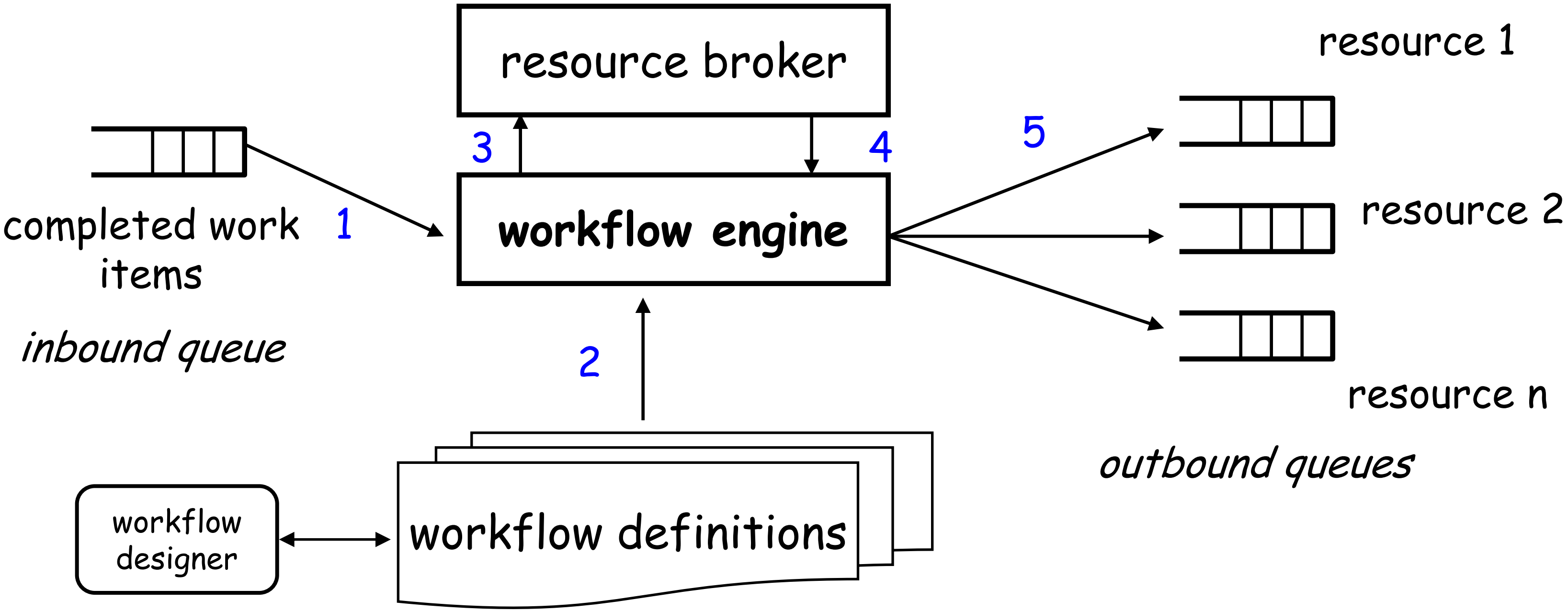
- WfMS
  - a software platform that supports the design, development, execution, and analysis of workflow processes
  - facilitates the definition and maintenance of the integration logic
  - originally aims at automating administrative processes based on paper documents
- production workflows
  - WfMS is used not only to control how to dispatch information among human participants of an administrative process, but also to define the business logic necessary to integrate heterogeneous and distributed systems
- combination of WfMS and EAI
  - EAI platform hides the heterogeneity and provides basic interoperability support
  - WfMS supports the definition of the business logic that governs the integration
  - evolution into BPMS
- commercial products: WebSphere MQ Workflow, Vitria BusinessWare, Tibco BPM, BEA WebLogic Integration, BizTalk 2004

functionalities of workflow engine

- **interprets** process definition
- **controls** process instance states such as creation, activation, suspension, termination, ...
- provides facilities for **navigating** between process activities that may involve sequential or parallel operations, deadline scheduling, interpretation of workflow relevant data, ...
- **signs** on and signs off specific participants
- **identifies** work items for user attention and provides an interface to support user interactions
- **maintains** workflow control data and workflow relevant data, and passes workflow relevant data to/from applications or users
- provides an interface to invoke external applications and link any workflow-relevant data
- supports supervisory actions for control, administration, and audit purposes

workflow execution

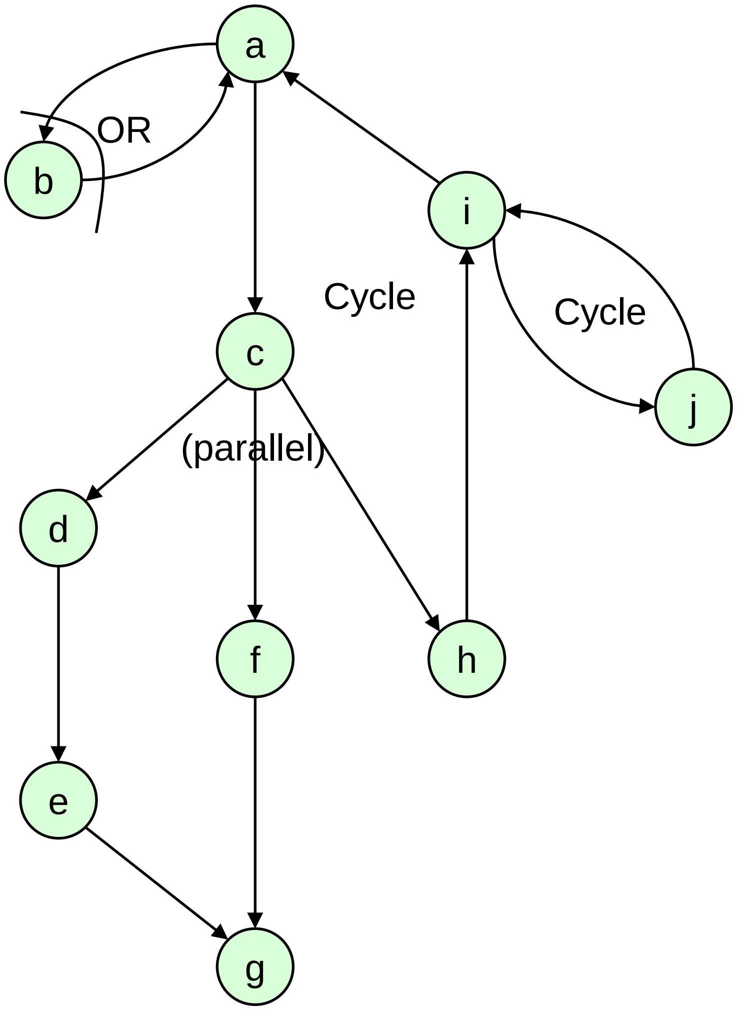
- workflow instances are executed by a workflow engine
- workflow engine is basically a scheduler
  - it schedules the work to be done and assigns it to an appropriate executor (i.e., resource)
  - the work can be a query to be executed on a DB, a SMS to be sent to a cellphone, or a form to be filled by a user
  - the variety of resource types



workflow execution

- Whenever a new workflow is instantiated, the engine retrieves the workflow definition from the repository and determines the node(s) to be executed
- If the node to be executed is a routing node, then the engine simply evaluates the condition and determines which output arc should be activated, and therefore which node should be executed next
- If the node is a work node, then the engine determines the resource to which the node should be assigned for execution, possibly by contacting a resource broker that executes some user-defined resource selection policy
- The engine places the work to be executed into the work queue of the selected resource
- Whenever the resource is ready to execute more work, it retrieves the work item from its work queue, executes it, and returns the result to the engine
- The engine continuously monitors the inbound queue to process work node completion messages
- For each message in the inbound queue, the engine determines the next node to be executed based on the flow graph and on the workflow instance execution state, and then proceeds with resource assignment

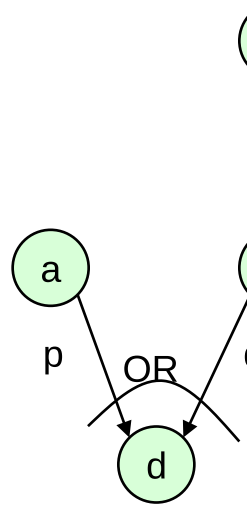
race conditions



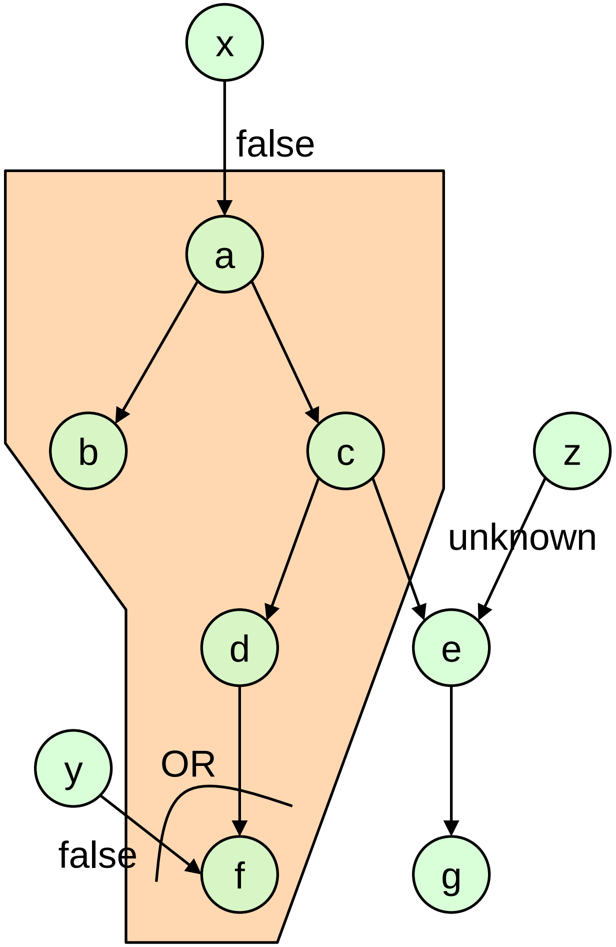
- cycles can introduce race conditions
- what should the process do after the executing cycle c -> h -> i -> a -> c ?
- what if c -> h -> i -> a -> c performs very quickly?
- what is the loop that the process modeler wanted to specify?

dead activities

- dead activity
  - a **regular activity** with an incoming control connector whose transition condition evaluated to be false
  - a **join activity** whose join condition must be evaluated to false
- example
  - p, q, and r are conditions
  - what if a and b are completed and r becomes false and p becomes true? "q cannot be evaluated"



dead path elimination



- *a* is a dead activity
- Dead successors of a, D(*a*) = {*a*}
- Determining all successors of *a* which are not join nodes having a join condition known to be true or unknown: D(*a*) = {*a*, *b*, *c*}
- *d* is a successor of *c* which is not a join node
- *e* is a successor of c and it is a join node of which the truth value of (*z*, *e*) is not yet known: : D(*a*) = {*a*, *b*, *c*, *d*}
- (*d*, *f*) is false by DPE and (*y*,*f*) is false (given) => D(*a*) = {*a*, *b*, *c*, *d*, *f*}
- Since *f* is the only node added and has no successor, the iteration stops

5 workflow interfaces

- process definition interface
  - deals with passing process definitions from external tools to the workflow engine where they are enacted
  - covers the interchange of information such as process start and termination conditions, identification of activities within a process, identification of data types and access paths, definition of transition conditions and flow rules
  - e.g., XPDL (XML Process Definition Language) from WfMC
- workflow client application interface
  - enables the access from a workflow client application to the workflow engine and work-list in a product independent manner
- invoked application interface
  - defines a standard interface allowing a workflow engine to invoke external applications
- workflow interoperability interface
  - defines the mechanism that workflow product vendors are required to implement so that one workflow engine may make requests of another to effect the selection, instantiation, and enactment of known process definitions by that other engine
- administration and monitoring tools interface
  - defines a standard that allows one vendor’s task status monitoring application to work with another vendor’s workflow enactment service

workflow as programming in the large

- As in programming languages, a workflow can have variables that can be passed as input to or taken as output from work node invocations
- Workflows typically compose **coarse-grained activities** and applications that can last hours or days
  - composition of large software modules, complex N-tier systems
- Workflow as a programming language for EAI
- Long-running activities
  - Need for sophisticated failure-handling techniques
  - Need for compensation as well as rollback
  - Should avoid a long lock of DB resource for rollback

dealing with failures

- Forward recovery
  - If the system fails, when it recovers from the failure it will be able to continue the execution of the workflow instances that were active at the time of the failure
- Backward recovery
  - Associates with each work node a compensating activity whose execution semantically undoes the effects of the work done
  - e.g., the compensation for sending a quote
- Exception-handling languages
  - include mechanisms for capturing events that can occur asynchronously w.r.t. to the control flow
  - ECA rules, event nodes, try-catch-throw
  - e.g., order cancellation
- Deadlines
  - Associates a time-out with each step of the process

state of the art

- There are at least 100 workflow tools
- Each tool provides some type of modeling mechanism coupled with an execution framework
- metamodels underlying most workflow tools are based on a variant of activity networks,
  - which show different activities as nodes and use links to represent various temporal and exception dependencies among the nodes
  - or in which nodes can be (i) work node: represents work items to be performed by a human or automated resource, (ii) routing node: defines the order in which work items should be executed, and allow the definition of parallel or conditional activivation of work nodes, or (iii) start and completion nodes
- improving efficiency through automation
  - when workflows involve human workers, the workers can be automatically informed of the tasks they should be performing and given the resources they need to complete the tasks, thereby reducing the idle time

challenges facing workflow technology

- workflow technology is not universally acclaimed, and many CIOs are not convinced of its capabilities and benefits
- current workflow technology is often **too rigid**
  - workflows are constructed prior to use and are enforced by some central authority
  - hard to accommodate the flexible, ad hoc reasoning when an exception occurs
- system requirements are rarely static
  - dynamically changing requirements on a process can necessitate arbitrary extensions not recorded in the workflow model itself
- WfMSs proved to be most useful with repetitive, well-defined processes!

enterprise information systems

- EIS encompasses the BPs and IT infrastructure within enterprises, and delivers the information infrastructure internal to an enterprise
- include
  - enterprise applications that have been developed by an enterprise to meet specific business needs: mostly legacy systems
  - legacy systems and applications that manage mission critical data to BPs within an enterprise
  - ERP systems that provide a unified, organization-wide view of BPs and associated information
  - CRM that encompasses an organization’s end-to-end engagement with its customers over the lifetime of its relationship with them
  - transaction processing systems and applications