Prof. Dr.-Ing. Stefan Deßloch
AG Heterogene Informationssysteme
Geb. 36, Raum 329
Tel. 0631/205 3275
dessloch@informatik.uni-kl.de

TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

Chapter 10 – Business Process Modeling and Workflow Management

Introduction & Motivation
Business Process Modelling
Workflow Management Systems
Web Services Orchestration & Choreography
Workflows and Transactions

Terminology

- Material process
  - assembles physical components, delivers physical products
  - may include moving, storing, transforming, measuring, assembling objects
- **Information process**
  - relates to automated and partially automated tasks that create, process, manage, provide information
    - involves programs, humans (interacting with computers)
  - infrastructure provided by database, transaction processing, and distributed systems technology
- **Business process**
  - market-centered description of an organization's activities
  - implemented as an information process and/or material process
- A **workflow** is a business process in execution (an instance of a process model) in a computing environment
  - Not all parts of a process are run in a computing environment - some processes are not run on a computer at all!
  - Often, "workflow" and "process" is identified

Role of Workflow Technology

- Applications support business processes and have to ensure compliance with business processes

  ➔ Application = Business Process + Business Functions

  - Large applications often use special "control programs" to ensure the appropriate/correct sequencing of business functions
- Changes in how to perform business must be reflected as soon as possible in applications
  - Requires code changes [which part to change?...], recompilation, redistribution of code,... to reflect new business processes
- What if users of standard applications want to reflect their own processes?
  - very difficult, cumbersome, expensive (service specialists, consultancy)
- Consequence: Implementation of control programs via workflows
  - Application consists of collection of business processes and collection of business functions (= "usual" programs)
  - Business processes are enacted by workflow system that invoke business functions "appropriately", i.e. according to process model
- No coding,... to adapt application to changed business process

Historical Background

- Electronic document and folder routing in *office automation*
  - Routing through enterprise's organizational structure
  - Potential flow of documents prescribed in advance
    - Routing conditions in terms of document content or document properties
- Business processes involving functions provided by application systems
  - Launch-pad for executables, work item management
  - Launching executables requires parameter passing
    - Data flow features complemented available control flows
    - Control flows expressed in terms of these new parameters ("business rules")
- "Production Workflow" driving operational aspects of an enterprise
  - Modeling of business process logic and separation from business functions
  - Significant improvements regarding high availability, scalability, robustness
    - advanced transaction management (forward/backward recovery, compensation)
  - WFMS become an EAI-tool (often integrated with other EAI middleware)
    - business process logic includes integration logic

Business Process (Re-)Engineering

- Make company more flexible, react faster to change
  - outsourcing of processes, supply-chains, virtual enterprises,...
- Speed up, reduce costs
  - eliminate unnecessary tasks, employ parallelism, deadline processing, monitoring/auditing, analysis, automation
- New processes are defined, existing are changed or abandoned
  - existing business processes must be analyzed, specified and modeled, optimized (includes simulation)
- Important to include resources used to perform processes
  - organizations, roles, people
  - IT resources
- Scope is not only intra-enterprise but also inter-enterprise
  - Business-to-Business, Consumer-to-Business, Business-to-Administration,...
- Reengineered processes may be implemented using a WFMS

Web Service Orchestration/Choreography

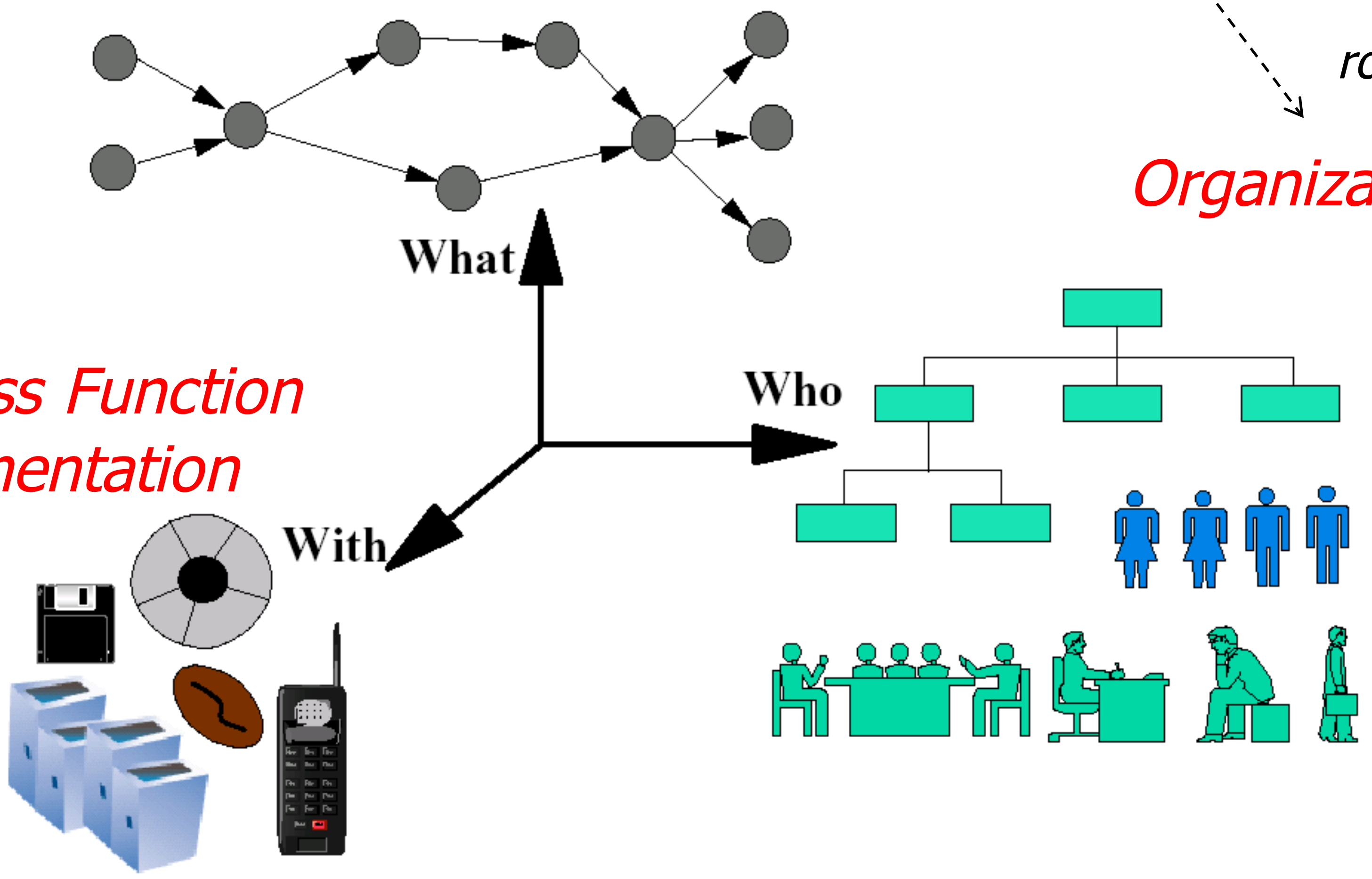
- Complex web services
  - need to interact with business partners through web services
  - may combine/utilize existing web services
  - requirements similar to BPM, Workflow Management
    - separate function from composition logic, …
- Web services composition (aka orchestration)
  - ability to create new web services out of existing (web service) components
  - focus is on implementation of operations in a web service
    - internal, private, for automation of the execution of a composite web service
  - composition can be iterated
    - composition result is again a web service
    - can be used as a building block for further composition steps
- Web services coordination (aka choreography)
  - process involves multiple enterprises or business partners
  - focus is on conversation protocols
    - public, standardized protocols
    - external coordination for verifying compliance

The Three Dimensions Of Workflow

Business Process ← BPR Focus

roles

Organization



Business Function Implementation

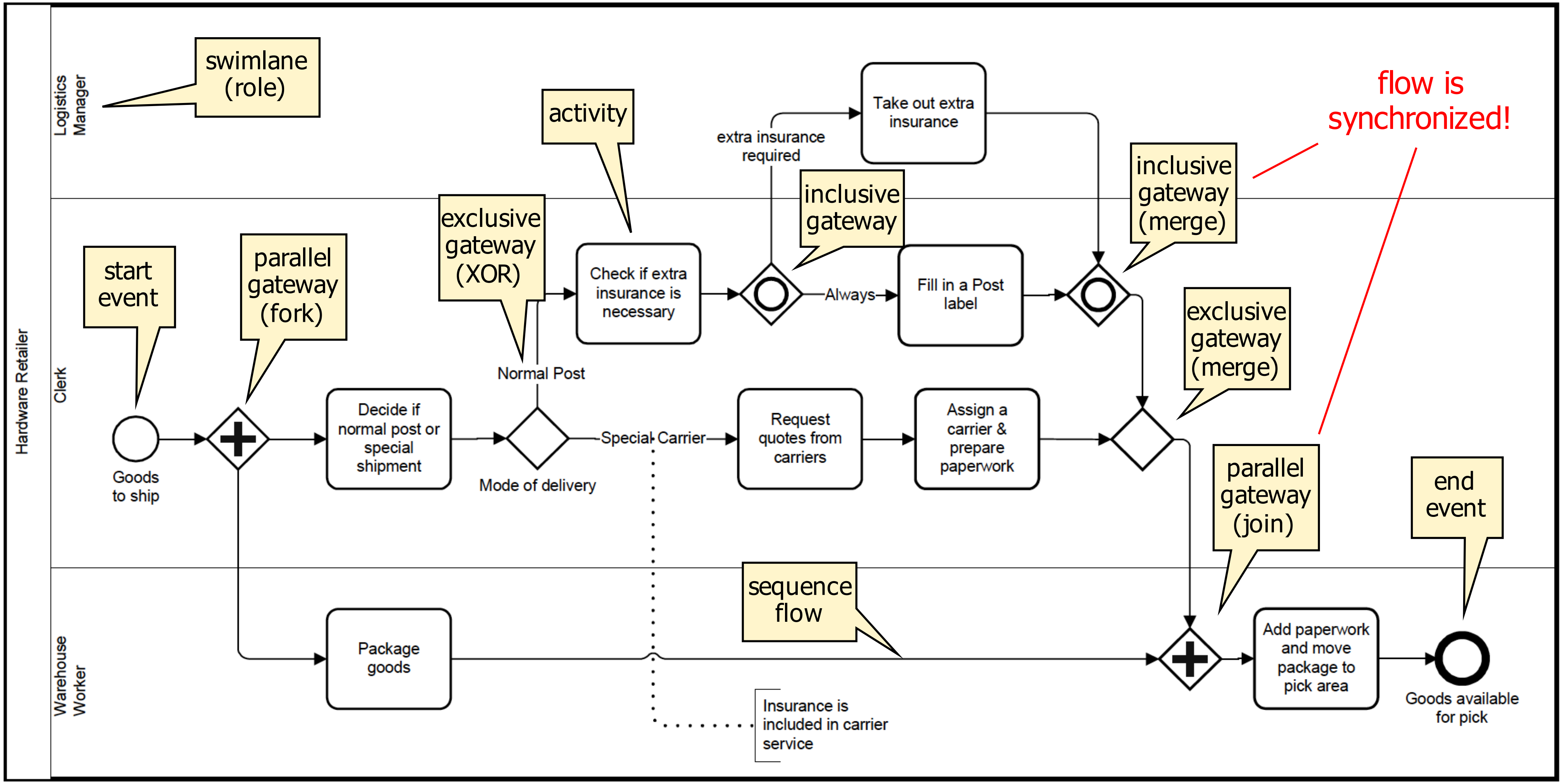
Business Process Modelling

- Business processes have to be modelled/specified explicitly
  - documentation
  - basis for analysis, optimization, reorganization, resource planning
  - starting point for (executable) business process definition (→ workflow management systems)
    - similar to information modelling for database schemas (ER → RM → DBMS)
- Business process modelling tools and languages
  - numerous approaches, at different levels of abstraction
  - are typically used by business analysts and process domain experts
  - usually follow a graphical notation approach
    - may or may not have a well-defined execution semantics or formal foundation
- Examples
  - ARIS (IDS Scheer) – event-driven process chain model
  - **B**usiness **P**rocess **M**odelling **N**otation - BPMN (OMG)
    - (graphical) notation for business processes with formalized execution semantics
    - interchange format
    - mapping to web service composition language WS-BPEL 2.0

BPMN Diagram Elements - Categories

- Flow Objects
  - events (start, intermediate, end)
  - activities (tasks, sub-processes)
  - gateways (for branching, forking, joining, merging of sequential flows)
- Data
  - objects, inputs, outputs, stores
- Connecting Objects
  - sequence flows
  - message flows
  - associations
  - data associations
- Swimlanes
  - pools, lanes
- Artifacts
  - groups, text annotations

BPMN Core Concepts – Shipment Example



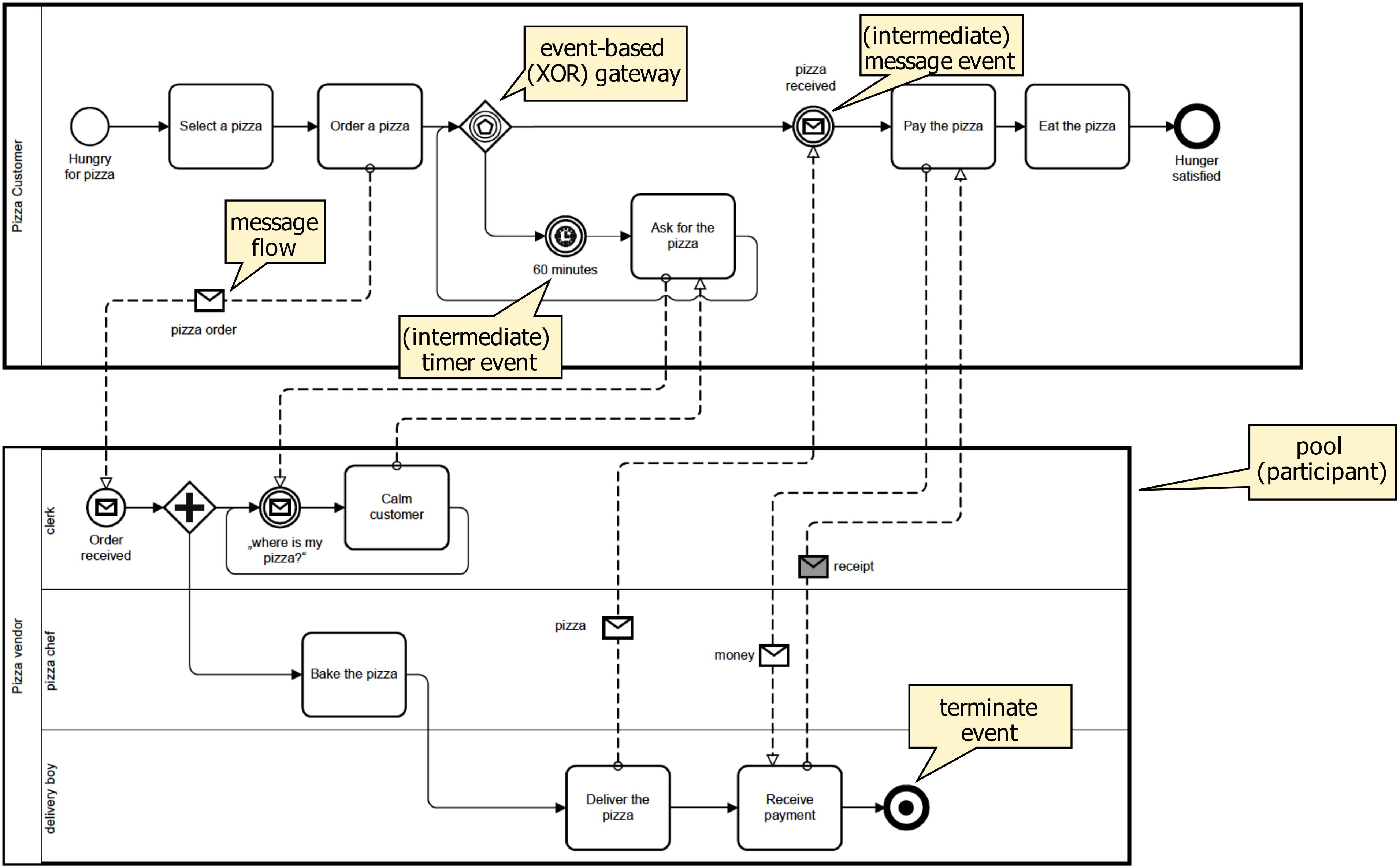
Source: *BPMN 2.0 by example,* OMG, http://www.omg.org/spec/BPMN/2.0/examples/PDF

BPMN – Modelling Scope

- Processes
  - private (internal to a specific organisation)
    - executable – full level of detail, well-defined execution semantics
    - non-executable – some details are omitted, modelled for documentation purposes
  - public
    - represents interactions between a private process and another process or participant
    - shows order of message flows needed to interact with the process
- Collaborations
  - interactions between two or more business entities
  - two or more public processes communicating with each other
- Choreographies
  - self-contained definition of expected behavior between interacting participants
  - similar to a private process with activities that represent sets of message exchanges involving two or more participants

Reference: *Business Process Model and Notation (BPMN) Version 2.0*
OMG, May 2010, http://www.omg.org/spec/BPMN/2.0

BPMN – Pizza Collaboration Example



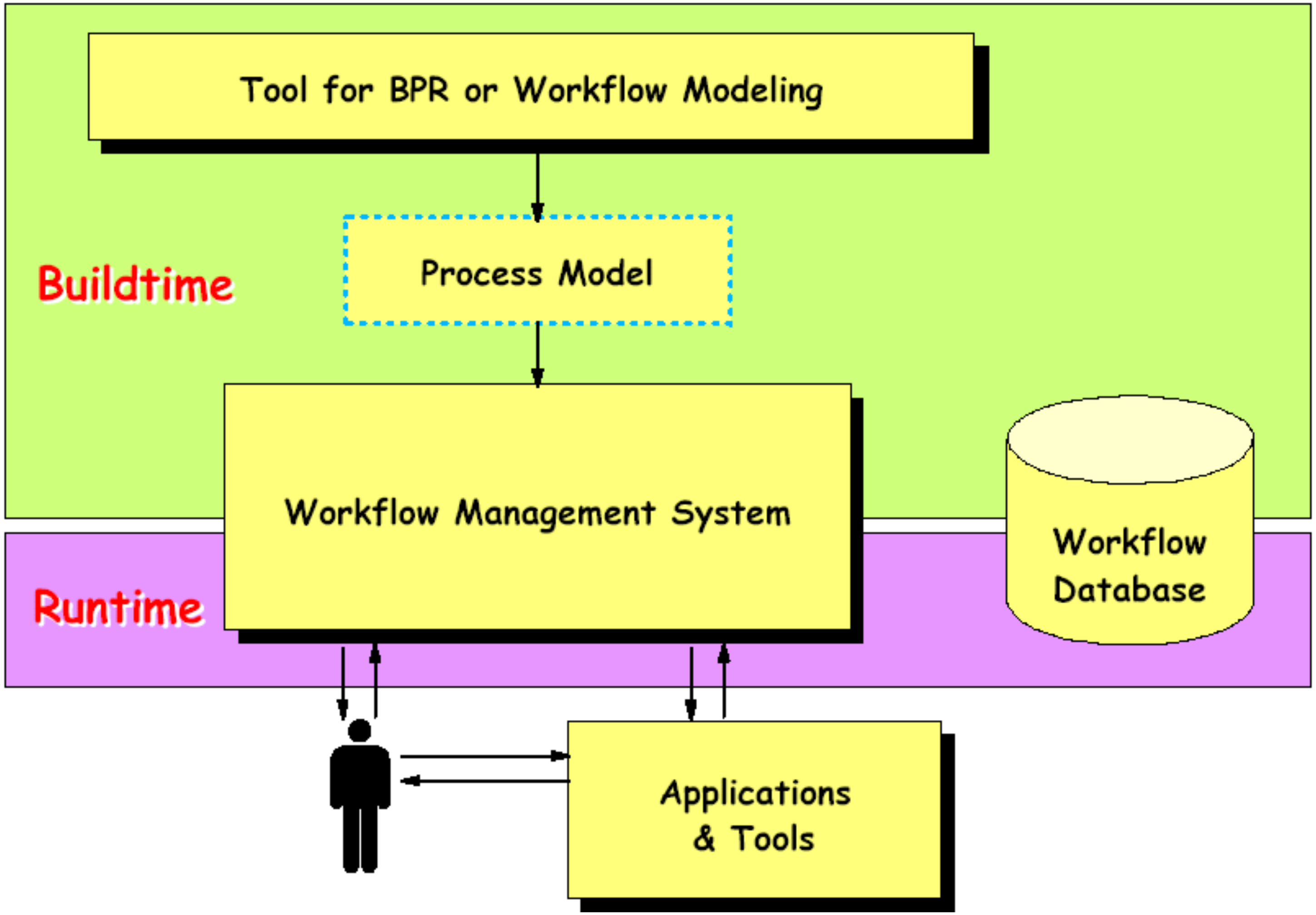
Business Process Optimization

- Static analysis of flows through organization
  - restructure to achieve high level of parallelism
  - minimize occurrences of control flow crossing organization boundaries
    - reorder and combine multiple activities into a single activity (or a stream of activities) per role
    - optimize the organization (i.e., change organization-department-employee aspects)
- Dynamic analysis (simulation)
  - used to compare and select from alternative models of a given business process the "optimal" one (in terms of metrics like cost, duration,...)
  - involves quantitative aspects
    - requires process instrumentation
      - number of processes per time unit, probabilities of routing conditions being satisfied, average duration of activities, processing power and availability of resources
  - analytical simulation
    - calculates how often an activity has to be performed, probability of execution paths, duration and probability of process execution paths
  - discrete (event) simulation
    - calculation/simulation at the level of individual resources
    - considers availability, resource competition

Business Process Definition

- Business process definition specifies
  - steps of the process
  - the work performed in each step
  - order of execution of steps (control flow)
  - how steps communicate (data flow)
  - the people, identified by roles, who carry out the steps
- Business process steps are
  - encapsulated
  - reusable
- Definition can be
  - distributed among the process steps
    - each step processes a request and includes control flow/data flow logic
  - or expressed as a single program (e.g., script-like programs)
  - or specified using a special-purpose programming/specification language
    - "programming in the large" - typical for workflow management systems

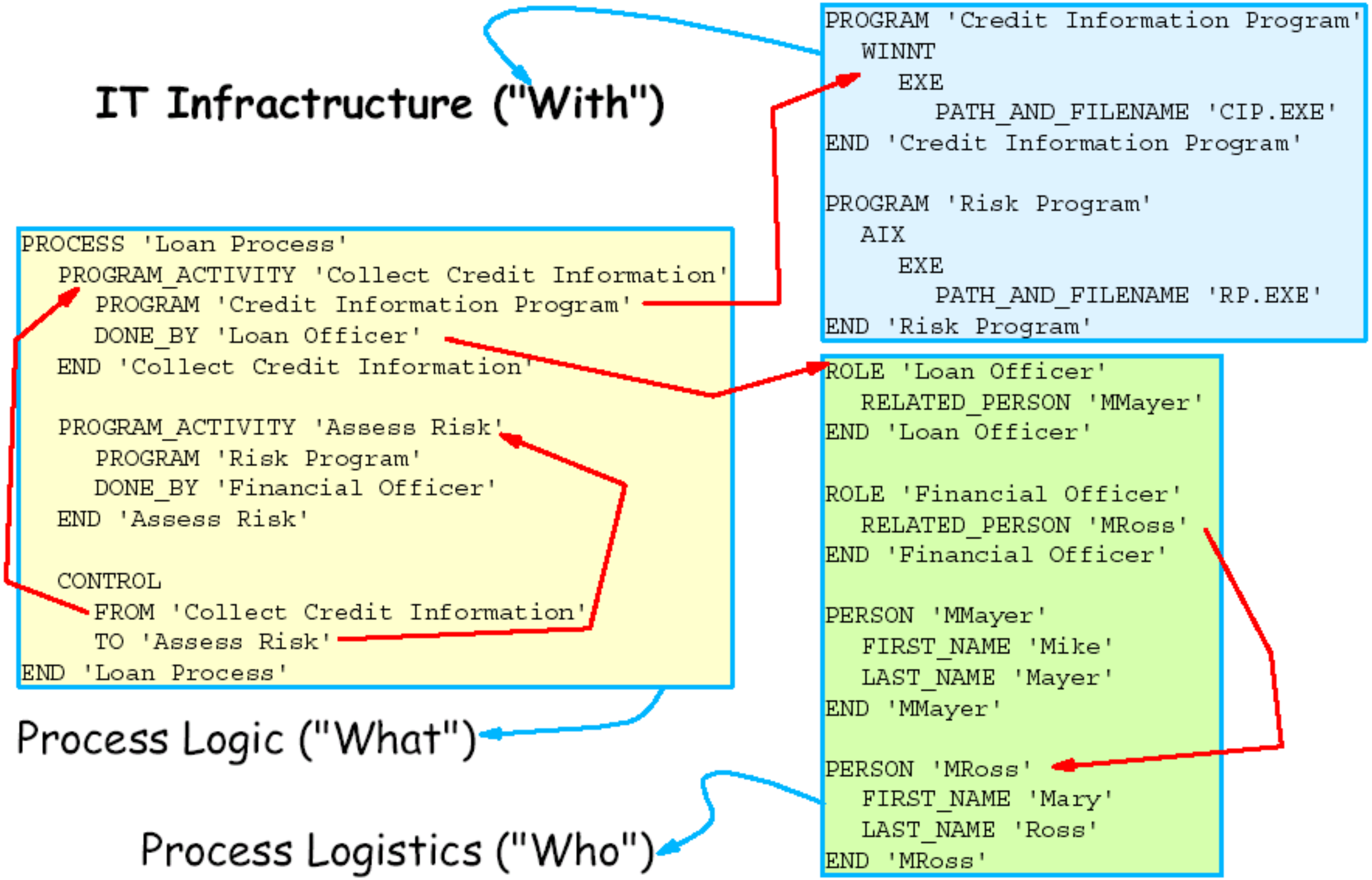
Major Building Blocks Of A WFMS



Buildtime

- Component providing all functions and capabilities to define, test and manage all workflow related information
  - Especially, all three workflow dimensions are covered
  - Often, administrative and systems management information is included, e.g.
    - Session threshold, i.e. maximum period of time a user can work with the WFMS
    - Actions to be taken when average response time exceeds threshold
  - All information stored in WFMS own database ("buildtime database")
- Two different kinds of interfaces
  - Graphical end user interface
  - Workflow Definition Language
    - ASCII text with special syntax/semantics
      - Most often vendor specific
        - e.g., IBM's FDL (Flow Definition Language)
      - Standard developed by Workflow Management Coalition (WfMC)
        - WPDL (Workflow Process Definition Language)
        - XPDL (XML Process Definition Language)
  - Both GUI and WFDL cover all concepts of the WFMS Meta Model

Workflow Definition Language: Example (FDL)



IT Infractructure ("With")

```
PROGRAM 'Credit Information Program'
  WINNT
    EXE
      PATH_AND_FILENAME 'CIP.EXE'
END 'Credit Information Program'

PROGRAM 'Risk Program'
  AIX
    EXE
      PATH_AND_FILENAME 'RP.EXE'
END 'Risk Program'
```

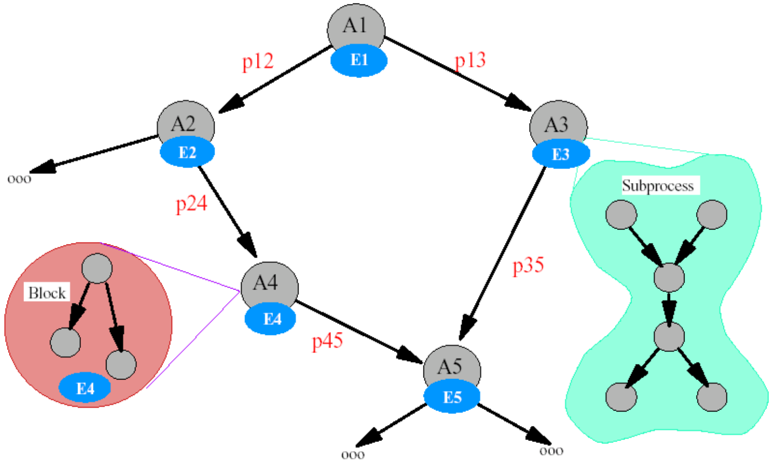
```
PROCESS 'Loan Process'
  PROGRAM_ACTIVITY 'Collect Credit Information'
    PROGRAM 'Credit Information Program'
    DONE_BY 'Loan Officer'
  END 'Collect Credit Information'

  PROGRAM_ACTIVITY 'Assess Risk'
    PROGRAM 'Risk Program'
    DONE_BY 'Financial Officer'
  END 'Assess Risk'

  CONTROL
    FROM 'Collect Credit Information'
    TO 'Assess Risk'
END 'Loan Process'
```

Process Logic ("What")

```
ROLE 'Loan Officer'
  RELATED_PERSON 'MMayer'
END 'Loan Officer'

ROLE 'Financial Officer'
  RELATED_PERSON 'MRoss'
END 'Financial Officer'

PERSON 'MMayer'
  FIRST_NAME 'Mike'
  LAST_NAME 'Mayer'
END 'MMayer'

PERSON 'MRoss'
  FIRST_NAME 'Mary'
  LAST_NAME 'Ross'
END 'MRoss'
```

Process Logistics ("Who")

"What" Dimension: Control Flow



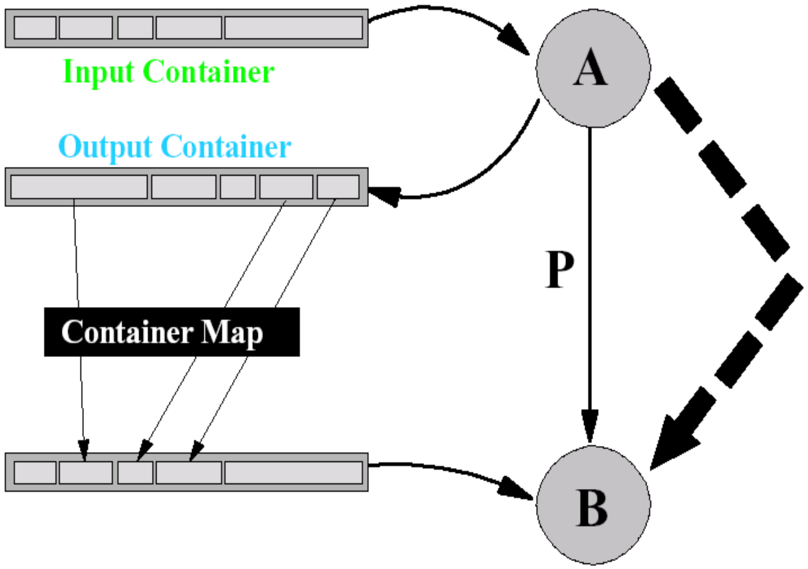
- Different Types of Activities
  - Information Activity
    - inform user to take some actions
    - no implementation
  - Program Activity
    - implemented by a program
      - different types of binding
  - Process Activity
    - activity implemented as a sub-process
      - different types of connection
  - Bundle Activity
    - the same activity is implemented on a set of objects
      - parallel execution as an option
  - Block Activity
    - provides DO-UNTIL behavior as special construct
      - process model often restricted to DAG
    - exit condition determines looping behavior

Subprocesses

- Subprocess may be **local** (performed by the same WFMS) or **remote**
  - remote WFMS may be from the same or a different vendor (requiring standardized formats and protocols)
- Autonomy of subprocesses (governed by autonomy rules)
  - **autonomous**: once started subprocess cannot be influenced by the parent
  - **controlled**: life-cycle of the subprocess is determined by the parent process, e.g.
    - suspension of the implemented activity forces the subprocess to suspend
      - tight administrative coupling
  - whole spectrum between these extremes can be defined
- Nested subprocesses
  - controlled subprocess returns control to parent after completion
  - nesting hierarchy

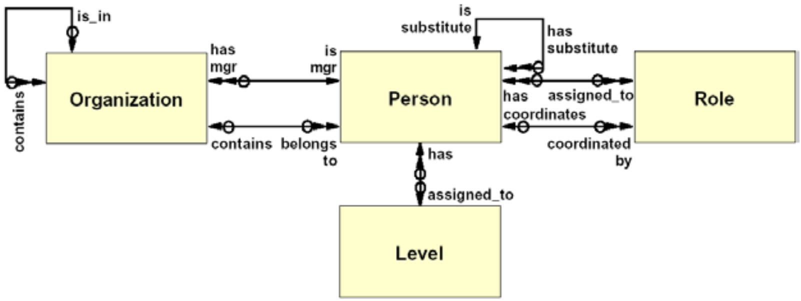
"What" Dimension: Data Flow

- Input/Output Container
  - defines data passed to/returned by process or activity
    - based on simple/structured types
      - definitions can be shared
    - can also specify default values
  - provides the execution context
- Data Connectors
  - specify which data needs to be copied where
  - details provided by container map
    - field/data type mapping
    - data transformations
- WFMS at runtime
  - materializes input container instance before activity is started
    - may utilize so-called dead data maps
  - makes output container instance persistent



“Who” Dimension: Organizational Aspects

- WFMS organization schema
  - either **fixed** (built-in) by the WFMS vendor
    - often simple, can be implemented efficiently
  - or **dynamic**, allowing to change the entities and relationships
    - very flexible, but hard to achieve efficiency
    - requires mapping (see discussion below)



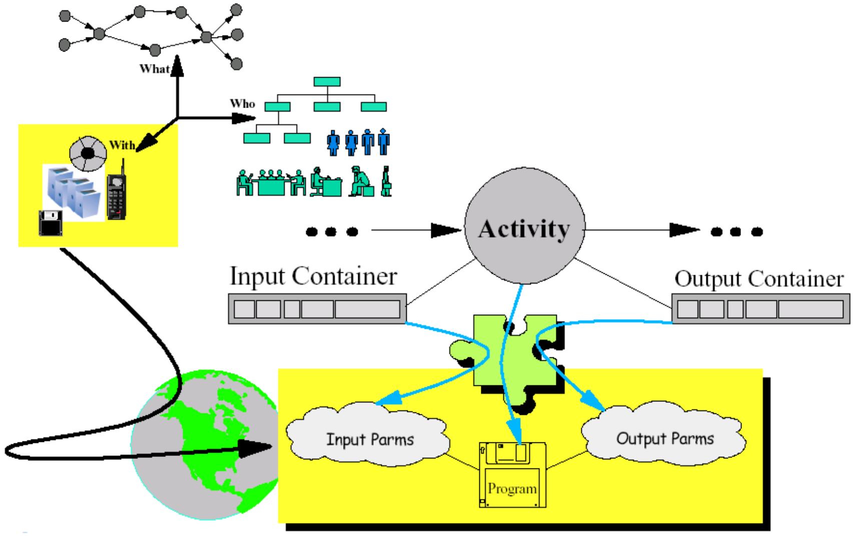
- Organizational data is
  - either managed by WFMS in its own database
    - optimized schema, no performance impact on source systems, <u>but</u>: often replica of "real" org. database, might run out of sync
  - or shared with other systems
    - no redundant data, <u>but</u>: performance impact, usually involves dynamic mapping
- Staff Resolution: performing staff queries at runtime
  - staff queries need to be attached to each activity for staff assignment
  - may require interaction with external org data system using a staff resolution exit (external program), mapping WFMS org schema to external schema, or directly using external DBMS interface/schema

Defining Worklists

- Worklist: collection of workitems that have the same common characteristics
- Characteristics are defined via queries on workitem properties
  - Especially, a workitem can be on multiple worklists
    - Worklists of different agents
    - Different worklists of the same agent
- Not only people or program executors may have worklists but also each instance of any element of the org metamodel
  - Worklists associated with an org instance that collects multiple people is called a group worklist
  - All users belonging to the group associated with the group worklist can pick a workitem from that list
- Modes of worklists
  - Pull
    - explicitly request refresh
    - suitable for high throughput environments, where certain worklists might be in constant flux!
  - Push
    - are always up to date
  - Grab
    - deliver a matching workitem on request ("get next workitem")
    - convenient for group worklists

"With" Dimension: Program Registration

- Decoupling activities and implementation
  - business process modeler can focus on process models, programs can be linked to activities later
  - programs depend on the environment they are running in
    - often program signatures depend on the environment
      - mapping from container to signature must be specified: "Data Mapping Language"
  - programs should be able to be exchanged without requiring to modify process models ("late binding")
- WFMS can resolve actual program to call when activity implementation must be invoked, based on meta-data for
  - program calls
  - method invocation
  - message queuing
  - TP-monitor interactions



Web service to the rescue!

Putting Process Models Into Production

- When modeling a process is finished it can be put into production
- Putting a process model into production means
  - ...to "freeze" the model, i.e. nobody can change it any more
    - Only "what" dimension (the activities and control-/dataflow between them) is really frozen
    - Organization model ("who dimension") can of course be modified
      - E.g. people can change departments
      - Might impact staff queries (e.g. dropping a department a query refers to): If no agent is found process administrator is notified
      - Often, organizational structure is completely maintained via separate application (e.g. Human Resource) and replicated periodically into the WFMS database in batch mode
    - Activity implementations ("with dimension") can be "early bound" or "late bound"
      - Early bound process model is frozen too, late bound process model is resolved at runtime
  - ...often to TRANSLATE the corresponding data into a different format
    - Modeling tool and WFMS runtime might use database structure optimized to their needs
  - ...often to create a new version of an already existing process model ("valid from")
    - Existing instances of earlier versions are run according to the model which was valid when the instance has been created (auditability is a key requirement!)
    - New instances are created according to the new version
- Once put into production, instances can be made from a model

Runtime

- WFMS proactively drives the processes
  - process navigation
  - interaction with end users, applications
- Support of process queries to locate a particular process or set of processes
  - may be based on operational (e.g., start date) or business selection criteria (e.g., customer name)
  - a *key container* may be defined and filled with interesting data for that purpose
- Audit trail
  - recording of important events in the life of a process
  - possible usages
    - legal requirements
    - analysis for process reengineering
- Monitoring process collections for out-of-line situations
  - "Leitstand"
- Process Repair
  - administrator gets notifications about erroneous situations
  - may manually fix errors (e.g., content of containers, state of activity, assigned implementation or resource)

Web Services vs. WFMS

- Limitations of conventional composition middleware (e.g., WFMS)
    - Significant effort to integrate existing applications
        - application-specific adapters, wrappers
    - Limited success of composition model standardization
        - WfMC standard is not widely implemented
- Opportunities for Web Services
    - Web Services seem to be adequate components
        - well-defined interfaces, described using WSDL, standardized invocation (SOAP)
    - Significant efforts in standardizing WS composition languages
    - Reuse of existing WS "infrastructure" (directory, service selection, ...)
        - WS composition tools are less expensive to develop
- Business Process Execution Language for Web Services (BPEL4WS)
    - XML-based language for specifying business process behavior based on web services (inspired by WSFL (IBM) and XLANG (Microsoft))
    - Describe business processes that both provide and consume web services
        - Steps (activities): implemented as an interaction with a web service
        - Information flow into/out of the process: externalized as web service

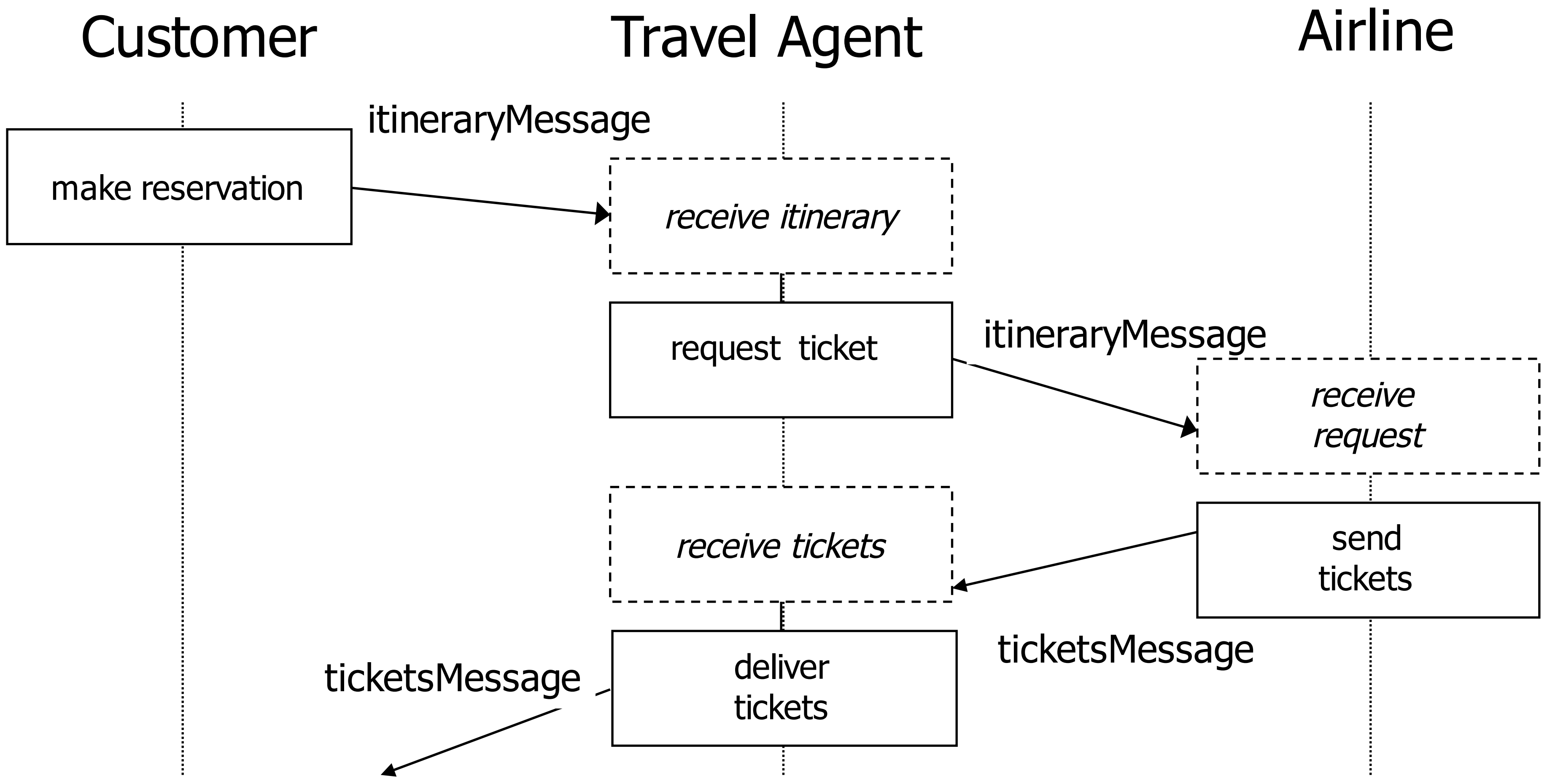
BPEL4WS

- BPEL can support specification of both, composition schemas and coordination protocols
  - can be used in both composition and coordination middleware
- Two types of processes
  - executable process (-> composition)
    - defines implementation logic for a composite web service
    - portable between BPEL-conformant environments
  - abstract process (-> coordination)
    - service-centric perspective on coordination protocols
    - describe message exchange between partners
- Business process defines
  - potential execution order of operations (web services)
  - data shared between the web services
  - correlation information
  - partners involved in business process and interfaces they need to implement
  - joint exception handling for collection of web services

BPEL Component Model

- Components are web services described using WSDL
  - abstract WSDL interfaces are referenced in BPEL scripts
  - no reference to bindings, endpoints, or services
- Basic activities in BPEL represent components, correspond to WSDL operations
  - Invoke
    - Issue an asynchronous request, or
    - Synchronously invoke a request/reply operation of a web service provided by a partner
  - Receive
    - Wait for a message to be received from a partner
    - Specifies partner from which message is to be received, as well as
    - The interface and operation provided by the process
      - Used by the partner to pass the message
  - Reply
    - Synchronous response to a request corresponding to a receive activity
    - Combination of Receive/Reply corresponds to request-response operation in WSDL

Example


Customer
Travel Agent
Airline
itineraryMessage
make reservation
receive itinerary
request ticket
itineraryMessage
receive request
receive tickets
send tickets
ticketsMessage
deliver tickets
ticketsMessage

Service Selection: Partner Links

- Partner link (BPEL process definition)
  - identifies the web services mutually used by the partner or process
    - e.g., agent process interacts with customer, airline
  - references a partner link type
  - defines role taken by the process itself (myRole) and role that has to be accepted by the partner (partnerRole)
- Partner link names are used in all service interactions to identify partners
  - see activities for invoking/providing services

- Partner link type (WSDL extension) defines
  - roles played by partners in a conversational relationship
  - web service interfaces that need to be implemented to assume a role
- Assignment of endpoints for partners
  - at deployment time
  - dynamically at run time

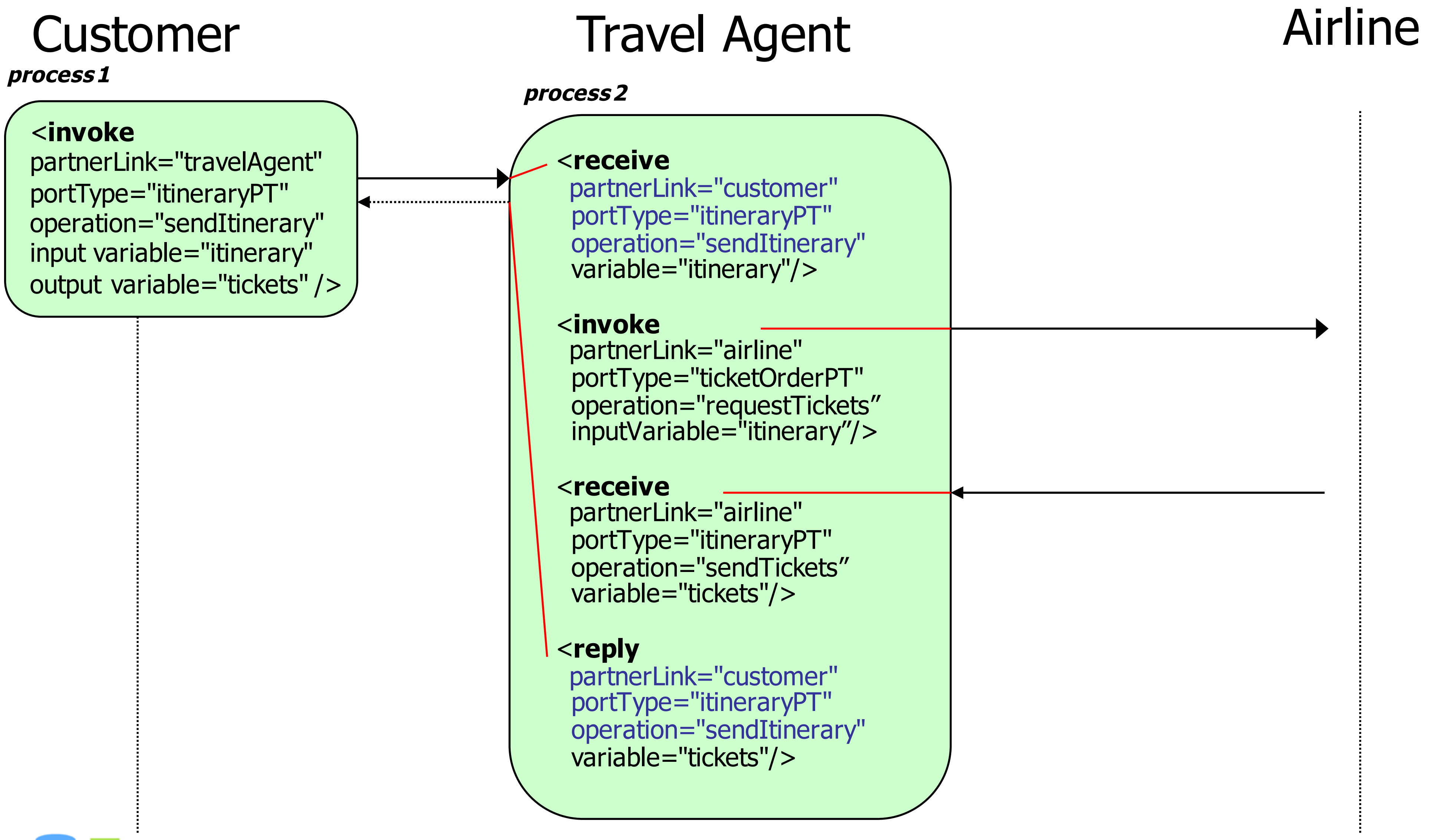
```
<process name="ticketOrder">
<partnerLinks>
  <partnerLink name="customer"
               partnerLinkType="agentLink"
               myRole="agentService"/>
  <partnerLink name="airline"
               partnerLinkType="buyerLink"
               myRole="ticketRequester"
               partnerRole="ticketService"/>
</partnerLinks>
```

↔

Partner link type definition

```
<partnerLinkType name="buyerLink">
  <role name="ticketRequester">
     <portType name="itineraryPT"/>
  </role>
  <role name="ticketService">
     <portType name="ticketOrderPT"/>
  </role>
</partnerLinkType>
```

BPEL Activities – Example



Customer

process 1

```
<invoke
partnerLink="travelAgent"
portType="itineraryPT"
operation="sendItinerary"
input variable="itinerary"
output variable="tickets" />
```

Travel Agent

process 2

```
<receive
 partnerLink="customer"
 portType="itineraryPT"
 operation="sendItinerary"
 variable="itinerary"/>

<invoke
 partnerLink="airline"
 portType="ticketOrderPT"
 operation="requestTickets”
 inputVariable="itinerary”/>

<receive
 partnerLink="airline"
 portType="itineraryPT"
 operation="sendTickets”
 variable="tickets"/>

<reply
 partnerLink="customer"
 portType="itineraryPT"
 operation="sendItinerary"
 variable="tickets"/>
```

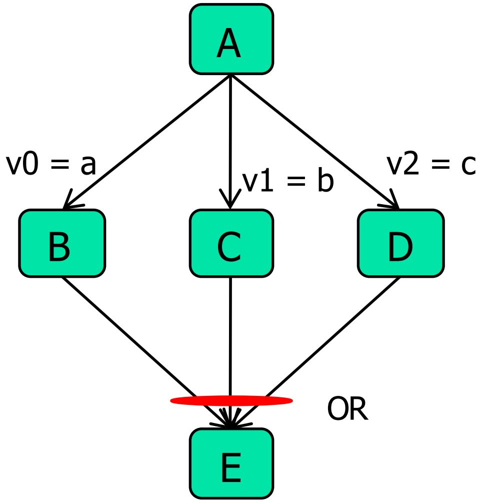
Airline

Orchestration Model - Structured Activities

- Sequence
  - Enclosed activities are carried out in listed order
- If-else (i.e., switch)
  - Selects one of several activities based on selection criteria
- Repetitive Activities
  - While, RepeatUntil,
    - repeatedly carry out enclosed activities while/until specified condition is true
  - ForEach
    - serial: enclosed activity (scope) is carried out repeatedly, based on counter, optional completion condition
    - parallel: (effective copies of) enclosed activity (scope) executed n+1 times in parallel, based on start/end counter values
- Pick
  - Specifies a set of activities with associated events (e.g., receipt of message)
    - messages can be received from the same or different partners
    - activity is completed when one of the events occurs

Structured Activities (cont.)

- Flow activity: defines sets of activities plus (optional) control flow
  - all activities can (potentially) execute in parallel
    - flow activity completes when all directly nested concurrent activities complete
    - implicit fork/join behavior
  - activities can be "wired together" via control links
    - link has one source activity, and one target activity
    - transition conditions
      - evaluated after source activity completes
      - determines the link status to be either true or false
      - links status also set to false, if source activity is determined not to be executed (e.g., if-else)
    - join conditions
      - can refer to status of incoming links of a target activity (e.g., AND, OR)
      - are evaluated only after the status of all incoming links is known
      - false join condition results in a join failure



Process life-cycle

- Start activities
  - receive, pick – createInstance attribute
    - creates a new process instance, if it doesn't exist already
  - Example:

```
<receive partner="customer",
      portType="itineraryPT",
      operation="sendItinerary",
      variable="itinerary”
      createInstance="yes"/>
```

  - each process must have at least one start activity as an initial activity
- Process termination
  - process-level activity completes successfully
  - fault "arrives" at the process level (handled or not)
  - terminate activity is invoked

Data Types and Data Transfer

- Variables can be used to define data containers
  - WSDL messages received from or sent to partners
  - Messages that are persisted by the process
  - XML data defining the process state
- Constitute the “business context” of the process
- Access to variables can be serialized to some extent

```
<variables>
     <variable name="itinerary“ messageType="itineraryMessage"/>
     <variable name="tickets" messageType="ticketsMessage"/>
</variables>
```

- Variable assignment
  - Receiving a message (or a reply of an invoke activity) implicitly assigns value
  - Alternative: assign activity (another simple activity)
    - Copies fields from containers into other containers

Correlation

- Message needs to be delivered not only to the correct port, but to the correct instance of the business process providing the port
  - conversation routing
- Correlation Set
  - one or more properties used for correlating messages
  - example

    -
      ```
      <correlationSets>
        <correlationSet name="Booking"
              properties="orderNumber"/>
        …
       </correlationSets>
      ```
  - correlation properties are like "late-bound constants"
    - binding happens through specially marked message send/receive activities
    - value must not change after the binding happens
- Often, more than one correlation set is used for an entire process
  - example: orderNumber -> invoiceNumber
  - correlated message exchanges may nest, overlap
  - same message may carry multiple correlation sets

Properties

- Property
  - Globally defined types
  - Primarily used to correlate a message with a specific process instance
    - E.g., order number
    - Usually included in the message
    - Often the same property is used in different messages
  - Can be defined in BPEL as a separate entity:

```
<property name="orderNumber" type="xsd:int"/>
```

- Property alias
  - Allows to point to a dedicated field of the message that represents the property
    - Usually different for each message type
    - Can be used in expression and assignments to easily use properties

```
<propertyAlias propertyName="orderNumber"
               messageType="ticketsMessage“
               part="orderInfo“
               query="/orderID"/>
```

Business Processes and Transactions

- One TA per business process usually does not work!
  - resources (people) may not be available at process instantiation time
  - real-world constraints on process steps and human interaction
  - system constraints (e.g., no support for distributed TA across partners)
  - function encapsulation (process involves several separate systems or services)
  - resource contention (affects performance and finally availability)
- Process requires execution of multiple transaction steps
  - further generalization of multi-step transactions, stratified transactions required
  - the complete process is no longer protected by ACID-properties
- Required failure characteristics of business process ("business transaction")
  - must survive (planned as well as unplanned) interrupts
    - rollback of the whole process due to local failure is not tolerable

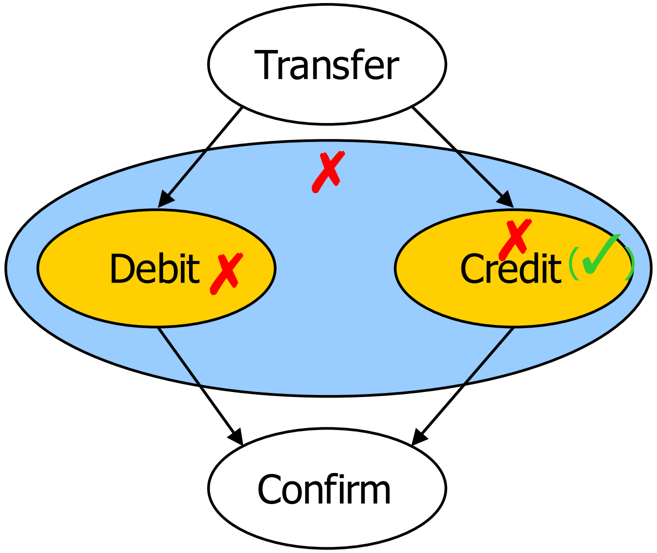
    ➔ forward recovery based on persistent process state
  - if process is terminated (i.e., finally fails), previous work needs to be reverted

    ➔ semantic (undo) recovery based on compensation

Atomic Spheres (global TAs)

- Set of activities where either all activities in a sphere commit, or none
- Properties:
  - Each activity in an atomic sphere is **transactional**
    - Manipulates resources in RM according to X/OPEN DTP
    - Does not establish TA boundaries by itself
  - All connectors entering the sphere have the same activity as their origin
    - Ensure fast execution of sphere (distributed TA)
  - If an activity is rolled back, then all previously completed activities in the sphere are rolled back as well
- In case the sphere is rolled back, it can be automatically rerun by the WFMS
  - based on persistent input containers
- Atomic spheres allow reuse of existing transactional programs

  → implement transaction steps within processes



Atomic Sphere (cont.)

- WFMS implementation
  - Start global TA when control flow enters atomic sphere
  - Wait for running activities in sphere to complete when control flow leaves the sphere, and commit global TA
    - If commit fails, carry out further steps (repeat, exception WF, …) based on sphere parameters
- Global Transactions: Practice
  - Transaction with multiple participants
  - Atomic commitment is the issue
    - E.g. 2-phase-commit protocol
  - Not realistic across organization boundaries
    - Not only „efficiency" issues but additional legal-, ownership-, privacy-,... issues
    - Especially not in internet scenarios

Compensation

- An action used to logically undo the effects of another action is called compensation action
  - Extends to real world actions
    - drilling a hole: throw away part
  - Semantic Recovery: Recovery schema based on compensation
  - Compensation very likely one of today's most frequently exploited techniques in transaction processing
- Compensation action is often dependent on context
  - E.g. writing an offer and sending it via mail to a customer
    - If letter is still in outbasket, simply remove it from outbasket
    - If letter is already received by the customer, write and send a countermanding letter
- Compensation often cannot recreate the same state that existed before the proper action had been performed
  - E.g. canceling a flight might cost a cancellation fee
    - Even more complicated, the cancellation fee might depend on the point in time, i.e. it is higher the later the cancellation is requested
- Compensation action may fail!

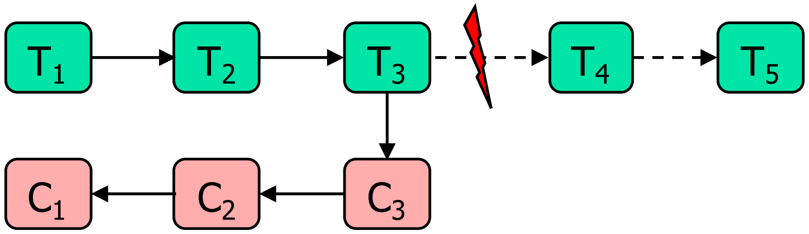
Sagas – Transactions and Compensation

- Sagas support specification of compensation actions in advance and run them automatically on abort
  - Sequence of (Sub-)Transaction/compensating action pairs
  - Guaranteed LIFO execution of compensation actions during abort/rollback of Saga
  - ACID for each sub-TA

Definition:

A <u>Saga</u> is a sequence $[(T_1,C_1),\ldots, (T_n,C_n)]$ having the following properties:

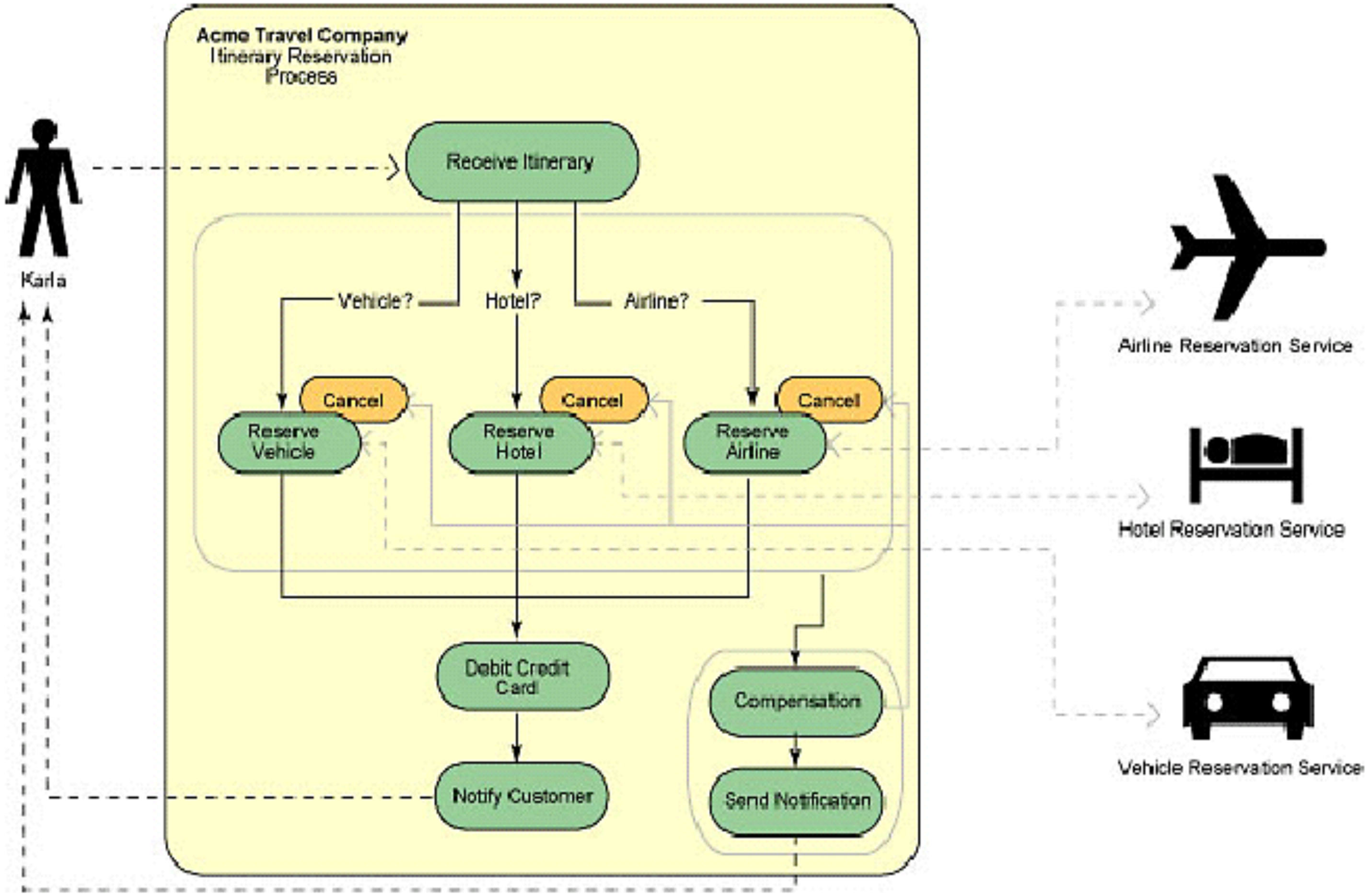
1. $T_1,\ldots,T_n$ and $C_1,\ldots,C_n$ are two sets of transactions, such that C_i is the compensation function for T_i,
2. $[(T_1,C_1),\ldots, (T_n,C_n)]$ is executed as one of the following sequences:
   i. $[T_1,\ldots,T_n]$, if all T_i committed, or
   ii. $[T_1,\ldots,T_i, C_{i-1},\ldots, C_1]$ if T_i aborts and $T_1,\ldots,T_{i-1}$ committed before.



Compensation Spheres in Workflows

- Set of activities that must complete successfully as a whole
  - Otherwise it must be undone semantically
- Activities can be arbitrary
  - Don’t have to be realized as transactions
- Each activity in the sphere and the compensation sphere itself is associated with a compensating action
  - Could be the NULL action
- A compensating action may be an activity or (complex) business process
- If an activity fails
  - Compensating actions of all completed activities in the sphere are executed in ‘reverse’ order, or
  - Compensating action associated with the compensation sphere is executed

Compensation Spheres – Example



Phoenix Behavior

- Forward Recovery ("recover out of the ashes")
  - workflow state itself must be recoverable
    - persistent, recoverable workflow context (using DBMS)
    - reliable messaging for communicating workflow events
  - implementations of activities must be included in the recovery processing of the workflow
    - "safe" activities
    - utilizes stratified transactions
      1. generate request for program execution agent
      2. transport request from server queue to program execution agent queue
      3. read message, execute activity implementation, generate completion message
      4. transport completion message to server queue
      5. read completion message, store workflow state change
  - number of retries typically limited
    - global parameter
    - local parameter for activity or sphere

Handling Failures in WS-BPEL - Scopes

- Defines the behavior context of an activity (primary activity)
  - simple or structured (group of activities)
- Can provide the following for a (regular) activity
  - Partner links
  - (Local) data variables
  - Correlation Sets
  - Event handler(s)
  - Fault handler(s)
  - Termination handler
  - Compensation handler
    - Scope acts as a compensation sphere
- Scopes can be arbitrarily nested

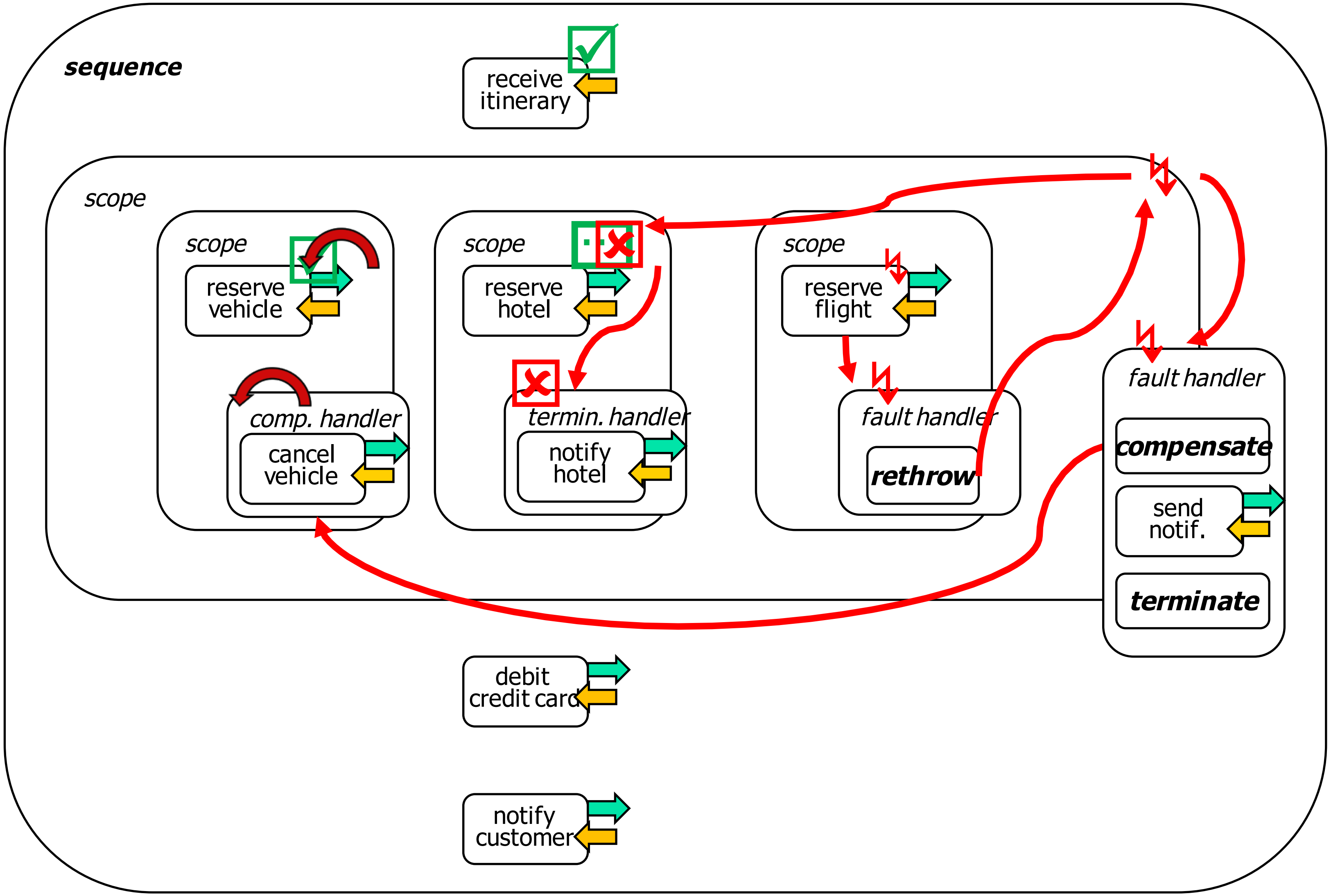
Fault Handlers and Termination Handler

- Fault handlers catch and deal with faults occurring in **active** scope
  - Can catch internal faults (throw activity), WS fault messages
  - All active work in the scope is stopped!
    - Results in invocation of termination handlers for active enclosed scopes
  - After fault handler completes successfully, processing continues outside the scope
    - Processing of the scope is still considered to have ended abnormally
- Termination handler allows to define scope-specific termination behavior
  - Invoked if an active scope needs to be terminated
    - Example: perform cleanup work, notify business partner, cancel activity
  - For nested scope: TH for inner scope is invoked before the TH of the outer

Compensation Handlers

- Compensation handlers reverse the work of a **sucessfully completed** scope
  - Compensation handler is "installed" after successful completion of the scope
  - Can be defined for each scope
  - Compensation activity can be any activity
  - Compensation handlers live in a snapshot world
    - When invoked, they see a snapshot of the variables at scope completion time
    - Cannot update "live" data variables
    - Can only affect external entities
    - Input/output parameters for compensation handler are future direction
- Compensate activity
  - Invokes compensation handler for named scope
  - Can be invoked only from the fault handler or compensation handler of the immediately enclosing scope

Fault-Termination-Compensation - Example



Default Compensation and Fault Handlers

- Default compensation handler
  - Invokes compensation handlers of immediately enclosed scopes in the reverse order of the completion of the scopes
  - Is used if a (enclosing) scope does not explicitly define a compensation handler
  - Can also be invoked explicitly
    - Useful if comp. action = “compensate enclosed scope in reverse order” + “additional activities”
- Default fault handler
  - Invokes compensation handlers of immediately enclosed scopes in the reverse order of the completion of the scopes
  - Rethrows the exception

Transaction and Retry Support in BPEL

- BPEL does NOT define support for
  - transactions, atomic spheres
  - forward recovery/retry behavior
  - interaction with or participation in business activities (see WS-BusinessActivity)
- This is provided by product-specific extension (IBM, Oracle, ...)
- Possible model for extensions (example: IBM Business Process Manager)
  - Process can be declared as
    - microflow: always runs in a single transaction
    - long-terms process: involves multiple transactions
  - Activities in long-term processes can be specified to start new transactions
  - Invocation of external sevices can happen in separate transactions
    - suspend-resume behavior
    - asynchronous/stratified transation behavior
  - Retry properties/behavior can be customized

WS-BA – Business Activities Framework

- Characteristics (see discussion in chapter on WfMS)
  - Usually long-running
    - Responding to a request may take a long time
  - May consume lots of resources, perform a lot of work
    - Early commit of atomic subactivities/transactions
    - Forward recovery, compensation
- Goal: define protocols that "wrap" proprietary business activity mechanisms to achieve interoperability
- Design points
  - State transitions need to be reliably recorded
  - All request messages are acknowledged
    - Detect problems early
  - Response to a request is a separate operation
    - Not the output of the request
    - Avoid problems with timeouts of message I/O implementations

Business Activities Model

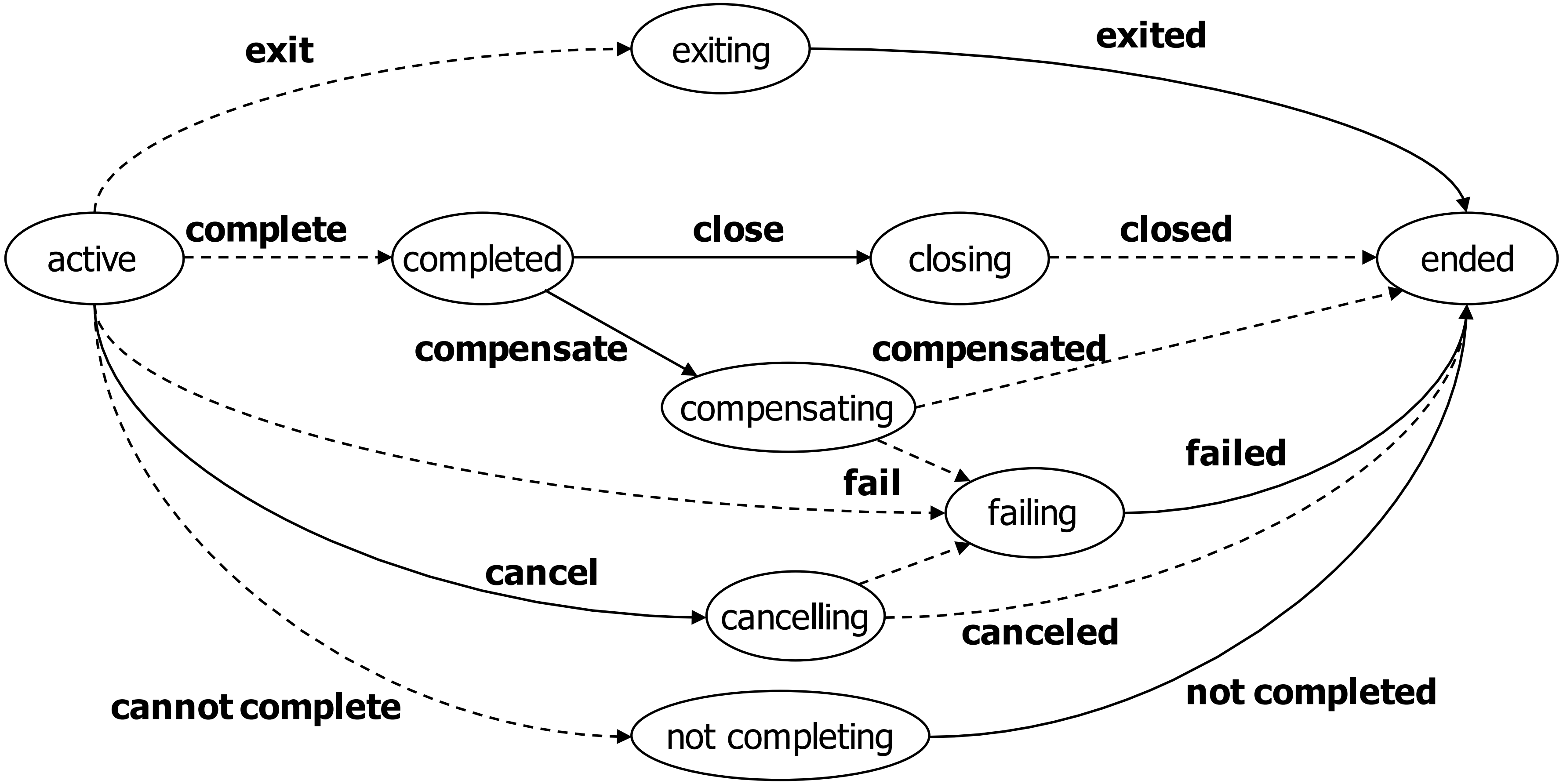
- Application is partitioned into business activity scopes
  - carries out business tasks using web services (participants)
  - mutually agreed outcome of all participants
- Participants registered with a coordinator of a BA
  - notify the coordinator about (successful) completion
  - may be asked by the coordinator to cancel an active task or to compensate a completed task
  - may indicate that it
    - cannot complete the task (and has cancelled it)
    - is leaving (exit) the BA (and has cancelled it)
    - has failed (during regular activities, when compensating or cancelling the task)
      - state of work is undetermined!
- Scopes may be arbitrarily nested

Business Activity (cont.)

- Business Activity (BA) coordination types
  - AtomicOutcome: coordinator directs all participants to either close or compensate
  - MixedOutcome: coordinator may direct some participants to close, others to compensate
- BA protocol types
  - BusinessAgreementWithParticipantCompletion protocol
    - participant must know when it has completed all the work for a business activity
  - BusinessAgreementWithCoordinatorCompletion protocol
    - participant relies on coordinator to tell it when it has received all requests for work in the business activity

Business Agreement Protocol

- BusinessAgreementWithParticipantCompletion – State Diagram



Summary

- Business (Re-)Engineering, Business Process Modeling
  - goal: efficient execution of core business processes
  - explicit specification of process models
    - focus on control flow, rudimentary data flow, organizational aspects
  - process optimization and analysis
    - static optimization
    - simulation (analytical, discrete event/based)
- Workflow Management Systems
  - middleware for management, control and execution of business processes
  - build time
    - extend process models created using BPR tools
      - data flow, control flow details, organization aspects, activity implementation
  - run time
    - work item lists, process life cycle and process management, audit trail

Summary (cont.)

- Workflows and transactions
  - ACID is too strict for long transactions
    - only appropriate for individual activities or restricted subset of activities (atomic spheres)
  - advanced transaction concepts
    - complex transaction structures
    - compensation
    - forward recovery
  - compensation spheres: sets of semantically linked transactional (sub-)activities
    - can be used in combination with atomic spheres
- Advantages compared to explicit modeling of exception/failure handling steps as part of the process model
  - Reduces complexity of the process
  - Separation of regular business logic from exception/failure handling
  - Increased flexibility
    - compensation of spheres that have completed successfully

Summary (cont.)

- Web service composition
  - means to implement web service by reusing/combining existing services
  - can be supported by WS composition middleware
    - borrowing concepts from WFMS
- BPEL
  - de-facto and de-jure (OASIS) web service composition/orchestration standard
  - allows definition of composition and coordination aspects
    - abstract vs. executable processes
  - main concepts
    - basic activities for web service operations
    - structured activities for defining service composition, control flow
    - blackboard approach for data flow based on variables
    - service selection based on partner link types, partner links, endpoints
    - elaborate model for failure and exception handling
      - fault handler, termination handler, compensation handler
- BPEL extensions:
  - people WF (BPEL4People, WS-HumanTask), Java/SQL snippets (BPELJ, BPEL/SQL)