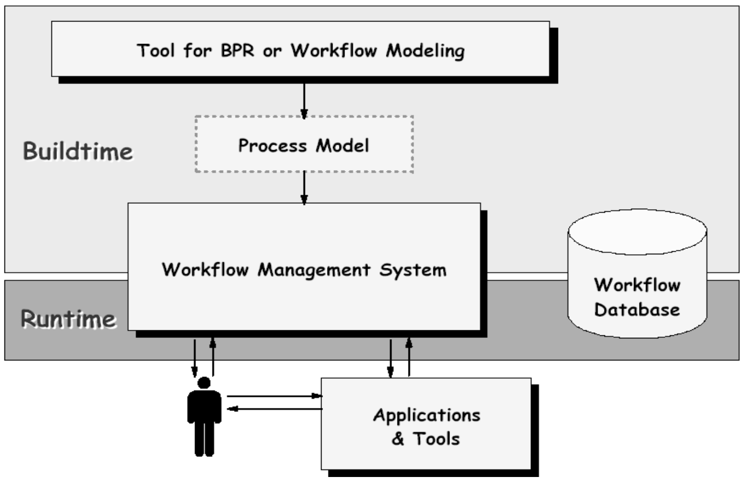
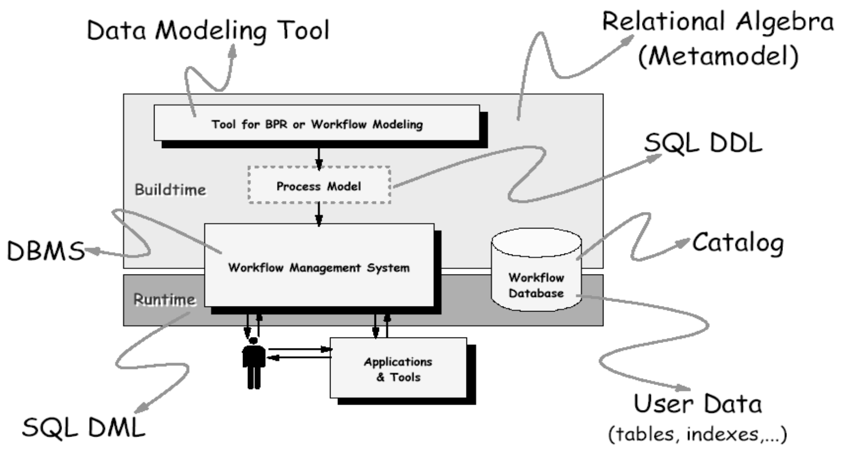
Workflow Management Systems

Workflows & Web Services
Kapitel 8

Major Building Blocks Of A WFMS



...And Their Correspondence In DBMS


Data Modeling Tool
Relational Algebra (Metamodel)
Tool for BPR or Workflow Modeling
SQL DDL
Process Model
Buildtime
DBMS
Workflow Management System
Catalog
Workflow Database
Runtime
Applications & Tools
User Data (tables, indexes,...)
SQL DML

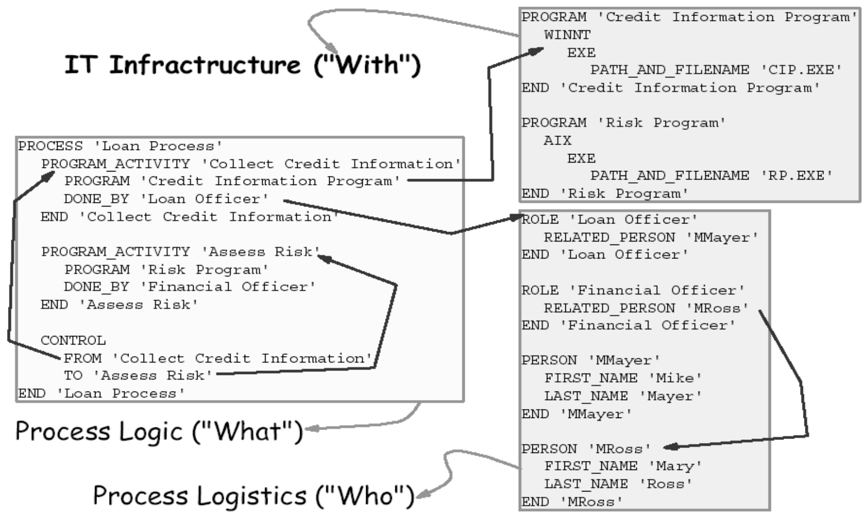
Types Of Users In Workflow Environments

- **End Users**: Perform work assigned to their worklist(s).
  - But also: Transfer work from others to them + being substitute for others
- **Process Modeler** or **business analyst**: Defines process models, organizational structure, IT structure
- **Process Administrators**: Manage running workflows
  - A person becomes ProcAdmin by
    - Explicit specification in process model or category (= grouping of process models)
    - Dynamic assignment via values associated with particular process instance
  - ProcAdmin is informed when something goes wrong with the workflow he is responsible for
    - E.g. staff resolution returns empty set, or activity implementation fails execution,...
- **Operation Administrators**: Keep the WFMS properly running
    - E.g. add resources when more users must be supported,...
- **System Administrators**: Responsible for the overall environment
- **Customer Support**: Mediate between customers & business
    - E.g. inquire state of workflows, or start, terminate,... workflows
- **External Users**: Mainly customers interacting with WFMS
  - Can replace mediation by customer support

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 are 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
  - (Work)Flow Definition Language (FDL)
    - ASCII text with special syntax/semantics
      - Most often vendor specific
      - Standard developed by Workflow Management Coalition (WfMC)
        - WPDL (Workflow Process Definition Language)
        - XPDL (XML Process Definition Language)
  - Both GUI and FDL cover all concepts of the WFMS Meta Model

Flow Definition Language: Example



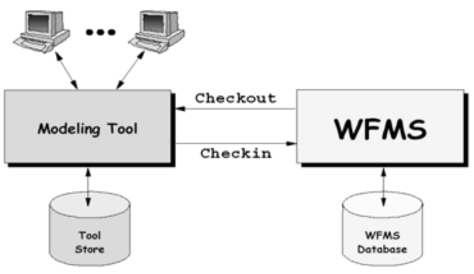
```
PROCESS 'Loan Process'
  PROGRAM_ACTIVITY 'Collect Credit Information'
    PROGRAM 'Credit Information Program'
    DONE_BY 'Loan Officer'
  END 'Collect Credit Information'

  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'
```

```
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'
```

```
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'
```

Modeling Tool Architecture

- Modeling tool can operate in connected or disconnected mode
  - In **disconnected** mode tool gets persistence via local store
    - Checkout to lock object in particular mode, move it into tool store & manipulate it there via GUI (if lock mode allows), checkin to workflow system & release locks
  - In **connected** mode the workflow database is updated directly
    - Some tools have no tool store at all, i.e. can only operate in connected mode
    - Some operations can be heavy duty with impacts on concurrency, throughput,...
- Tool store can be separate database shared by multiple tools
  - Tool clients may have their own local store, work connected or disconnected
    - Additional level of checkout/checkin by tool (client) from shared database



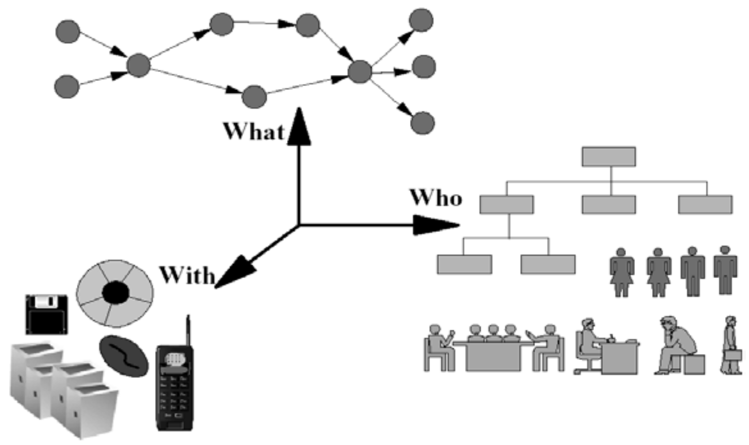
Modeling API

- Only WFMS engine has direct access to the WFMS database
  - Often, not even the buildtime component of the WFMS may access the database!
  - Risk of damaging database is too high
    - WFMS engine must be high available - Corrupted DBMS will produce outage!
- WFMS provides an API to access its database
  - Typically, this API supports batch mode and interactive mode
  - In batch mode data is exchanged via FDL format
  - import/export, check-in/check-out
    - FDL may be exchanged as file or as memory buffer
      - additional keywords for indicating create, insert, replace, delete operations, locking, transaction demarcation
    - Typically, mass of data are exchanged in batch mode
  - For interactive mode CRUD methods for all metamodel elements are provided
    - Used for manipulating selective objects
      - E.g. modifying the properties of a person like when s/he becomes absent
    - Additional methods provided for explicit locking/unlocking of higher-level constructs and explicit process model correctness/completeness checking

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 (ie. "time interval" versioning)
- Once put into production, instances can be made from a model

The Three Dimensions Of Workflow



“Who” Dimension: Organization Metamodel

- WFMS can support fixed or dynamic organization meta-model
  - **Fixed** org meta-model does not allow to change the entity and relationship types supported to model organizations
    - Org. metamodel is built-in by the WFMS vendor
    - Can be implemented efficiently
    - Simple org. metamodel (Person, Department, Role,..., Managed_By, Substitute_Of) often sufficient
  - **Dynamic** org. meta-model allows to change the entities and relationships of the built-in meta-model, or even to create a complete new meta-model
    - Very flexible, but hard to achieve
      - efficiency
      - schema versioning
    - Requires WFMS to dynamically
      - translate modified org. meta-model to an underlying DBMS schema
      - translate staff queries over org. meta-model to queries over org. database

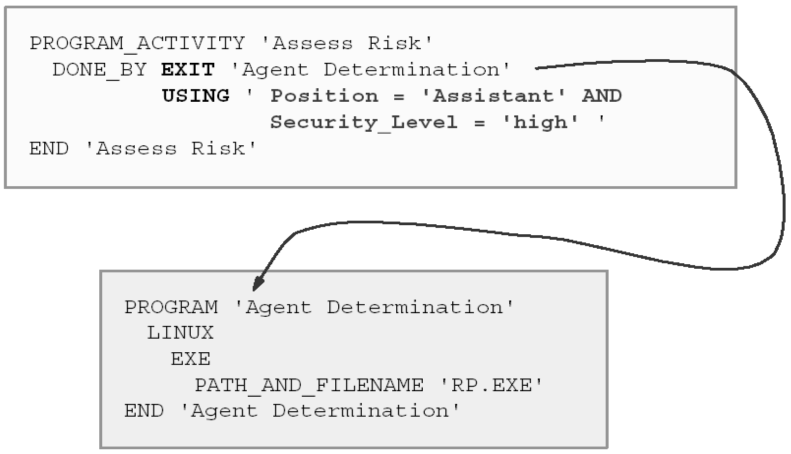
Where Organizational Data Is Managed

- WFMS manages org. data in its database
  - Pro: Database schema is optimized for access by WFMS
  - Data might be replica of "real" org. database (often the case!)
    - Pro: WFMS does not influence performance of source system and vice versa
    - Pro: WFMS might be a distributed system; replica allow local access, no access to central org database required (efficiency, availability is the issue)
    - Con: Data might run out of sync
- WFMS shares org. data with other systems, and each of the systems can modify the data
    - Ideal when holding org. data in a directory (LDAP, X.500,...) or HR system
    - Pro: No redundant data
    - Con: Performance
    - Con: Org metamodel of directory very likely different from that of WFMS
      Thus, dynamic mapping of org metamodels required at runtime (at build time only, if org data is replica!)
- WFMS has read only access to the org data in another system
    - Same pros and cons as before

Performing Staff Queries

- When org data is managed by WFMS it can execute staff queries directly on its own internal database
- When org data is not managed by WFMS it must run each staff query against the external org data system...
  - Using a staff resolution exit
    - When WFMS must retrieve agents it simply invokes a user provided program
    - This program can perform any kind of computation but must return a set of agents
    - Parameter of the exit can be a query supported by the external org data system
  - By mapping the WFMS org metamodel onto the external metamodel
    - Problem: Can the metamodels be mapped at all without loosing too much semantics?
    - If metamodels can be mapped a tool is needed to transform each staff query formulated in terms of the WFMS metamodel into the external query language
  - By directly using the external system's database
    - Can be done if
      - WFMS and external system use the same type of DBMS (e.g. relational)
      - WFMS metamodel is a "subset" of the external metamodel (e.g. as views on external tables)

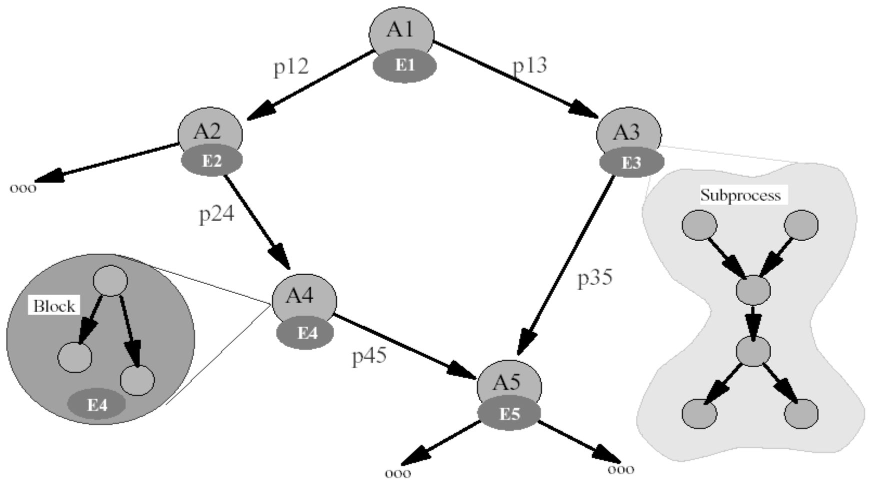
Sample: Staff Query Via Exit



```
PROGRAM_ACTIVITY 'Assess Risk'
  DONE_BY EXIT 'Agent Determination'
          USING ' Position = 'Assistant' AND
                  Security_Level = 'high' '
END 'Assess Risk'
```

```
PROGRAM 'Agent Determination'
  LINUX
    EXE
      PATH_AND_FILENAME 'RP.EXE'
END 'Agent Determination'
```

“What” Dimension: Control Flow Specification



Subprocesses

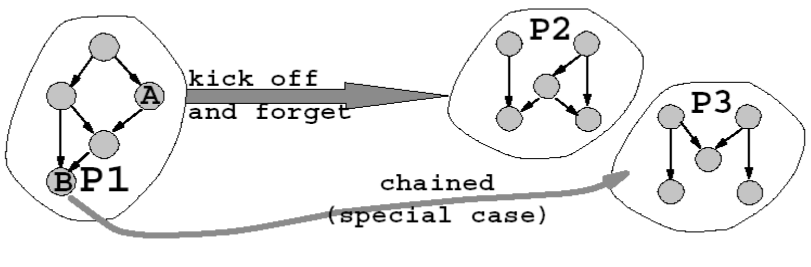
- An activity implementation may be another process, called subprocess from the point of view of the process that owns the implementing activity (so called parent process)
- A subprocess is called
  - local if it is performed by the same WFMS that runs the parent process
  - remote otherwise
- The WFMS running the remote subprocess can be...
  - ...from the same vendor
    - Private vendor-specific FAPs (Formats And Protocols) can be used for communication (e.g. parameter passing, state exchange, monitoring data,...)
  - ...from a different vendor
    - FAPs must be standardized (e.g. via WfMC) or negotiated between vendors
      - Much more cumbersome than in "homogeneous" environment

Autonomy Of Subprocesses

- A subprocess is a process in its own rights
  - It is derived from a complete and correct process model that has been defined independently
  - Especially, the model of the subprocess can be instantiated alone (i.e. without being invoked by some parent process) resulting in a "standalone" workflow
  - Even as subprocess the workflow runs to a certain degree "independent" from the parent process
- The degree of independence is governed by autonomy rules
- Autonomy rule defines the rights of a parent on a subprocess
  - Completely autonomous: Once kicked-off the parent cannot influence the execution of the subprocess
    - E.g. termination of the parent does not terminate the subprocess
  - Totally controlled: The life-cycle of the subprocess is determined by the parent process, e.g.
    - Suspension of the implemented activity forces the subprocess to suspend
    - The process administrator of the subprocess must the administrator of the parent
  - Whole spectrum between these extremes can be defined

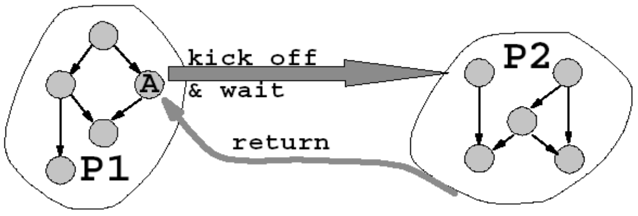
Spawned Subprocesses

- Activity A of workflow P1 causes another workflow P2 to start
- P1 and P2 operate completely independent from each other, e.g.
  - A can complete without having P2 complete
  - P2 might terminate abnormally without affecting P1
- P3 is started when P1 completes (i.e. the end-activity B is implemented by P3)
  - P3 is called "chained": Special case of a spawned subprocess
- Chained workflows are often used



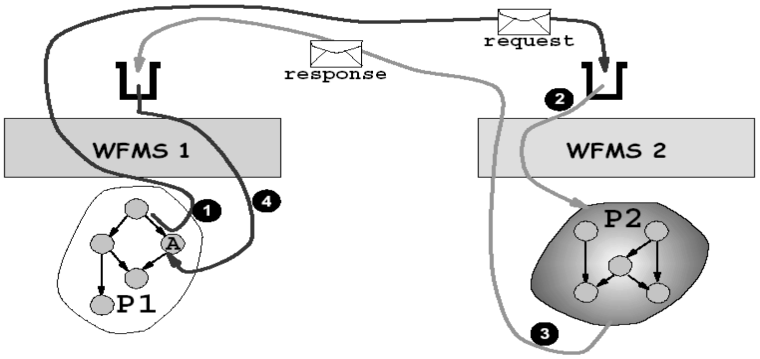
Nested Subprocesses

- Activity A of workflow P1 causes another workflow P2 to start
- P1 and P2 are dependent on each other, e.g.
  - A completes if and only if P2 completes and returns
  - Termination of P1 or A causes P2 to terminate
- A whole hierarchy of nested subprocesses can be defined, i.e. P2 might have nested subprocesses etc.



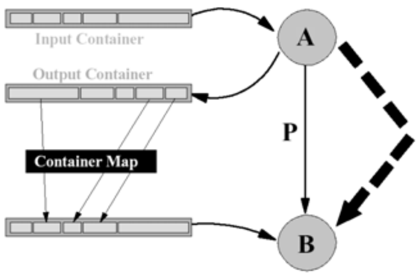
Remote Nested Subprocess

- Request/response messages could be exchanged via message queuing, e-mail etc..
- Exchange mechanism determines properties like guaranteed delivery (MQ), ability for "ad hoc" bindings between WFMSs etc..

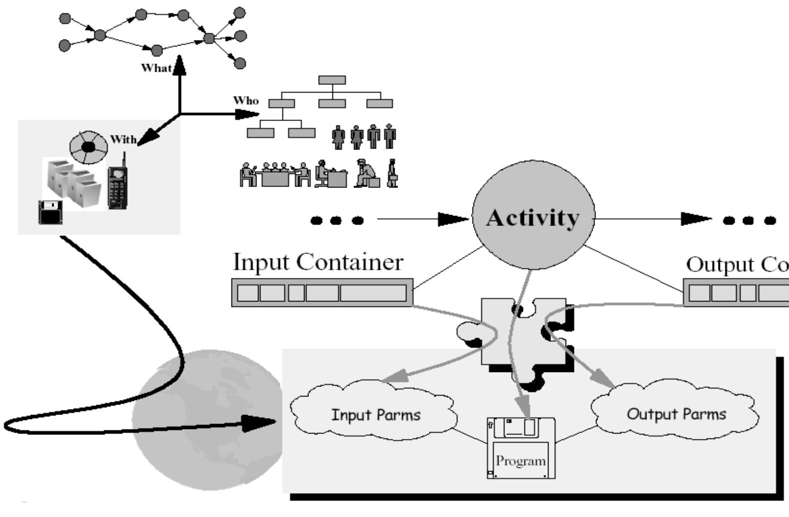


"What" Dimension: Data Flow

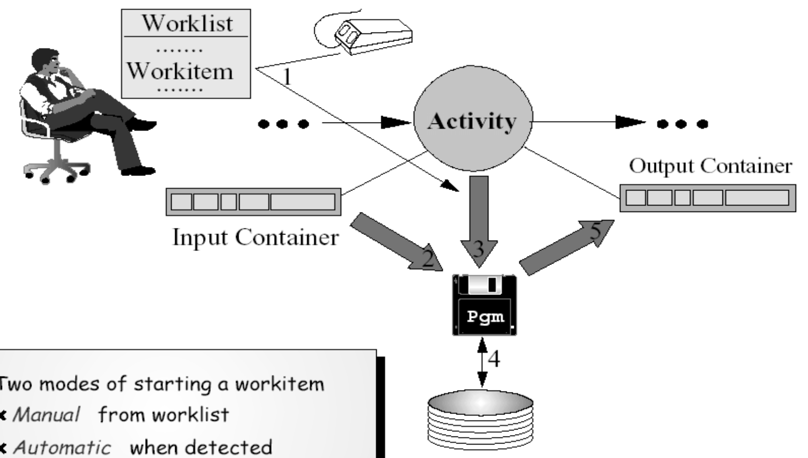
- Input/Output Container
  - defines data passed to/returned by process or activity
    - based on simple/structured types
      - definitions can be shared
    - can also specify default values
  - provides the execution context
- Data Connectors
  - specify which data needs to be copied where
  - detailes 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
  - de-materializes output container instance (makes it persistent)



"With" Dimension: Program Registration



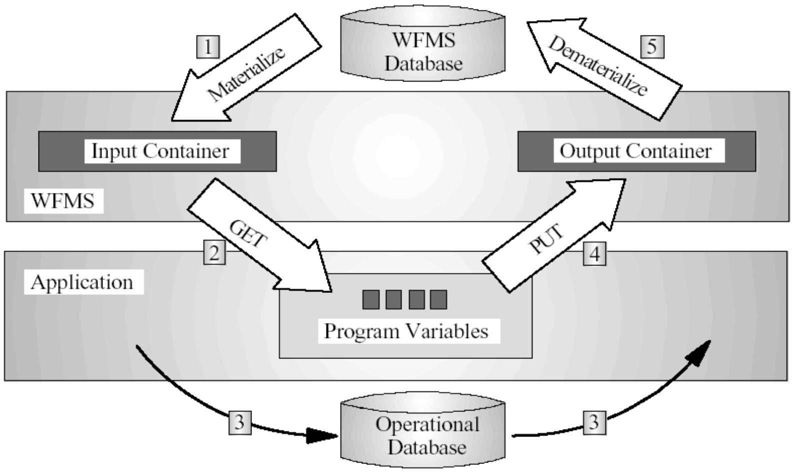
Invoking Activity Implementations



Two modes of starting a workitem

- *Manual* from worklist
- *Automatic* when detected

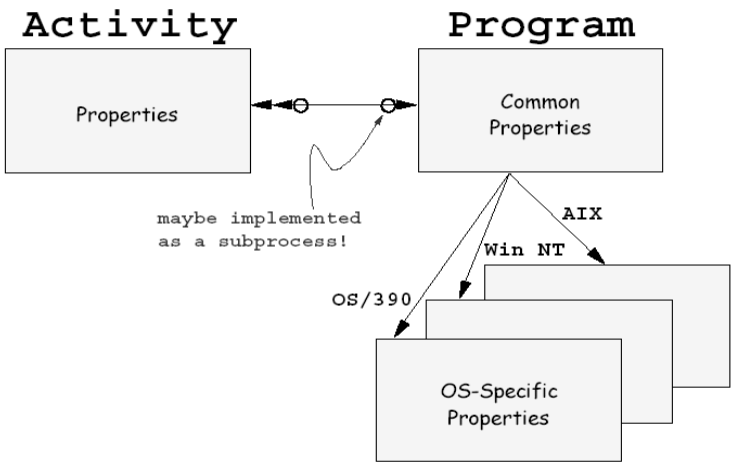
Processing Containers



Decoupling Activities And Their Implementation

- Business modeller want to focus on process models
  - Thus, allowing to specify programs separately and link them to activities separates between activities as conceptual constructs and programs as implementation constructs
- Programs depend on the environment they are running in
  - In general, their signatures depend on the environment
    - Mapping from container to signature must be specified: "Data Mapping Language"
  - Each environment has its own environment parameters and formats
- Programs should be able to be exchanged without requiring to modify process models ("late binding")
  - WFMS resolves actual program to call when activity implementation must be invoked
  - Of course, "early binding" is supported too

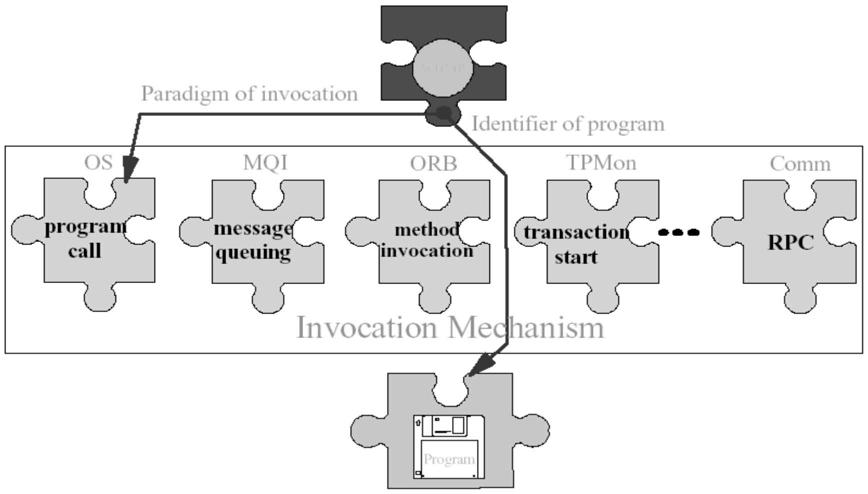
Program <-> Activity Relationship



The Relationship in FDL

```
PROGRAM_ACTIVITY Collect Information
  PROGRAM CollectInfo
END Collect Information

PROGRAM CollectInfo
  WINNT
      DLL
        PATH_AND_FILENAME d:\pgm\ci.dll
  AIX
      EXE
        PATH_AND_FILENAME infocoll.exe
END CollectInfo
```

Program Invocation: Metadata

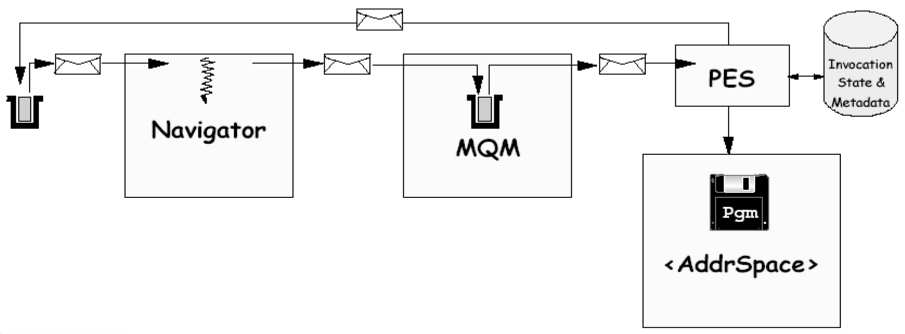


Sample Metadata

- Program call
  - Mechanism to invoke EXE, DLL, CMD files on workstations
  - Requires the name of the program to call
- Message queuing
  - Asynchronous protocol
  - Requires the name of the queue to send the invocation message to
  - Requires the name of the response queue where the reply is expected
  - WFMS continuous navigation iff reply is received
- Method invocation
  - Mechanism to invoke remote objects
  - Requires the identity of the object and the method name to invoke
  - Requires the signature of the method to map container onto
- TP Monitors
  - Requires the transaction identifier
  - Requires to map between containers and "wire format" of transaction

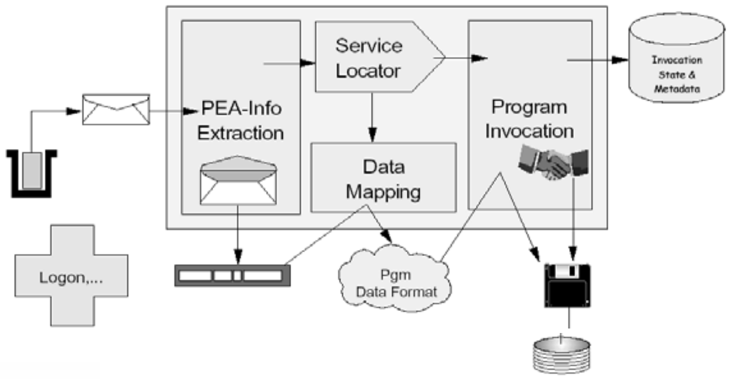
Internal Component Flow

- WFMS Navigator determines program to be executed
- "Execution Messages" sent to launching component called Program Execution Server (PES)
- "Completion Message" sent back to Navigator when invoked program returns



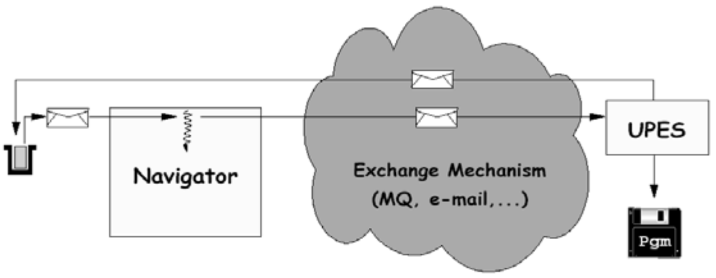
Program Execution Server

- WFMS won't be able to support all mechanisms to launch executables
- Thus, users should be able to build their own PESes, e.g. WFMS
  - provides interfaces between PES building blocks
  - defines required messages exchanged between navigator and PES (User-provided PES (UPES))
- Specific metadata are needed by PES, e.g. security information, mapping prescriptions etc.

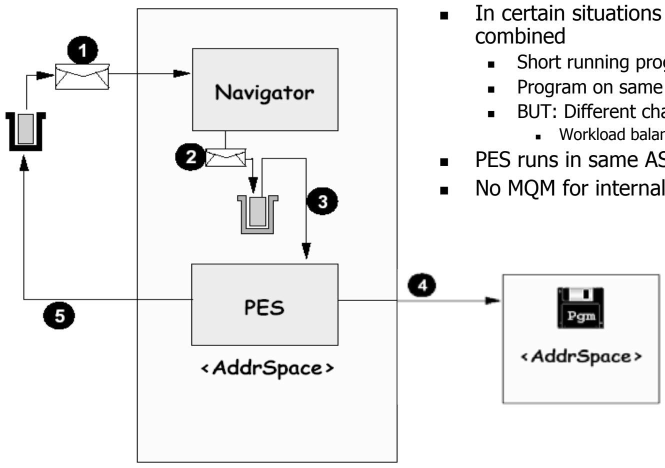


User-Provided PES (UPES)

- UPES has to commit certain quality of services like the corresponding PES provided by the WFMS vendor
  - For example, exactly once invocation for safe activities (see later)
- Otherwise, UPES can be any kind of implementation

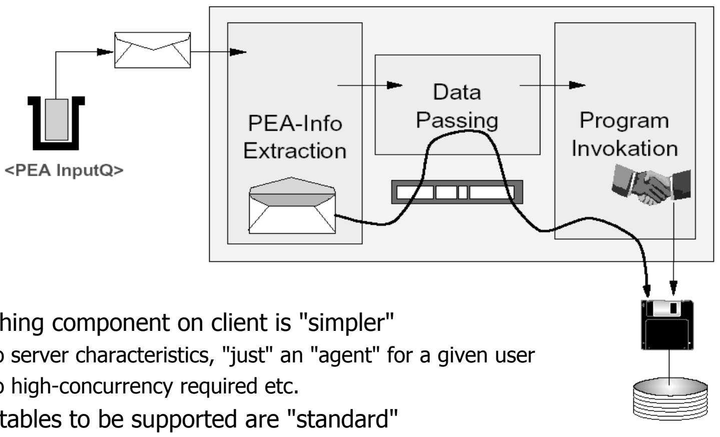


Fast Path Invocation



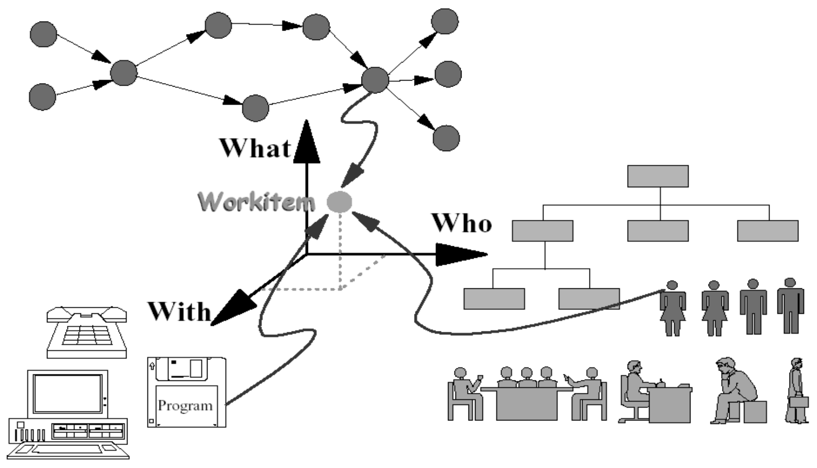
- In certain situations PES and Navigator can be combined
  - Short running programs
  - Program on same machine as navigator
  - BUT: Different characteristics
    - Workload balancing (see later!)
- PES runs in same AS as navigator
- No MQM for internal communication

Program Execution Agent



- Launching component on client is "simpler"
  - No server characteristics, "just" an "agent" for a given user
  - No high-concurrency required etc.
- Executables to be supported are "standard"
  - EXE, DLL,... with "straightforward" data mapping

The Three Dimensions Of Workflow



Worklists

- ... a launchpad for business functions
  - Filtered list of workitems per agent
  - Automatic prioritization of work
  - Associates tools to pieces of work
  - Automatic data provision
  - . . .
- User gains focus on business aspects of work instead of computer aspects

Defining Worklists

- A worklist is a 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
    - More specific, a group worklist associated with a role is called role worklist
  - All users belonging to the group associated with the group worklist can pick a workitem from that list

Defining Worklists: Example

```
WORKLIST LoanProcesses
   TYPE PUBLIC
   VIEW
     WHERE PROCESS_NAME = LoanProcess
   ORDER BY Priority
   REFRESH_MODE PULL
END LoanProcesses

WORKLIST ImportantLoanProcesses
   TYPE PUBLIC
   VIEW
     WHERE PROCESS_NAME = LoanProcess
     AND PROCESS_STARTER = MY_MANAGER
   ORDER BY Priority
   REFRESH_MODE PUSH
END LoanProcesses
```

Modes Of Obtaining Workitems

- Pull
  - User must explicitly request to refresh a particular worklist
    - Get new workitems, remove workitems started by other users
  - Content of worklist does not change without request
    - In high throughput environments certain worklists might be in constant flux!
      - Users might be disturbed, confused,... thus less productive
- Push
  - New workitems are immediately put onto the corresponding worklist(s)
  - Workitems started by others are immediately removed
  - Push worklists are always up to date!
- Grab
  - Whenever a user needs work the WFMS delivers a matching workitem
    - Explicitly on request ("get next workitem")
    - Automatically when current workitem completed successfully
      - On completion of a workitem a next workitem is automatically started
  - Especially convenient for group worklists

Deadlines

- Most processes must be performed in a certain time
  - E.g. for legal reasons or to meet company specific quality goals
- To support this, the WFMS allows to specify...
  - ...time limits at both, the process model level and the activity level
  - ...actions that should happen when a time limit is exceeded
    - Typical action is to notify somebody who has to take corrective actions
      - This facility is called "notification"
  - The processing of deadlines is called "escalation"
    - Deadlines can also be specified for actions associated with escalations
    - Escalations are escalated via notifying the process administrator
- The time measured for detecting out-of-line situations can be
  - ...the absolute time passed since the beginning of the situation to be monitored ("soccer semantics")
    - Time since activity has been schedule, arrived on worklist, started to be worked on,...
  - ...the time passed on working on the activity or process to be monitored ("base ball semantics")

Deadlines: Example

```
PROGRAM_ACTIVITY RequestApproval
  DURATION 2 DAYS
   WHEN EXCEEDED NOTIFICATION TO MANAGER
   SECOND NOTIFICATION AFTER 10 DAYS
          [or FORCE TERMINATE
          or ...]
END RequestApproval
```

Speedup Of Business Processes

- Parallelism in workflows
  - Parallel branches of process can be worked on in parallel
- Descriptive staff assignment
  - Query-based determination of responsible staff at run-time instead of fixed persons associated at buildtime
  - With number of qualified people receiving workitem likelihood of earlier execution increases
- Notification processing
  - Exceeding maximal duration is signaled to allow bottleneck detection
  - Transfer of workitems in overload situations
- Substitution principle
  - Work for absent people is routed to substitutes

Managing Errors

- A large number of errors can occurs while a workflow is running
  - Activity implementation cannot be located, or it returns wrong data in its output container (e.g. wrong type), or a resolved user is not authorized to execute it etc.
- WFMS supports default actions to cope with such situations
  - Put the activity into the state InError
  - Inform the process administrator to correct the situation
- Sometimes, default actions must be overridden and more specific actions must be taken
  - Both, at the process level or at the activity level

Managing Errors: Example

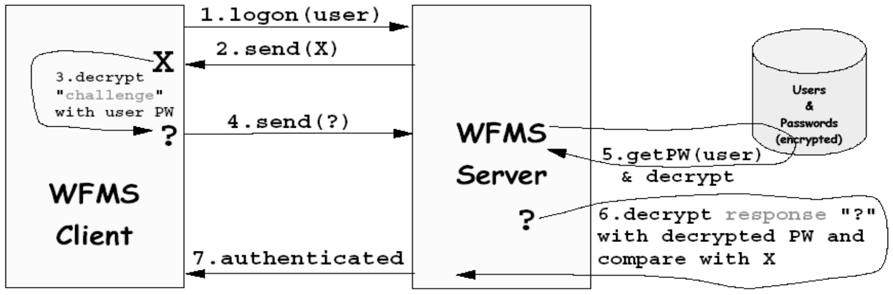
```
PROGRAM_ACTIVITY RequestApproval
  ON_ERROR
   WHEN_PROGRAM_NOT_FOUND
          NOTIFICATION TO OPERATOR
   WHEN_USER_NOT_AUTHORIZED
          NOTIFICATION TO SEC_OFFICER
END RequestApproval
```

User Session

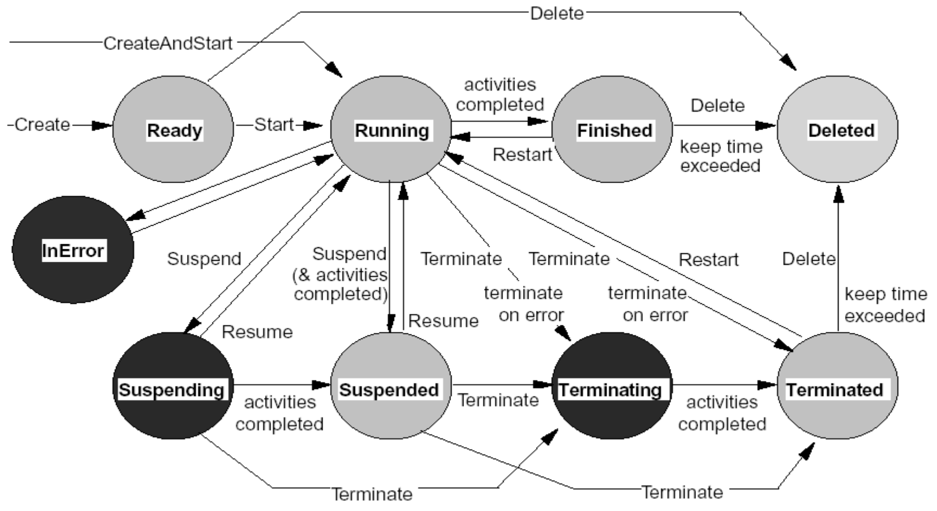
- To work with a WFMS a user has to establish a session
- Session is initiated by starting appropriate client component of the WFMS and by providing user_id and password
- Within a session WFMS assumes that all requests come from the user identified before via user_id and password
- Session is ended...
  - ...when user explicitly terminates the session
  - ...automatically when user was inactive for a predefined period of time
    - used to avoid unnecessary resource consumption
    - reduces risk of unauthorized access if user forgets to terminate the session

Authentication

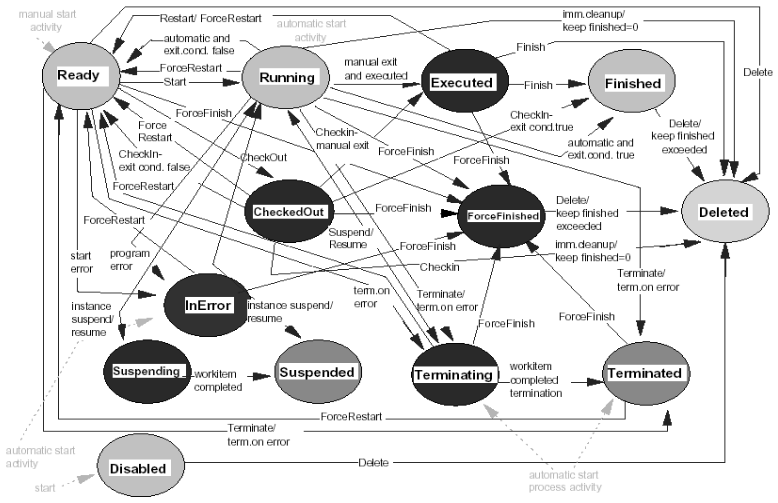
- How to logon without sending passwords?
  - Plain text password could be read by wire tapping
  - Secure encryption expensive
- Variants of "Challenge/Response" mechanisms possible
  - E.g. messages exchanged can be encrypted or plain or...


1.logon(user)
2.send(X)
X
3.decrypt "challenge" with user PW
?
4.send(?)
WFMS Client
WFMS Server
Users & Passwords (encrypted)
5.getPW(user) & decrypt
?
6.decrypt response "?" with decrypted PW and compare with X
7.authenticated

The Life Of A Process



The Life Of An Activity And Workitem

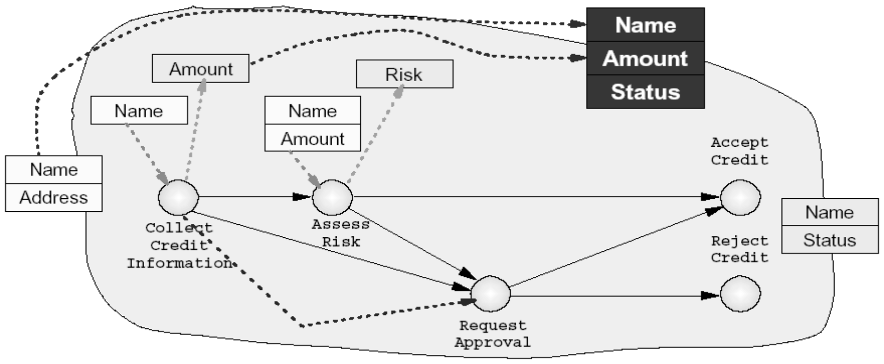


Process Queries

- Purpose is to locate a particular process or set of processes
- Two different kinds of selection criteria
  - Operational, e.g. start date, state,... of a process
    - Often used by process administrators
  - Business, e.g. name of customer, order value,...
    - Used by business people, e.g. call center personnel
- Queries return process identifier, especially
  - Process identifier can be used to start process monitor or to retrieve other data about the process
    - Enables combination with prediction capabilities, e.g. time to finish a given process
  - Detailed execution history inquired by accessing the audit trail

Key Container

- Process model may have a key container assigned. Key container can be filled by usual data flow mechanisms. It can be used to locate process instances via queries: No knowledge of process identifier needed!



Audit Trail: Structure

- All important events in the life of a process can be recorded as an entry in the audit trail
- Sample events:
  - Creation, start & termination of a process
  - start, termination, restart & completion of a WI
  - transfer of a WI
  - ...
- Sample fields of an audit trail entry
  - Date & Time the event took place
  - The name/identifier of the event itself
  - Requester of the action (e.g. a certain user or the WFMS itself)
  - Name/identifier of the associated activity, process model
- Key container stored in audit trail to allow for better analysis of execution histories.

Audit Trail: Purpose

- Sample usages of the audit trail
  - Laws require to maintain the life cycle of certain business processes
    - The life cycle will be audited on demand
    - Audit records must often be kept for many years
      - E.g. in airline industry for 30 years
  - Process reengineers want to derive statistical data about processes
    - Average durations of processes or activities
    - Paths taken through process models
  - Audit trail might become extremely huge!
    - WFMS must allow to specify which data is written to the audit trail
      - Influence on amount of data:
        - Fields to record for each event
        - Events to record (e.g. only start and completion, not terminations and restarts)
    - Archiving/Restore functions must be provided
- In distributed environments merge facilities must be provide to consolidate specific audit records from different locations
  - E.g. all records for a specific process model, involving a particular user,...

Monitoring Process Collections

- Notification is appropriate if out-of-line situations occur infrequently
  - Otherwise, people get swamped by notifications!
- Aggregated monitoring functions try to avoid individual out-of-line situations
  - The execution of (definable) groups of processes is monitored
    - Snapshots are taken to trace and graphically represent
      - the workload generated and processed by individuals as well as groups of users
  - Thresholds can be defined in terms of workloads and actions that have to take place when thresholds are exceeded

Leitstand

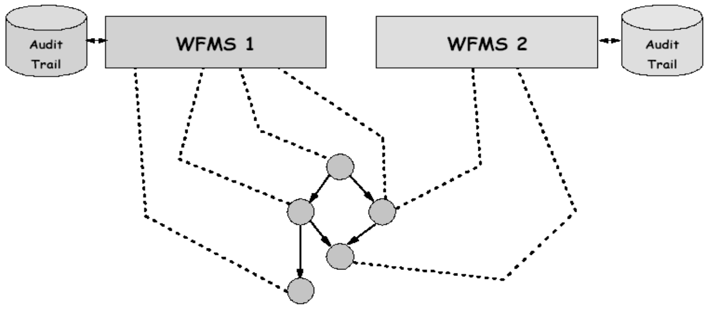
- Leitstand reports for groups of instances of a particular process model the...
  - State of each activity within each instance of the group
  - Number of current instances within the group
  - Min/Max/Average number of
    - processing time of each activity
    - number of corresponding workitems,...
- Based on defined thresholds...
  - Results are depicted in a color code
  - Actions take place
    - like notification to process administrator
      [instead of individual notifications!]
- Leitstand reports worklists of users and groups of users
  - Administrator can reassign work from places where work piles up to places where not enough work is available

Process Repair

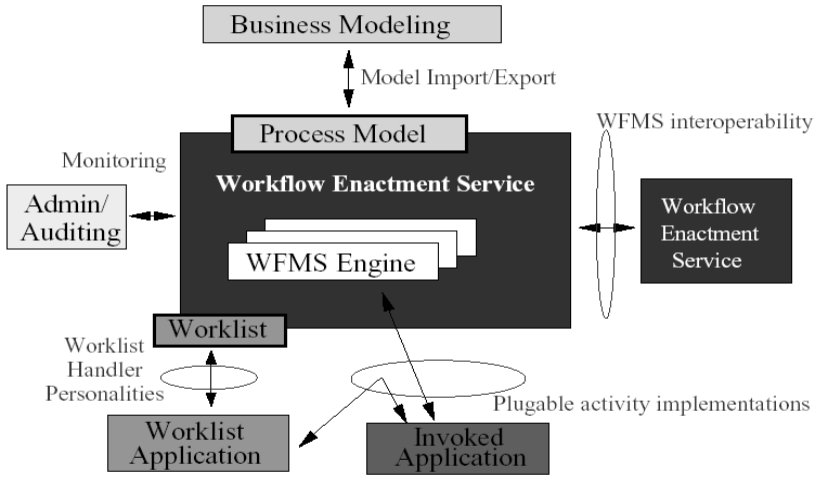
- Administrator get nofications about erroneous situations
- WFMS must provide functions to fix such situations, e.g.:
  - Input and output containers must be updatable from the outside
    - E.g.: An activity implementation ABENDed because of incorrect input due to data mapping from incorrect values in an output container. The administrator can manually correct the input data of the ABENDed activity.
    - E.g.: An activity implementation returned incorrect output and the WFMS cannot continue processing (like evaluating the exit condition, performing data mapping,...). The admininstrator can manually correct the output data.
  - The state of an activity must be updatable from the outside
    - The administrator can force restart an activity (e.g. after repairing its input)
    - The administrator can force finish an activity (e.g. after repairing its output and navigation continues with the manually provided data)
  - The implementation an activity must be exchangeable for all running instances of a process model
    - The implementation might not be locateable, i.e. this is an error applicable to all running instances
      - corrective actions on a per instance base is not sufficient

Distributed Workflows

- Distributed workflow involves agents of multiple WFMS (WFMSs might be from same of different vendors, at same or different locations, within same or different companies)
- In case of different vendor WFMSs, remote subprocesses are used for implementing distributed workflows; based on same WFMS vendor within same company distributed workitems are realized.



The WfMC Reference Model



Business Modeling
Model Import/Export
Process Model
WFMS interoperability
Monitoring
Workflow Enactment Service
Admin/ Auditing
WFMS Engine
Workflow Enactment Service
Worklist
Worklist Handler Personalities
Plugable activity implementations
Worklist Application
Invoked Application