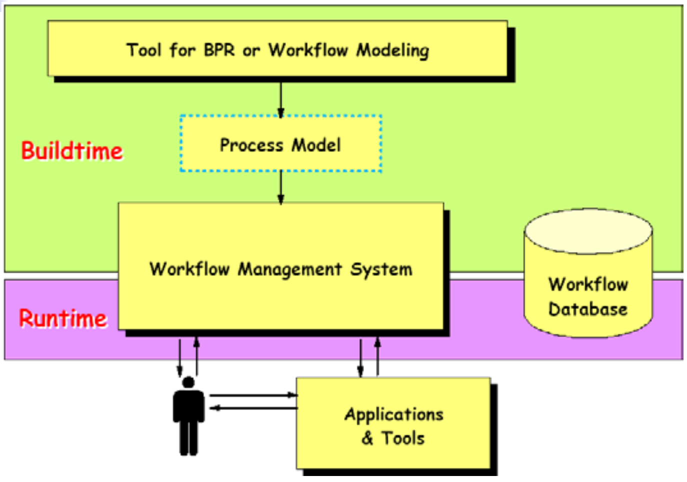
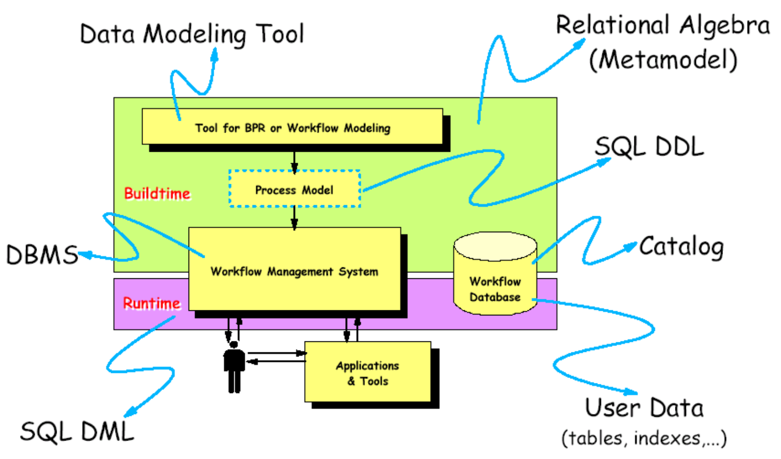
Workflow Management System Basics

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
Buildtime
Process Model
DBMS
Catalog
Workflow Management System
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

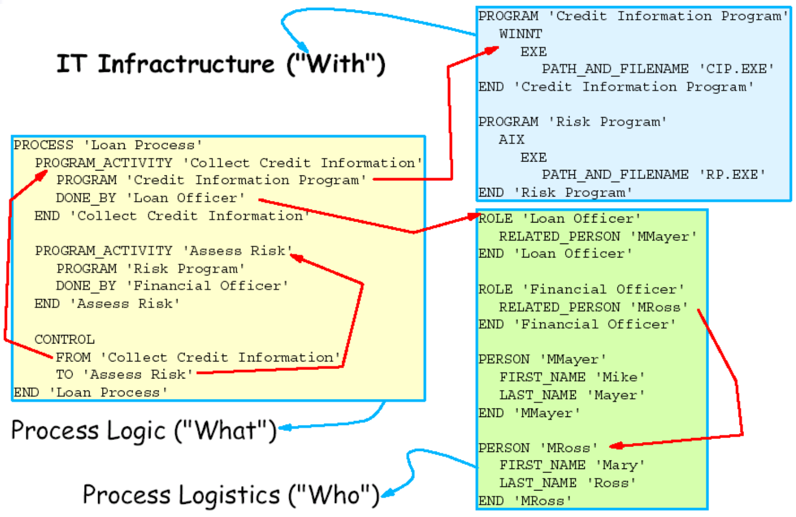
Buildtime

- 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 proposed but no widespread implementation
        - Workflow Management Coalition (WfMC) - official standard called WPDL (Workflow Process Definition Language)
      - Object Management Group (OMG) - work in progress

Flow Definition Language: Generics

- Scripting language to define workflows to WFMS
- Especially used for exchange between...
  - BPR tool and WFMS, WFMS of different vendors, WFMS of same vendor on different machines
- Language covers all constructs of the WFMS metamodel
  - Separate keyword for each metamodel construct
    - E.g. process, activity, control flow, container, person,...
  - Instances of a metamodel construct must have unique name/identifier
  - Definition of an instance begins with corresponding keyword and is ended by END keyword followed by unique name of what has been defined
  - Properties of instances specified via associated parameters or via additional keyword clauses within defining clause (i.e. proper nesting)
- Keywords, parameters and values all specified as strings
  - Thus, simple text editor can be used to create such a script (e.g. ".FDL file").

Flow Definition Language: Example

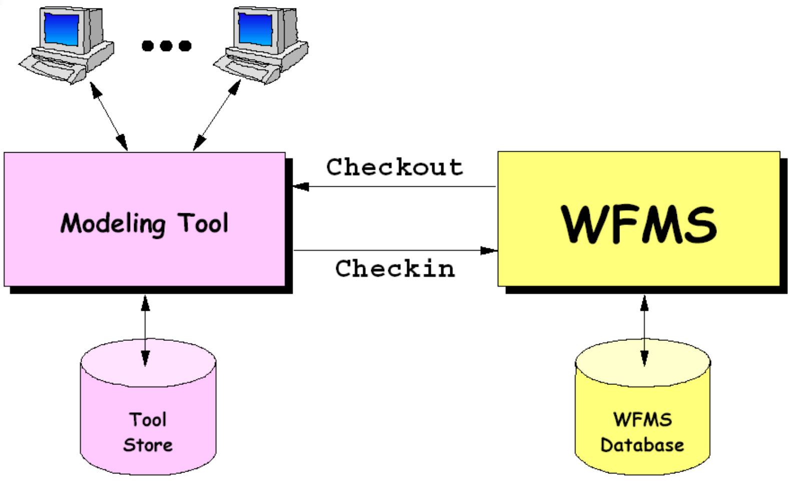


```
PROCESS 'Loan Process'
  PROGRAM_ACTIVITY 'Collect Credit Information'
    PROGRAM 'Credit Information Program'
    DONE_BY 'Loan Officer'
  END 'Collect Credit Information'

  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'
```

Support Of Modeling Tools



Modeling Tool Architecture

- Modeling tool can operate 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
  - Concurrent modeling on different metamodel instances of same model must be supported by appropriate granularity of checkout/checkin
  - Reconciliation might be required (e.g. changes of containers might impact data flow)
  - By separating modeling work and runtime of WFMS no mutual impact occurs
  - Modeling staff (e.g. consultants) prefer disconnected mode!
- 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
    - E.g. browser based tools
  - Some operations can be heavy duty with impacts on concurrency, throughput,...
    - E.g. staff resolution in simulation runs impact runtime that accesses same data
    - Can be weakened by separating test and production systems
- 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
    - FDL may be exchanged as file or as memory buffer
    - 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

FACHBEREICH INFORMATIK

Interactive Mode

- Often, object-oriented interface is provided
  - Each metamodel element is represented by an object
  - C++ sample:

```
APIRET rc;
FmcjModelingService service;
FmcjPerson person;
rc = service.Logon("userid", "password");
person.FirstName(“Mike");
person.LastName(“Mayer");
rc = service.CreatePerson(person);
```

- Methods to explicitly acquire and release locks on objects
  - Used for concurrency control
  - Checkin/Checkout might use lock/unlock on behalf of user
- Verify: Checks instances for completeness and correctness
  - Enables buildtime tool to defer checking
    - Allowing temporary inconsistencies is a must in modeling efforts
    - List of inconsistencies is returned and must be corrected before checkin

FACHBEREICH INFORMATIK

Batch Mode

- **Export**: Extracts selective model information as FDL file
  - Which information to extract is passed as query parameter, e.g.
    SELECT PROCESS WHERE CATEGORY = "CONFIDENTIAL"
  - Optional, all constructs exported can be locked ("checkout")
- **Import**: Modifies the constructs in model database as specified in the FDL file
  - Action to take place precedes each construct in FDL file to be imported
    - **CREATE**: Adds a new construct to the database (must not exist in DB - error if already exists) - default action!
    - **INSERT**: Existing construct will be updated, otherwise new one created
    - **REPLACE**: Completely replace existing construct; error if not exist
    - **DELETE**: Removes construct from the database
  - Similar actions on relationships between constructs
  - Each construct uniquely identified in database and FDL file

Batch Mode (cont.)

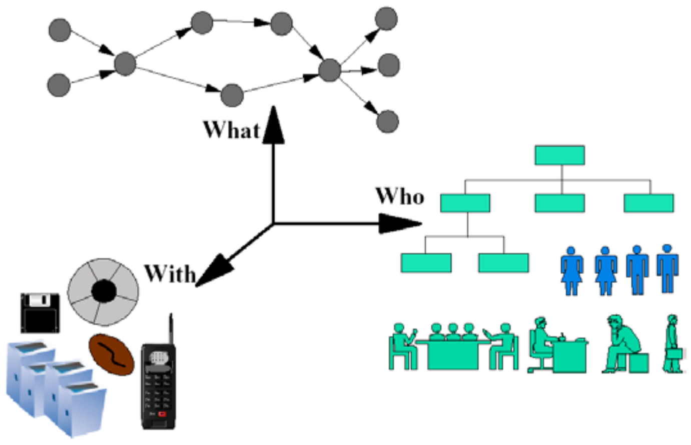
- Complete set of constructs and their relationships contained in FDL file is manipulated in single transaction, by default
- FDL file can be partitioned into collections each of which are committed indiviually
  - Transaction demarcation via BEGIN and COMMIT keywords
  - Improves concurrency with runtime when large FDL files are to be imported
    - E.g. mass updates of large organization
- CHECKIN is combination of IMPORT and UNLOCK
- CHECKOUT is combination of EXPORT and LOCK

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

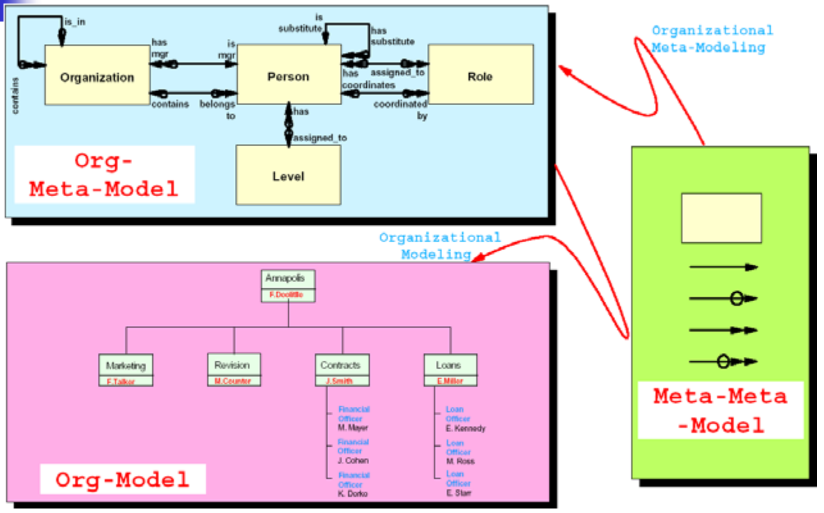
The Three Dimensions Of Workflow

FACHBEREICH INFORMATIK



Intension/Extension Pairs

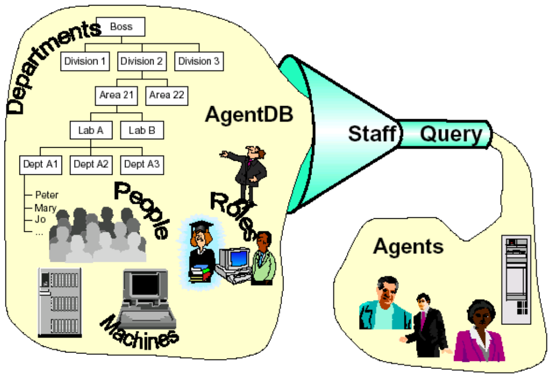
FACHBEREICH INFORMATIK



"Who" Dimension: Organization Metamodel

- WFMS can support fixed or dynamic organization metamodel
  - Fixed org metamodel does not allow to change the entities and relationships supported to model organizations
    - Org metamodel is build-in by the WFMS vendor
    - Can be implemented efficiently
    - Simple org metamodel (Person, Department, Role,..., Managed_By, Substitute_Of) often sufficient
  - Dynamic org metamodel allows to change the entities and relationships of the build-in metamodel, or even to create a complete new metamodel
    - Very flexible, but efficiently hard to achieve
- The language to specify org metamodels (i.e. the "meta-metamodel") can be any data modeling language
  - Entity/Relationship model, object-oriented data model,...

Staff Resolution



When org metamodel can be changed by users, the WFMS must provide a means to transform org metamodel into a schema of the underlying DBMS and to transform staff queries (formulated in terms of org metamodel!) into query on the resulting database schema - efficiency is the issue as well as schema versioning!!!

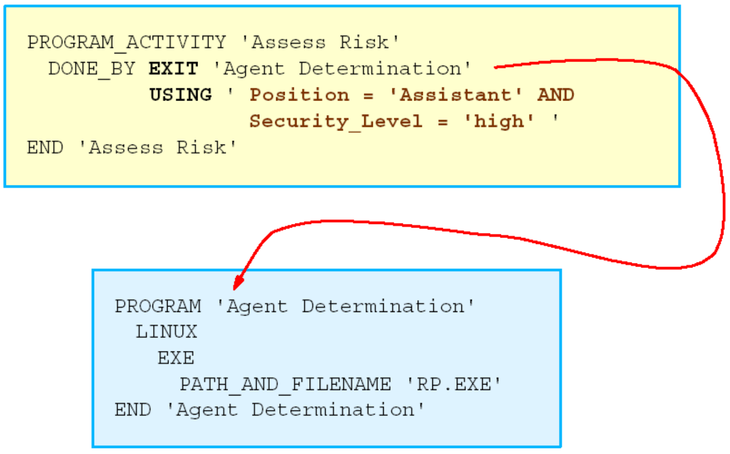
Where Organizational Data Is Managed

- 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)

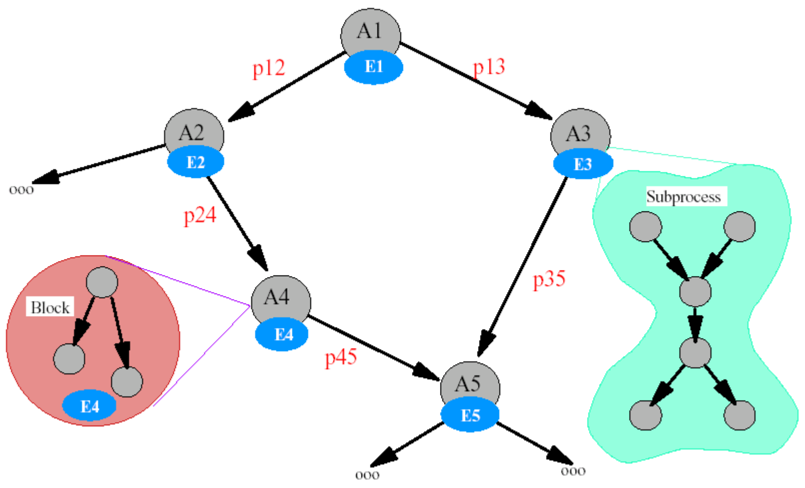
Sample: Staff Query Via Exit



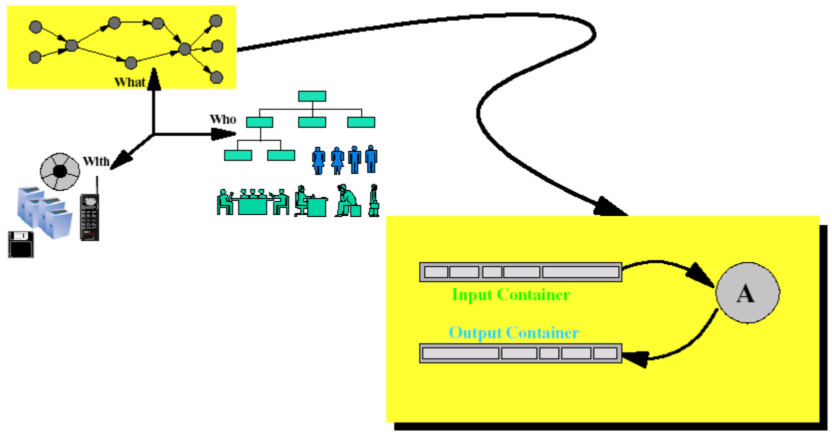
```
PROGRAM_ACTIVITY 'Assess Risk'
  DONE_BY EXIT 'Agent Determination'
          USING ' Position = 'Assistant' AND
                  Security_Level = 'high' '
END 'Assess Risk'
```

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

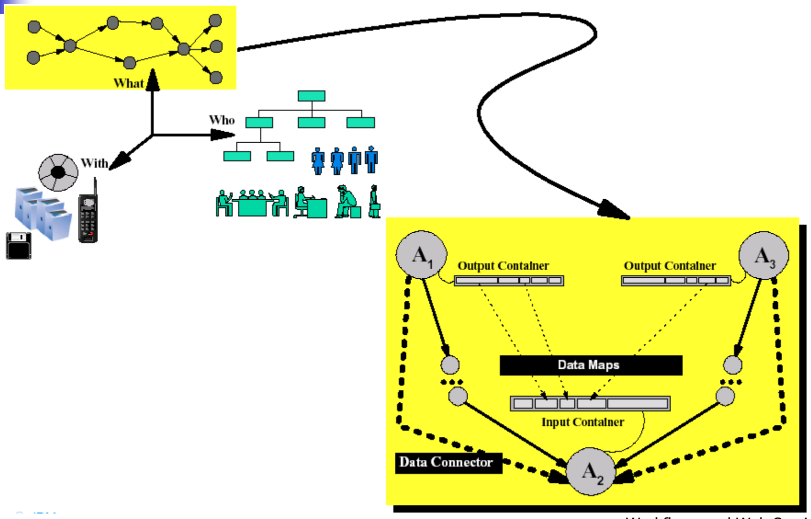
"What" Dimension: Control Flow Specification



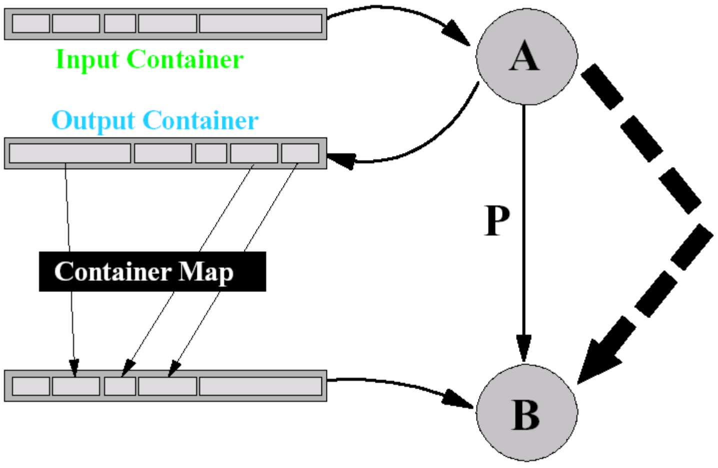
"What" Dimension: Activity Specification



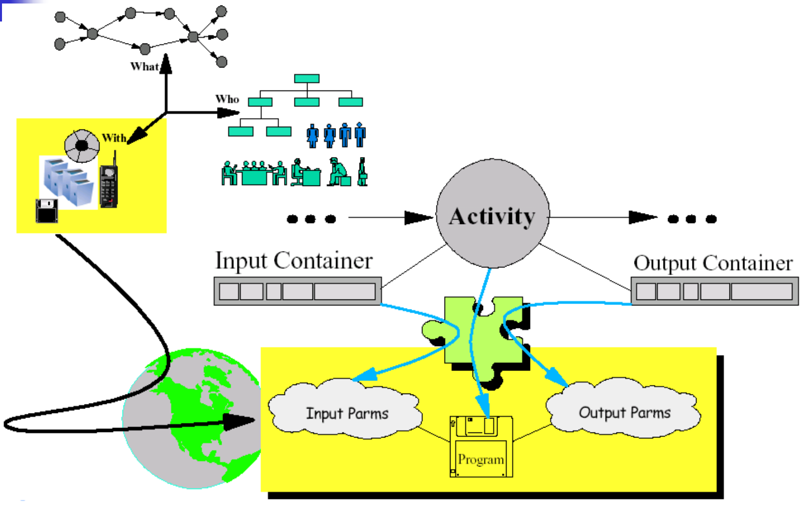
"What" Dimension: Data Flow



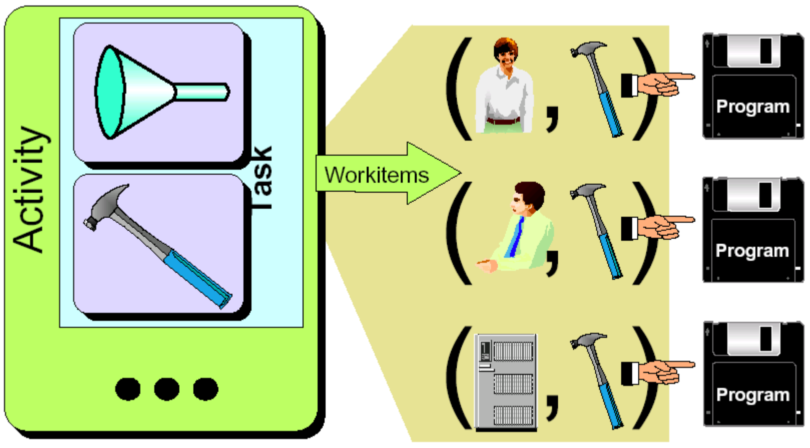
Data Flow Specification



"With" Dimension: Program Registration



Activity Structure



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

FACHBEREICH INFORMATIK

- 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

FACHBEREICH INFORMATIK

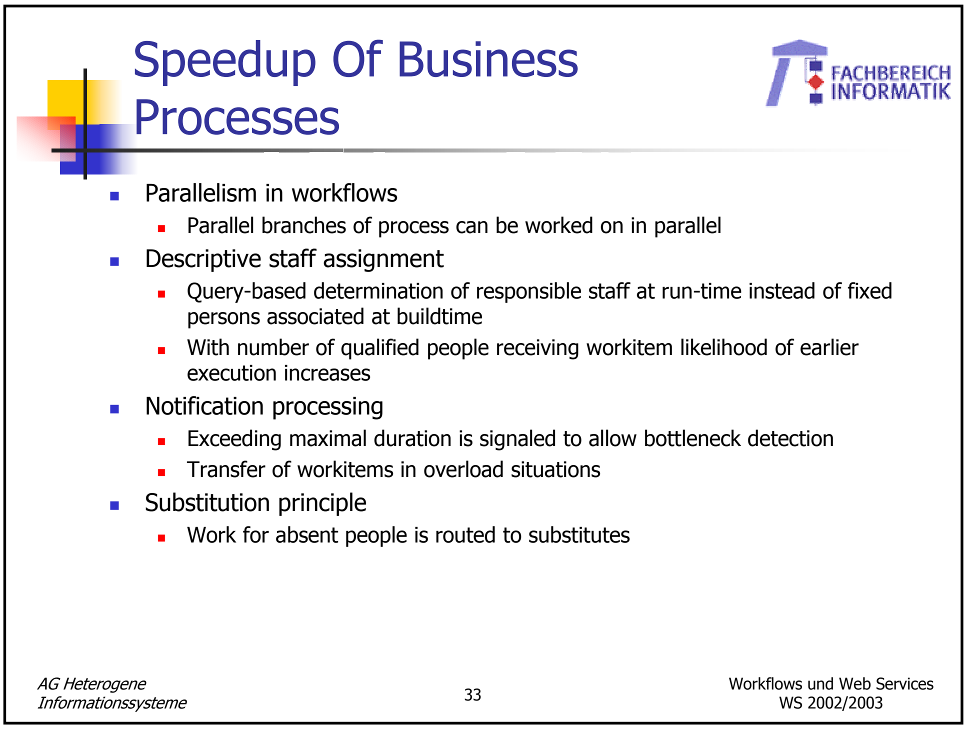
```
WORKLIST LoanProcesses
   TYPE PUBLIC
   VIEW
     WHERE PROCESS_NAME = LoanProcess
   ORDER BY Priority
   REFRESH_MODE PULL
END LoanProcesses

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

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")

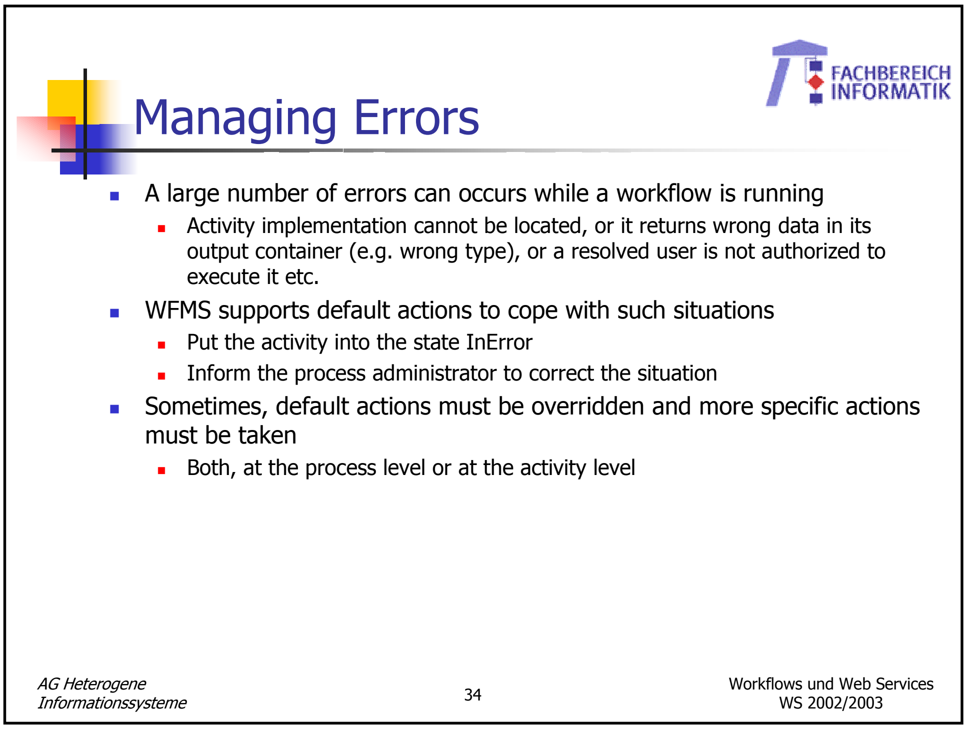
Deadlines: Example

```
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



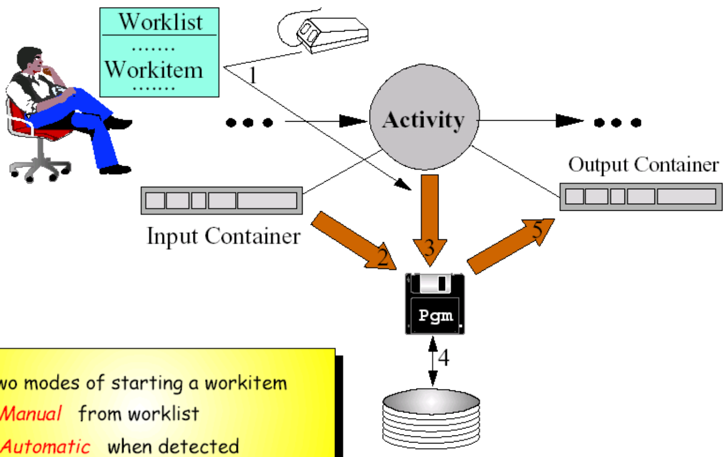
Managing Errors

- 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

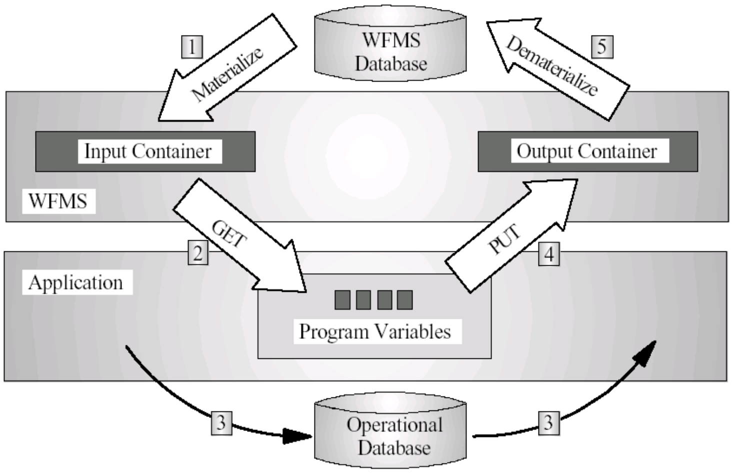
```
PROGRAM_ACTIVITY RequestApproval
   ON_ERROR
     WHEN_PROGRAM_NOT_FOUND
               NOTIFICATION TO OPERATOR
     WHEN_USER_NOT_AUTHORIZED
               NOTIFICATION TO SEC_OFFICER
END RequestApproval
```

Invoking Activity Implementations



Two modes of starting a workitem
- ✖ *Manual* from worklist
- ✖ *Automatic* when detected

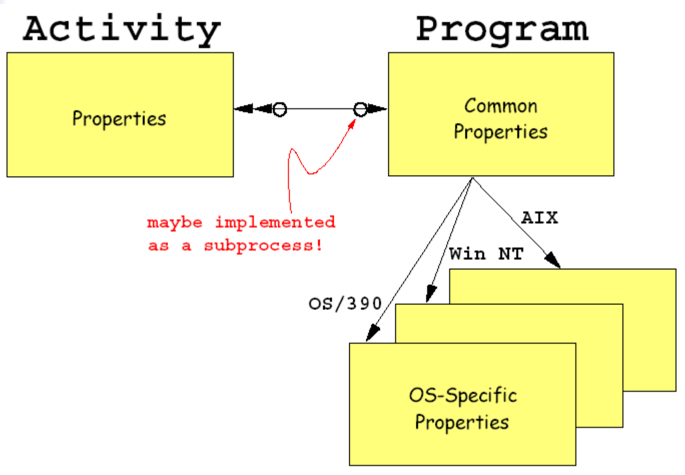
Processing Containers



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

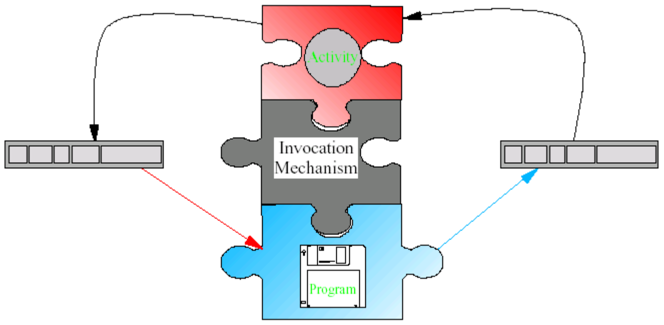
Program <-> Activity Relationship



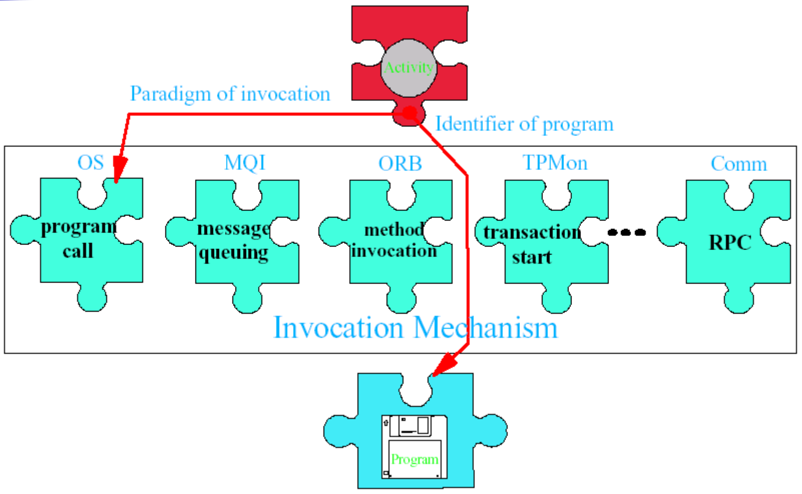
The Relationship in FDL

```
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
```

Invocation Mechanisms



Program Invocation: Metadata

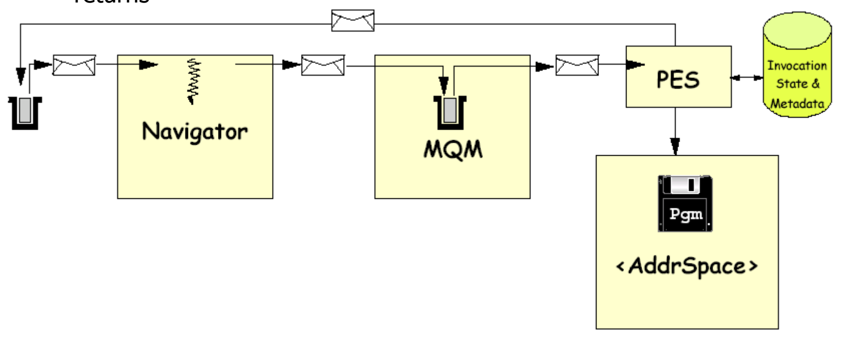


Sample Metadata

- 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

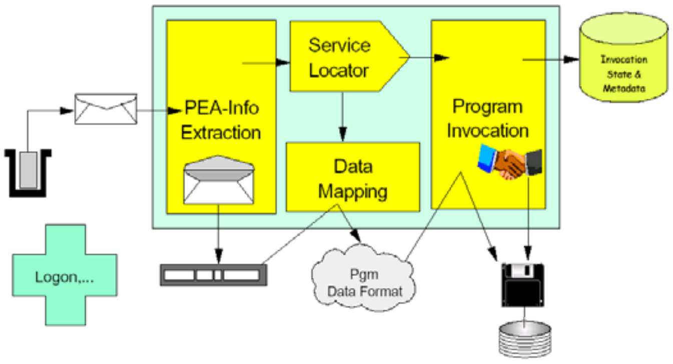
Internal Component Flow

- 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



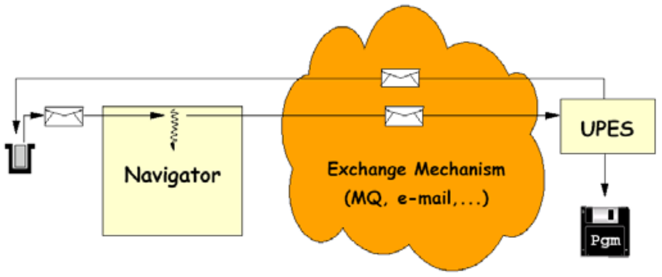
Program Execution Server

- 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.

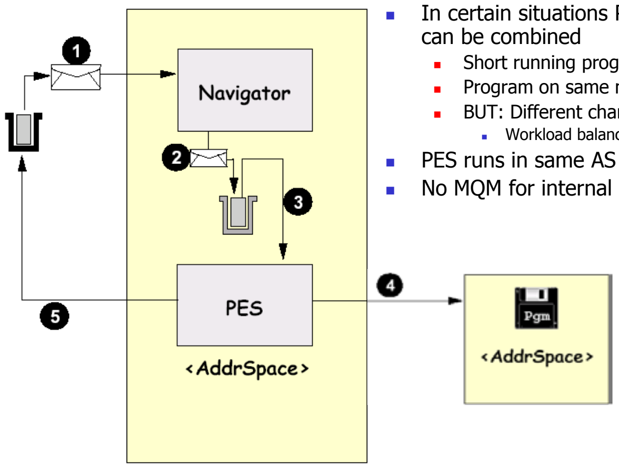


User-Provided PES (UPES)

- 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

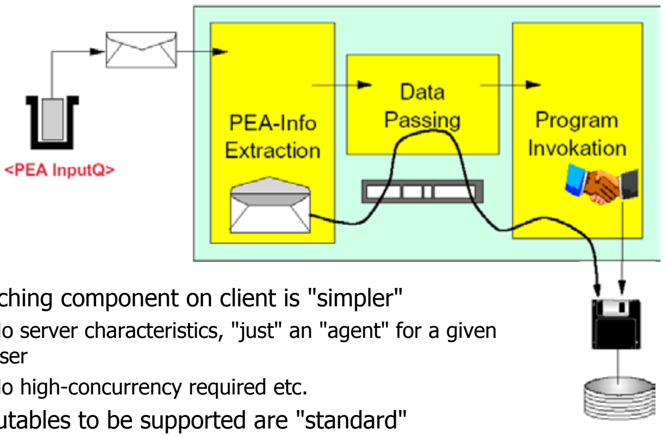


Fast Path Invocation



- 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

Subprocesses

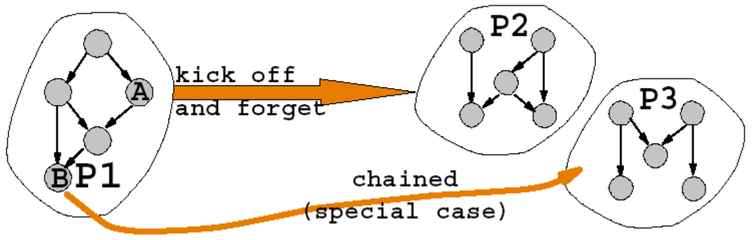
- An activity implementation may be another process, called subprocess from the point of view of the process that owns the implementing activity (so called parent process)
- A subprocess is called
  - local if it is performed by the same WFMS that runs the parent process
  - remote otherwise
- The WFMS 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

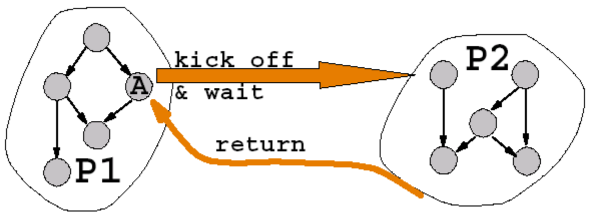
Spawned Subprocesses

- 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



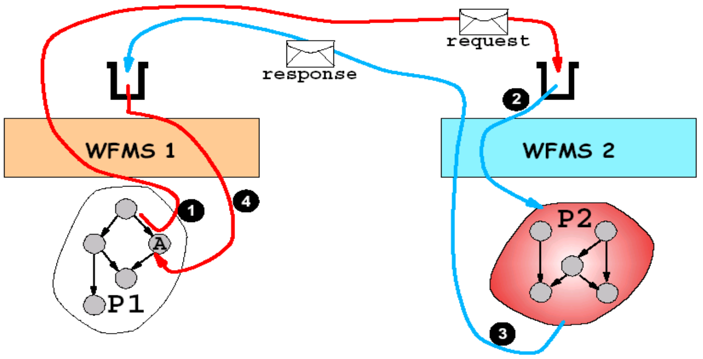
Nested Subprocesses

- Activity A of workflow P1 causes another workflow P2 to start
- P1 and P2 are dependent from 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

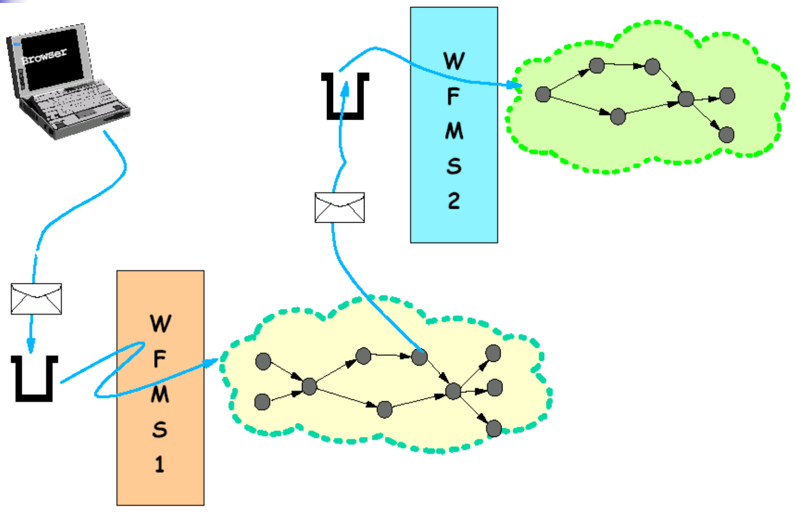
- Request/response messages could be exchanged via message queuing, e-mail etc..
- Exchange mechanism determines properties like guaranteed delivery (MQ),
- ability for "ad hoc" bindings between WFMSs etc..



Start Process Request Message In XML+

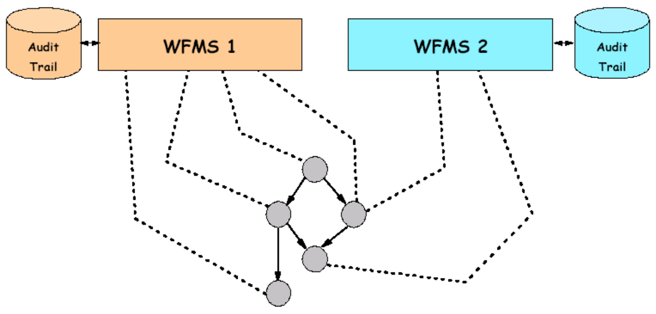
```
<?xml version='1.0' standalone='yes'>
<DOCTYPE WorkflowMessage SYSTEM 'WFMSXMLIF.DTD'>
<WorkflowMessage type='request'>
   <CorrelId>MyCorrelId</CorrelId>
   <ProcessStart>
     <ProcessTemplate>LoanProcess
     </ProcessTemplate>
     <InputData>
           <Name>Joe Smith</Name>
           <Amount>10000</Amount>
     </InputData>
   </ProcessStart>
</WorkflowMessage>
```

Starting Workflows: E2E Scenario



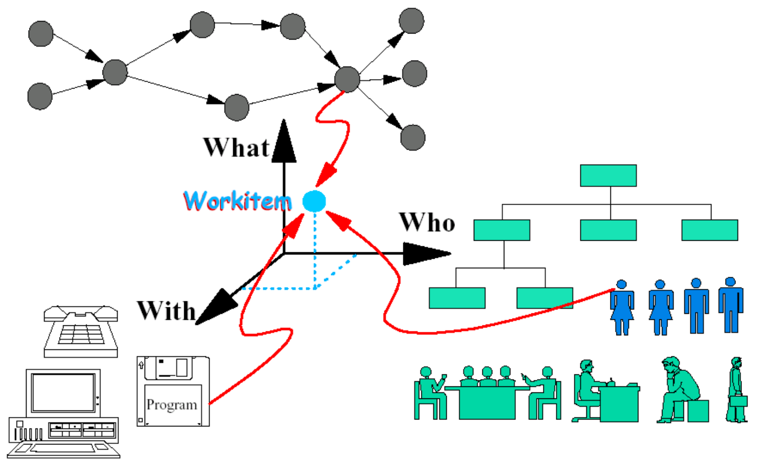
Distributed Workflows

- 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 Three Dimensions Of Workflow

FACHBEREICH INFORMATIK



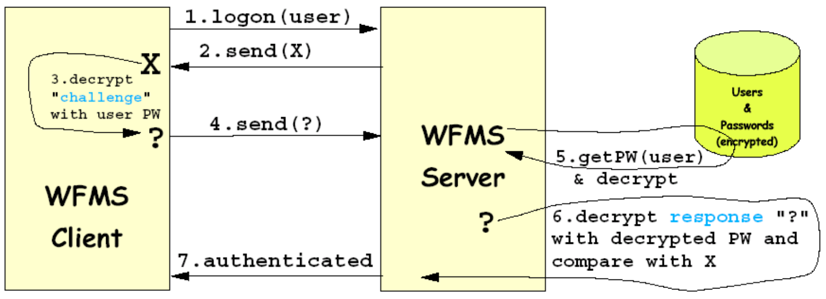
User Session

FACHBEREICH INFORMATIK

- 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...



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

Obtaining Workitems

- Three different 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

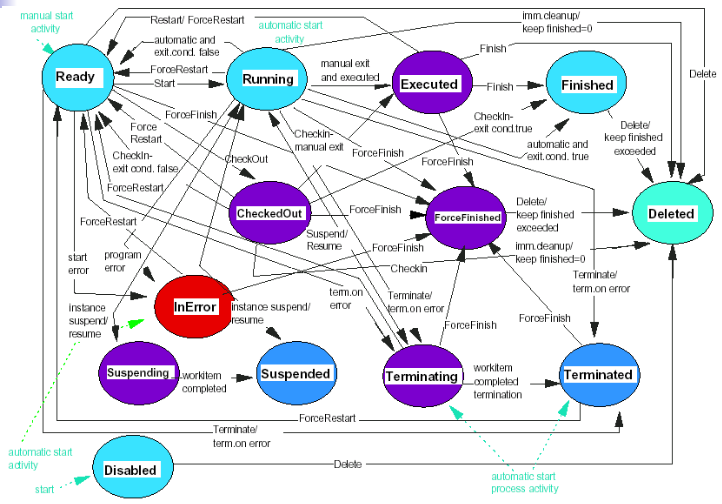
FACHBEREICH INFORMATIK

Working With Workitems

- START:First time invocation of the associated program
- TERMINATE: Stop running the associated program
- FINISH: Process exit condition for manual-exit-mode activities
- RESTART: Invocation of program after error situation
- DELETE: Remove a WI from a user's worklist
  - Valid as long as another user has a corresponding WI on a worklist
  - To be used for non-automatic-removal activities
- TRANSFER: Move WI to another user's worklist
  - Useful if original user is unable to process the WI (due to missing skill or time or...)
- HOLD: Put a WI on hold for a period of time
  - Useful when prereqs (external to the WFMS) for processing the WI are not yet met
  - WI is removed from the worklist and put back when time expired
- CHECKOUT: Treat activity as started without invoking it via the WFMS launching mechanism
  - Used for activities that are implemented by programs that have to run out of the scope of the WFMS (e.g. if WFMS does not provide the required invocation mechanism)
- CHECKIN: Accept the passed container as output container and continue usual navigation
- SUSPEND/RESUME: Applies to process activities

FACHBEREICH INFORMATIK

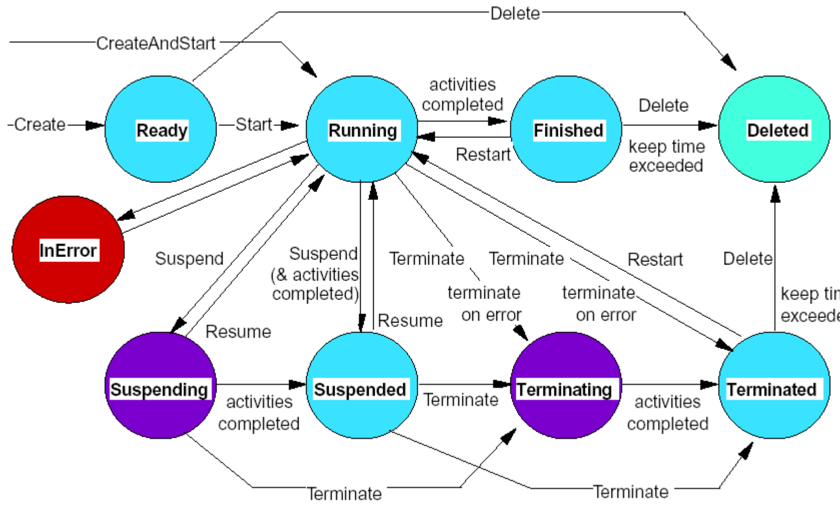
The Life Of An Activity And Workitem

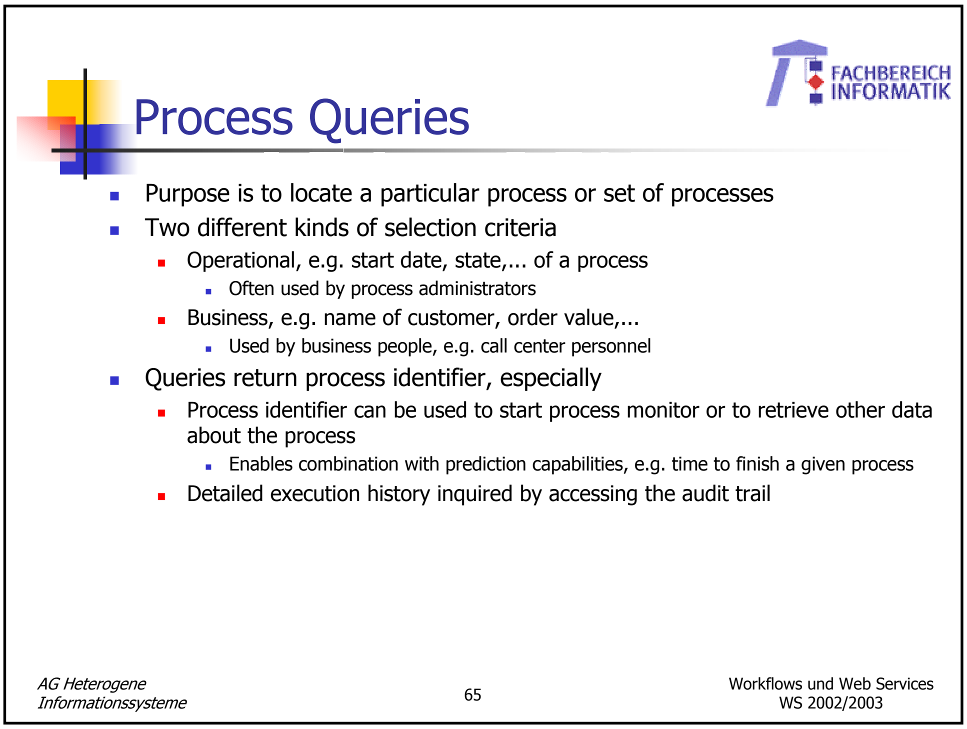


Working With Processes

- CREATE: Instantiate a process model but don't start it now
  - The values of the process input container must be provided
  - The created instance is put onto process lists of process admin, process starter ("creator") and explicitly registered (authorized) users
    - Process lists are similar to worklists
      - Push and pull mode
      - Used for monitoring particular instances
    - On completion, processes are immediately removed from process lists
- START: Begin navigating through an already created instance
- CREATE_AND_START: ... obvious...
- SUSPEND: Interrupt processing for a specified period of time
  - Wait until all running activities terminate
  - Don't accept new start request for associated workitems
- RESUME: Continue processing
- QUERY: Inquire properties of an instance
  - E.g. description, name,...
- UPDATE: Modify selective properties of an instance

The Life Of A Process

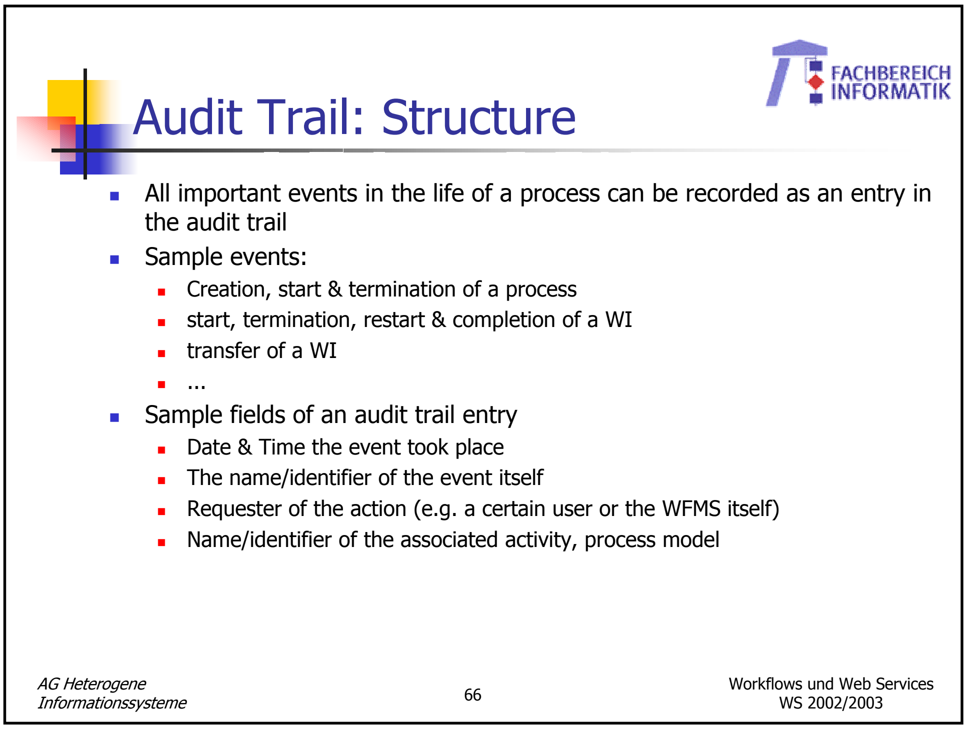




Process Queries

FACHBEREICH INFORMATIK

- 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



Audit Trail: Structure

FACHBEREICH INFORMATIK

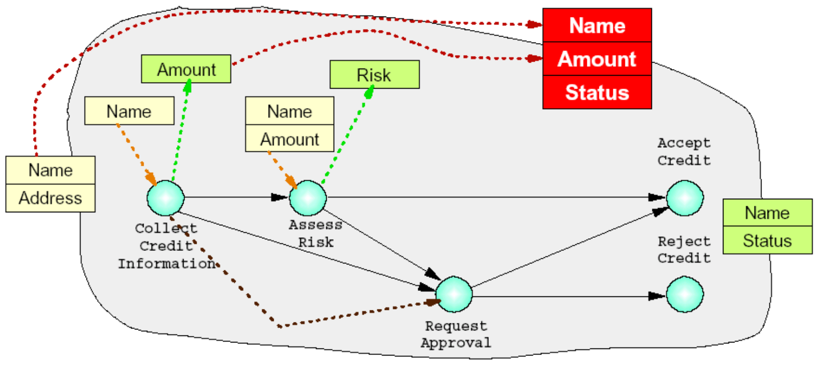
- All important events in the life of a process can be recorded as an entry in the audit trail
- Sample events:
  - Creation, start & termination of a process
  - start, termination, restart & completion of a WI
  - transfer of a WI
  - ...
- Sample fields of an audit trail entry
  - Date & Time the 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

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,...

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!
- Key container stored in audit trail to allow for better analysis of execution histories.



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

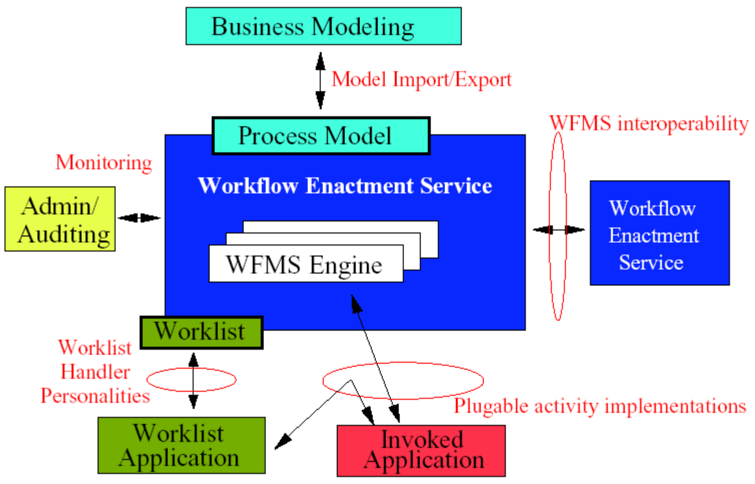
Leitstand

- 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

The WfMC Reference Model


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