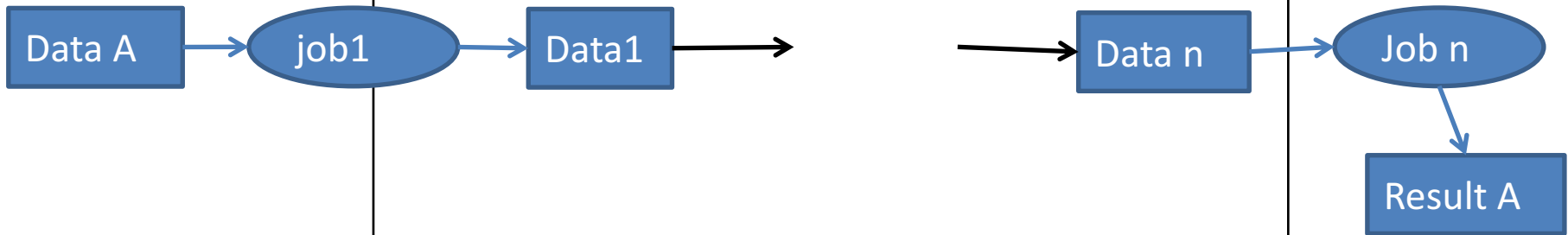


Chapter 14

WFMS

Data analysis may require more than 1 step.



In this case each job needs output from a preceding job.

I wrote such jobs for
compute farms in the
1990's

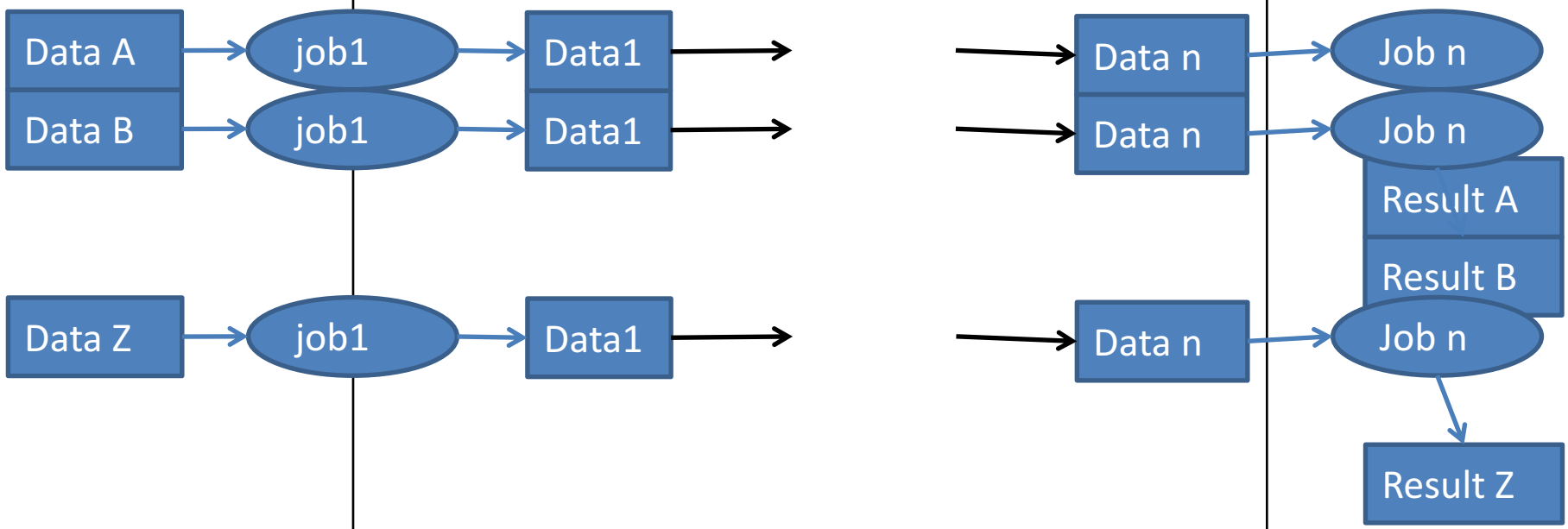
May do it by hand

- Time consuming for you
- Liable to mistakes
- Slow – suppose job k finishes at 03:00 and job k+1 must be submitted

A unix programmer would naturally write script which would handle all the steps in one file.

Would have a method of checking for job completion and only submit jobs on successful completion of antecedents.

May be repeated on more than 1 data set



Additional problems if run by hand.

- How do you keep track of which output files are related to which input files
Here the data flow is simple – but suppose some job steps need more than 1 input file.
We may run the same data more than once and alter other parameters of the job

- How do you keep track of the log files created by these jobs

Moving information through a number of steps is a common one in business it may be co-ordinated by a person or by a computer

Workflow is the computerised facilitation or automation of a business process in whole or in part

Workflow management Coalition wfmc.org

Advantages

- Reliable - less likely to miss a step
- Automatic recording of process (potentially)
- No individual involved in the fulfilling of the flow needs to understand the whole process

Can write a set of instructions for a particular process – doing it by hand.

Can write a script to control the flow for each individual workflow – but each new process is a new problem.

Split the problem of controlling a workflow into several parts.

The workflow management system (including enactment service)

The workflow description

Automating Workflows

Useful when

- the person guiding the process does not have the knowledge to perform the coordination

- the task is frequently repeated

- the task is sufficiently complex human error is likely

- time constraints are such that a human will be unable to react fast enough

- automatic and reliable response to events whose timing is unpredictable

Easier it is to define and implement workflows

- the more circumstances it will be useful to implement workflows

Separating data flow and control helps

Ad hoc v. Systematic

Unix programmers have always used scripts and pipes to help with complex tasks.

The structure tends to be light weight and flexible

BUT needs expert to write and each task is a new problem

Work flow management systems generic solutions provide

"A system that completely defines and executes 'workflow' through the execution of software whose order of execution is driven by a computer representation of the workflow logic" WFMC

Grid computing adds another layer of abstraction to the process.

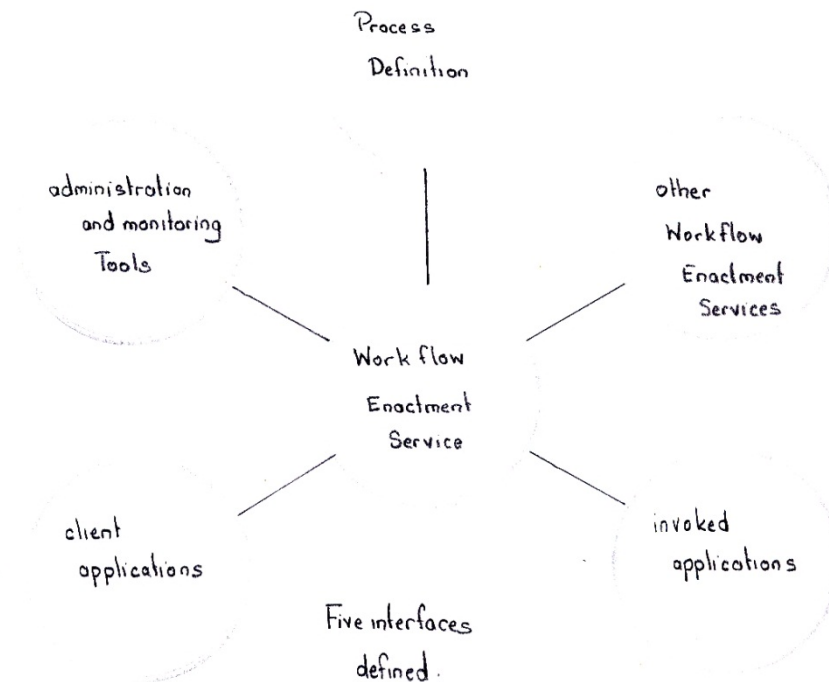
More to learn to make effective use of the resources
a greater target audience for the facility

Workflow Reference Model

Workflow Management Coalition Specification

Workflow Reference Model (WFMC-TC00-1003)

Technical Report



In principle components should be swappable between implementations

Enactment service

Workflow Enactment Service

The environment in which the applications run, responsible for interpreting and executing the process definition and interacting with the external resources needed to com

Process Definition

Any tools can be used to define a business process or task.

The output is the process definition.

WFMC leaves the tool undefined.

Specific to the application defines the interface

Allows the interchange of relevant information
via definition of formats and API

- Process start and termination conditions
- Identification of applications
- Data types and access paths
- Transition and branch conditions
- Resource allocation

Which processes to run under what conditions where to get input data what to do with output data and allow different actions dependant on the results of intermediate processes.

Client Applications & Invoked Applications

An interface which defines the way the applications which are required to complete the task interact with enactment service

Invocation and passing of parameters/data

Return of data and termination conditions

Client applications interact with users and may allow modification of the worklist (the list of items allocated to a user)

Administration Tools

Allocation of authorities to users

control of work flows

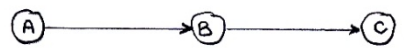
allocation of resources and priorities to work flows

Work flow interoperability

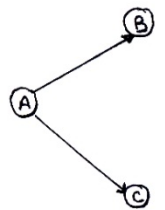
Different systems to co-operate in completing complex tasks.

The ability to exchange information about control flow and also to transfer application specific data.

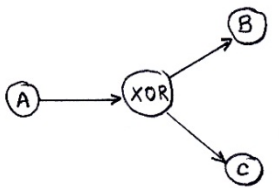
Workflow Patterns



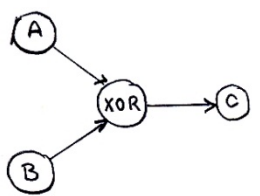
Sequential



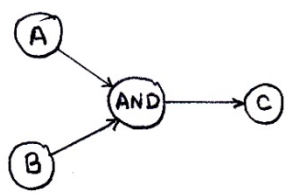
Parallel



Conditional
(Exclusive choice)



Simple Merge

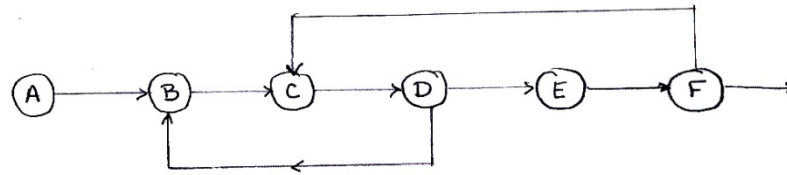


Synchronise

.... structural

Workflow Patterns: Structural

Arbitrary Cycles



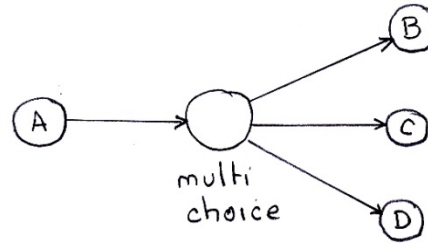
Interlocking OK

Implicit Termination

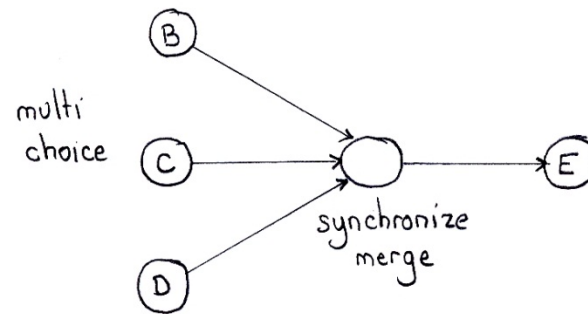
If nothing to do, no active process, nothing which can be made active (not in dead lock) end..

.... advanced

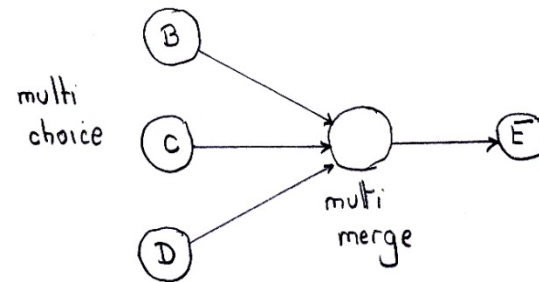
Workflow Patterns : Advanced



Control
can choose
one branch or
any combination



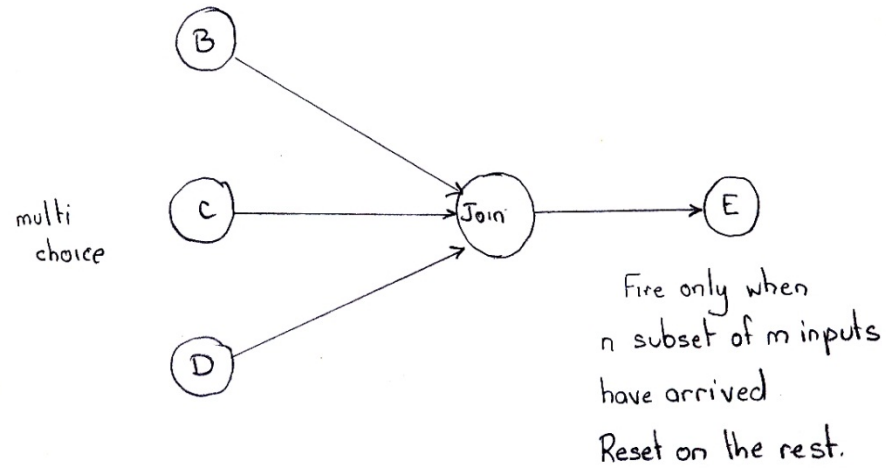
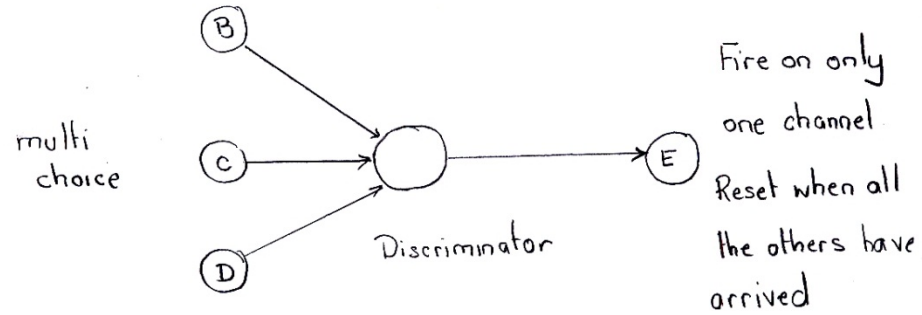
After multichoice
pass on single paths
synchronise multiple
paths.
(need to know what
choice was made)



Following multi choice
E is started for
every activation of the
incoming branch

..... (ii)

Workflow Patterns Advanced (ii)



Area of research – no definitive solution.

Some of the problems:

- A job on the Amazon cloud fails?

Should the infrastructure control resubmission?

Boundary between application and middleware.

Or between application and infrastructure

- Dynamic environment in resources and data paths may appear and disappear during the course of the workflow.

- Controlling response for time sensitive applications.

- Large datasets – transfer problems

- Timescales of operations from milliseconds to day/weeks.

To leverage distributed infrastructure existing monolithic scientific applications are currently being re-engineered and decomposed in a set of atomic activities orchestrated in a loosely coupled scientific workflow.

Workflow

Despite their similarities with the workflows originating from the business world, scientific workflows to be executed in distributed infrastructures present fundamental differences that make them rather unique and, therefore, impose specific requirements to support them:

Too many activity instance to monitor (or even name) individually.

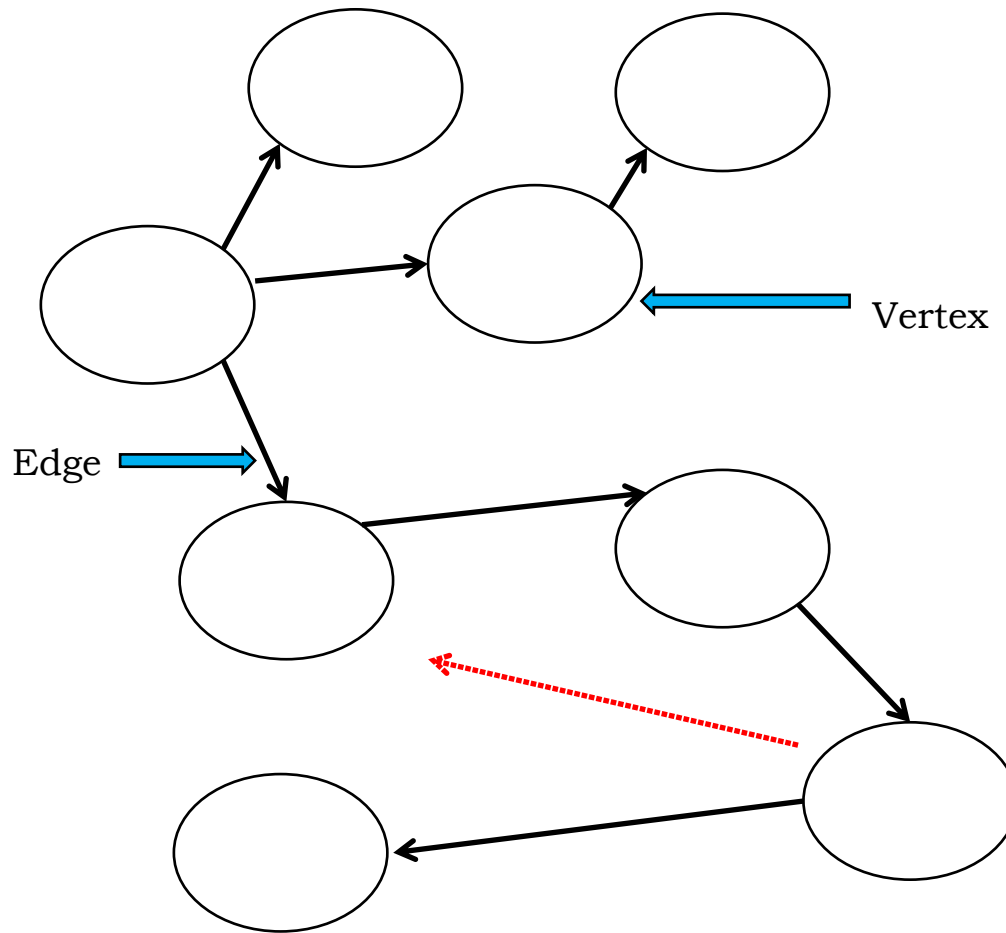
- computationally intensive activities with long and often unpredictable execution times;
- complex data dependencies of various sizes ranging from few bytes to several gigabytes;
- sequential loops that transform workflows into complex DG-based structures, as opposed to simpler DAGs characteristic to the business world;
- dynamic control and data flow structure, may change at runtime depending on input workflow parameters or on output results produced by the workflow activities.

and

- unreliable execution resources that raise complex fault tolerant issues (including network failures).

They list some problems relevant to GC

Directed (Acyclic) Graphs



Di(irected) Graph. Vertices joined by edges, with an arrow indicating direction of join.

Redline introduces a cycle.

Acyclic graphs are partially ordered which makes them easier to handle.

- The initiation of a set of jobs must be automatic
It must be possible to start a set of processes running with a small number of actions. (or even have them triggered by external events)

- The merging of the output at the conclusion must be automatic

As jobs end and output is returned it must be automatically merged.

Jobs will rarely if ever return output in the order in which they are submitted.

- The detection of job failure must be (mostly) automatic

Rare cases may be handled by user action

- It must be possible to interact manually with the job flow

- Resubmission must be automatic

Including jobs that fail with no output

- Merging must allow for the return of the output of data segments more than once.

Slow jobs which appear to have failed

CMS a LHC experiment has written its own workflow system.

- Simulation over a set of input conditions
- Analysis of subsets of data

Actions for failure conditions are likely to be very application specific.

eg

multiple jobs to search a data to see if a condition is ever satisfied.

*Multiple returns of yes or no from the same subset will be harmless – **But** all the data must be searched.*

multiple jobs to search a data to see how often a condition is satisfied.

Multiple returns of a number of “hits” from the same data will result in an incorrect answer.

Return of zero does not have to be checked.

If the data has a meaningful order then the results must be merged in a suitable order.

If a simulation is split into tens of thousands of sub-sections, missing a sub-section is unlikely to be significant (plan to make it insignificant). Keep submitting until sufficient data is collected.

Stage of a workflow kicks off another process/application.

How long does it take to start?

Network latency – in job transfer – in data transfer?

Job submission latency?

Possible queuing time?

This may lead to significant overheads in execution time

Problem for time critical tasks.

Overlap tasks – but not always possible.

Awareness and imagination!

Data set for step n ,
may depend on step
 $n-1$.

Grid computing has utilised a different approach to multiple jobs.

- same job with different parameters
- same job with different data set
- a number of interlinked jobs

a number of interlinked jobs: WFMS

Same job with different data sets: each job must be scheduled

- that means a target (lightly loaded machine must be identified. This takes time.
- Each time it must then copy the application to the target system,
- The job will enter a queue and wait. Time
- Recover output.

Same job with different parameters: this may be worse since the same data set may be copied repeatedly to a target system.

No good just gluing jobs together to make a single big job:

- separate log files and output files
- problem with job failure

To mitigate some of these problems the concept of a pilot job is used.

A pilot job is submitted in the normal way. But there is a control job constantly executing

The job communicates with a control job when it is ready to run.

The control job hands out a “token”
The token describes the job:

Description:

- parameters;
- data set;
- parameters & data set;
- application,

Reduce SE to WN
copying.
Saves time SRM latency
Bandwidth

(In particle physics the application is often large and time consuming to compile – separate jobs then run to install the applications on a target machine– it then becomes reasonable for a pilot job to be told the app to run.

When finished it does any necessary book keeping.
Making result files available to the user.
Perhaps reporting that job is finished.

Checks how much CPU time/ Wall time it has left.
If sufficient asks for a new token

A pilot job creates one log file :

There needs to be records of each job – can be done by the pilot job creating separate files or by the control job keeping a record of the jobs which complete and the associated information. (book keeping)

There is some automatic load balancing
faster machines request tokens more often
dynamic: not dependent on published
performance figures, but actual performance.

User has more reliable response:
can keep a population of pilot jobs running
permanently and be immune from vagaries of the
queuing system.

For instance a pipeline



Conventional workflow says app2 is not submitted until app1 has completed. App2 may then have to spend time in a queue – for multistep may be significant.

Operation (ii)

Beware
A pilot job may sit around waiting for data and burning up wall-time.
If you are paying for occupying a slot, this may cost you as much as running.

Lessons

Automate complex processing systems by using workflows.

Pilot jobs provide an example of a user job augmenting the facilities of the computing resource.