The UNICORE Work Flow Model

Dietmar Erwin
Zentralinstitut für Angewandte Mathematik
Forschungszentrum Jülich
D.Erwin@fz-juelich.de
GGF5 – GCE, July 23, 2002

Contents

UNICORE

UNICORE is funded in part by BMBF, the German Ministry of Education and Research under project grant:

UNICORE Plus: 01-IR-001

January 1, 2000 - December 31, 2002

http://www.unicore.de

UNICORE Goals

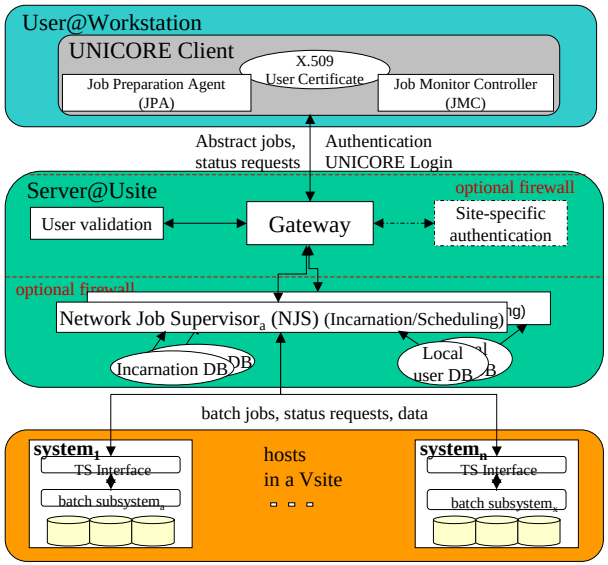
UNICORE:

UNiform Interface to COmputing Resources

- **Seamless**
- **Secure**
- **Intuitive**

access to distributed German HPC resources in a production environment

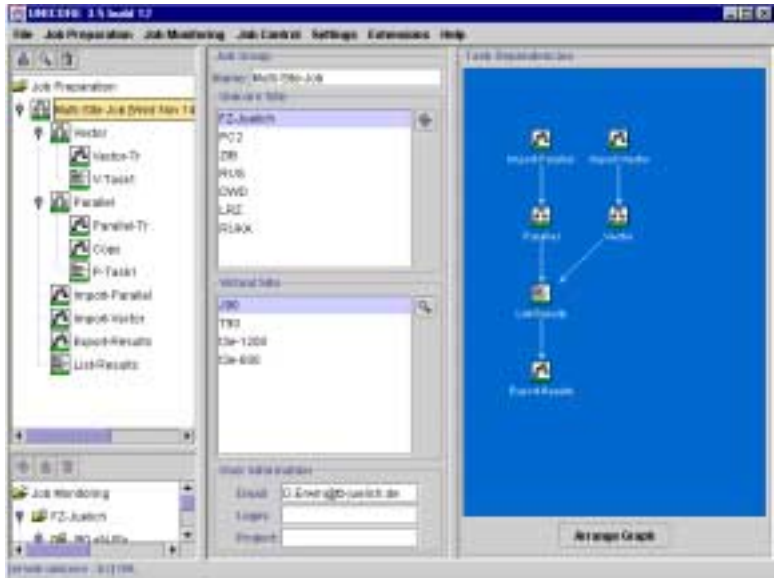
UNICORE Architecture

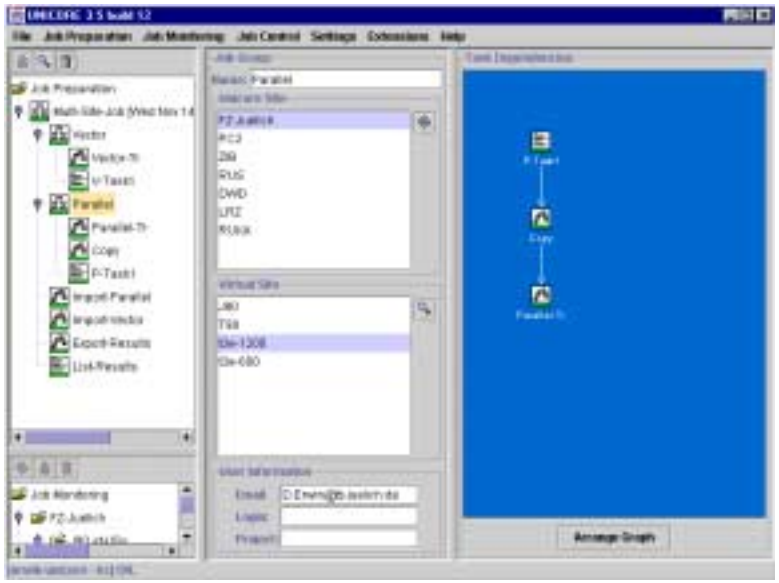


UNICORE Client

UNICORE Client assists in creating, manipulating and managing

- Complex, interdependent
- Multi-system jobs
- Multi-site jobs
- Synchronisation of jobs
- Movement of data between systems and sites and storage spaces





UNICORE Server

The client creates an Abstract Job Object (AJO) represented as serialised Java object or in XML

The Server (NJS) performs

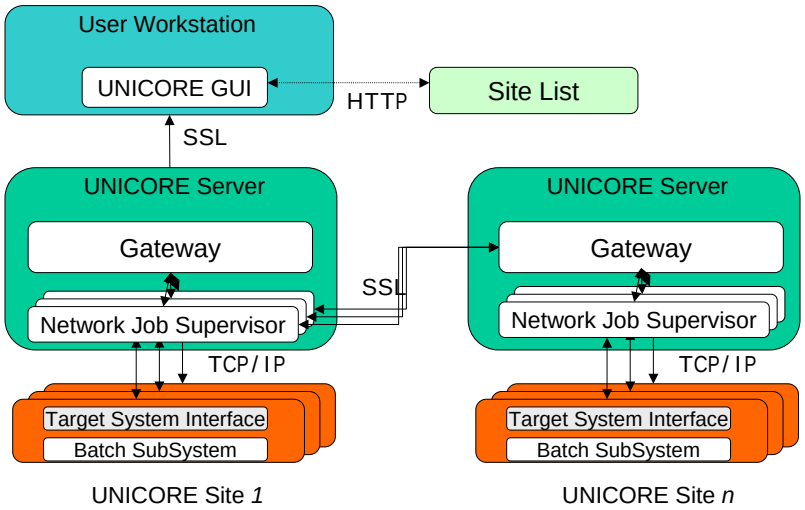
- Incarnation of the AJO into target system specific actions
  - Real batch jobs
  - File transfers, ….
- Synchronizes actions (work flow)

UNICORE Server

The Server (NJS) performs

- Transfers of jobs and data between
  - User Workstation
  - Target Systems
  - Other sites
- Monitoring of status

UNICORE Architecture



Work Flow

- UNICORE Work Flow can be modelled by a Directed A-cyclic Graph (DAG)
- A UNICORE Job consists of a set of DAGs
- Successors are executed if and only if all predecessors complete successfully

Users ask for

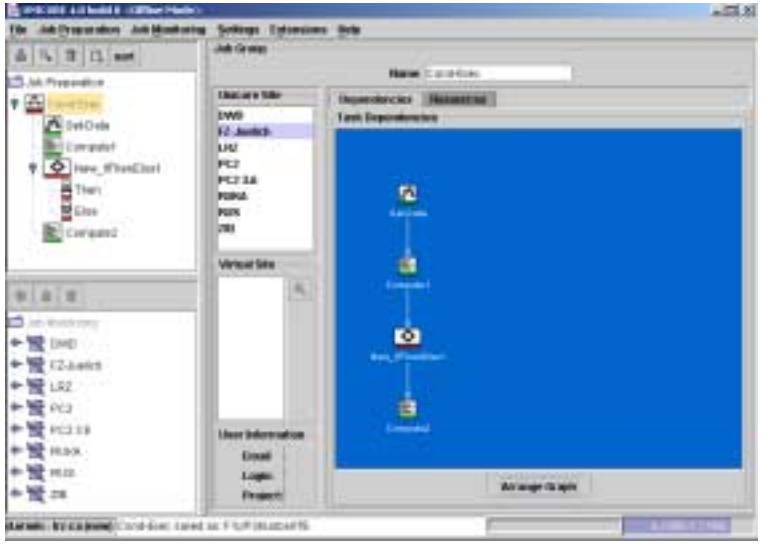
- Conditional Execution
- Repeated Execution

Extended Work Flow

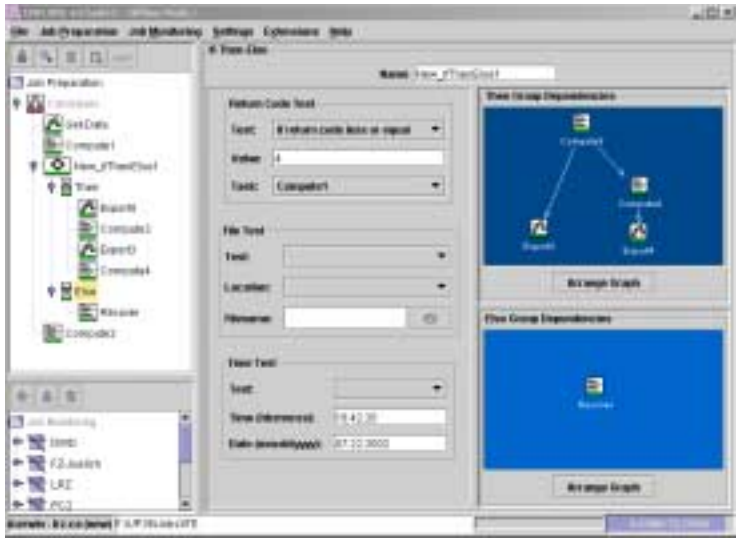
Following constructs are added:

- If-Then-Else construct
- Do N
- Repeat until

If-Then-Else



If-Then-Else



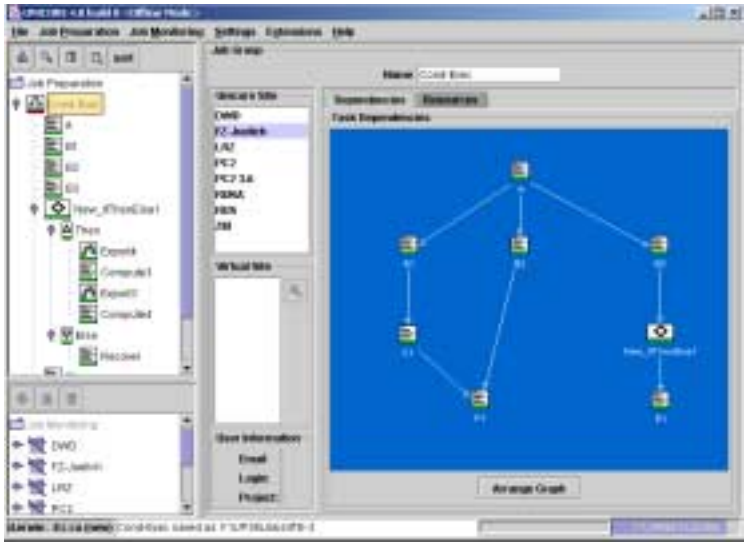
Work-Flow Implementation

- Part of the UNICORE Object Hierarchy
  - AbstractAction: Parent class of all UNICORE actions.
    - ActionGroup: Container for UNICORE actions.
      - AbstractJob: ActionGroup which can run remotely.
      - RepeatGroup: Actions in a loop.
    - AbstractTask: A computational action, e.g. copy file.
    - AbstractService: A service action, e.g. kill job.
    - ConditionalAction: If-then-else for actions.
- ActionGroup contains a DAG of AbstractActions
  - Dependencies between actions (nodes) define control flow.
  - Actions in the DAG can be any subtype of AbstractAction.
  - An action starts when all it predecessors are “DONE”.
  - Subclasses of “DONE” are used for control.

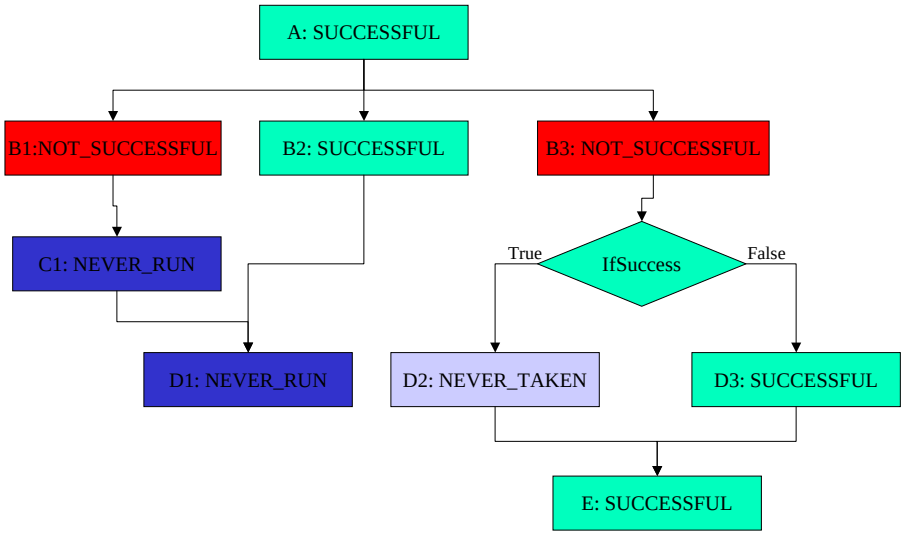
Subclasses of “DONE”

- SUCCESSFUL:
  - The AbstractAction completed without error.
- NOT_SUCCESSFUL:
  - The AbstractAction failed.
- NEVER_RUN:
  - A predecessor of the AbstractAction failed.
- NEVER_TAKEN:
  - The AbstractAction is on the not taken branch of a conditional action.

If-Then-Else



Example



Looping Constructs

- Looping Constructs
  - An ActionGroup where the actions can be re-run.
  - All iterations are performed in the same directory (on the same Vsite).
  - At least one iteration must be run.
  - All Action statuses set to NOT_DONE before each iteration.
  - Effectively, Tail Recursion.
- ForGroup (DO n)
  - Loop for a constant number of iterations.
- RepeatGroup (REPEAT UNTIL)
  - Loop until a condition is met.
  - Condition based on expression involving the return code of an action in the group.

Summary

The work flow constructs in UNICORE allow:

- Automating complex multi-site, multi-system chains of Jobs
- Run computational experiments like parameter studies
- Use all features of UNICORE, like security and seamlessness

UNICORE **is available as 'open source' at:**

- **http://www.unicore.org**
  Download for Software + Sources under Community License
- **http://www.fz-juelich.de/unicore-test**
  UNICORE Client and access to a free computational Grid to explore the functions and features