EDG WP4 (fabric mgmt): status&plans

Large Cluster Computing Workshop

FNAL, 22/10/2002

Olof Bärring

Outline

- What's "EDG" and "WP4" ??
- Recap from LCCWS 2001
- Architecture design and the ideas behind…
- Subsystem status&plans&issues
  - Configuration mgmt
  - Installation mgmt
  - Monitoring
  - Fault tolerance
  - Resource mgmt
  - Gridification
- Conclusions

"EDG" == EU DataGrid project

- Project started 1/1/2001 and ends 31/12/2003
- 6 principal contractors: CERN, CNRS, ESA-ESRIN, INFN, NIKHEF/FOM, PPARC
- 15 assistant contractors
- ~150FTE
- http://www.eu-datagrid.org
- 12 workpackages

"WP" == workpackage

This is what I'm gonna talk about today

EDG WPs

- WP1: Workload Management
- WP2: Grid Data Management
- WP3: Grid Monitoring Services
- WP4: Fabric management
- WP5: Mass Storage Management
- WP6: Integration Testbed – Production quality International Infrastructure
- WP7: Network Services
- WP8: High-Energy Physics Applications
- WP9: Earth Observation Science Applications
- WP10: Biology Science Applications
- WP11: Information Dissemination and Exploitation
- WP12: Project Management

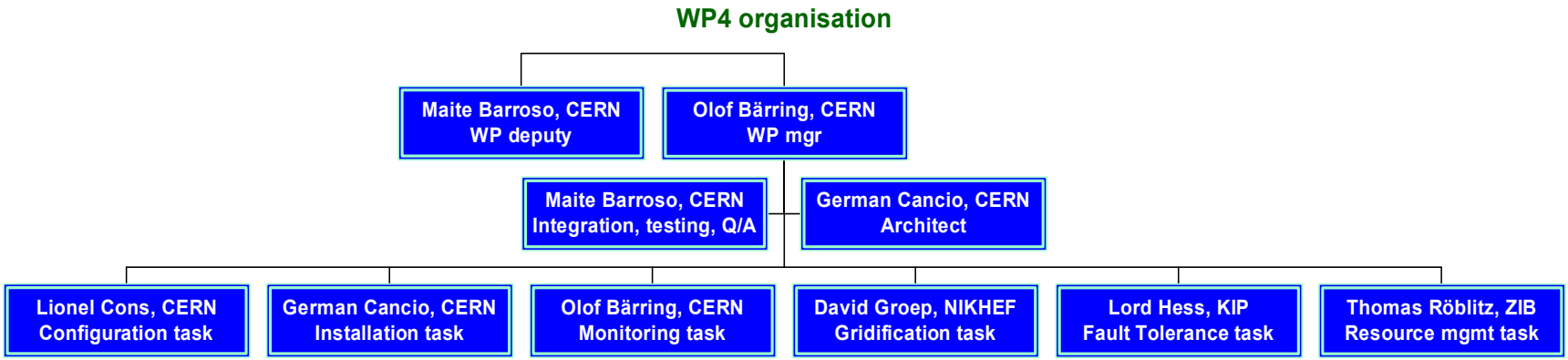
WP4: main objective

"To deliver a computing fabric comprised of all the necessary tools to manage a center providing grid services on clusters of thousands of nodes."

- **User job management (Grid and local)**
- **Automated management of large clusters**

WP4: structure

- ~14 FTEs (6 funded by the EU). Presently split over ~ 30 - 40 people
- 6 partners: CERN, NIKHEF, ZIB, KIP, PPARC, INFN
- The development work divided into 6 subtasks



Recap from LCCWS-1

EDG WP4 presentations in LCCWS-1 //-sessions

Session	What we said	What happened
Installation	Plans for using the LCFG tool from Edinburgh Univ. as an interim installation/maintenance system	LCFG in production on EDG testbed since 12 months. Will be replaced by new system 2Q03.
Monitoring	PEM vs. WP4. Design for node autonomy where possible	System deployed on EDG testbed since one month
Grid	Early architecture design ideas and development plans up to Sept. 2001	Architecture design refined and adopted. Delivery OK.

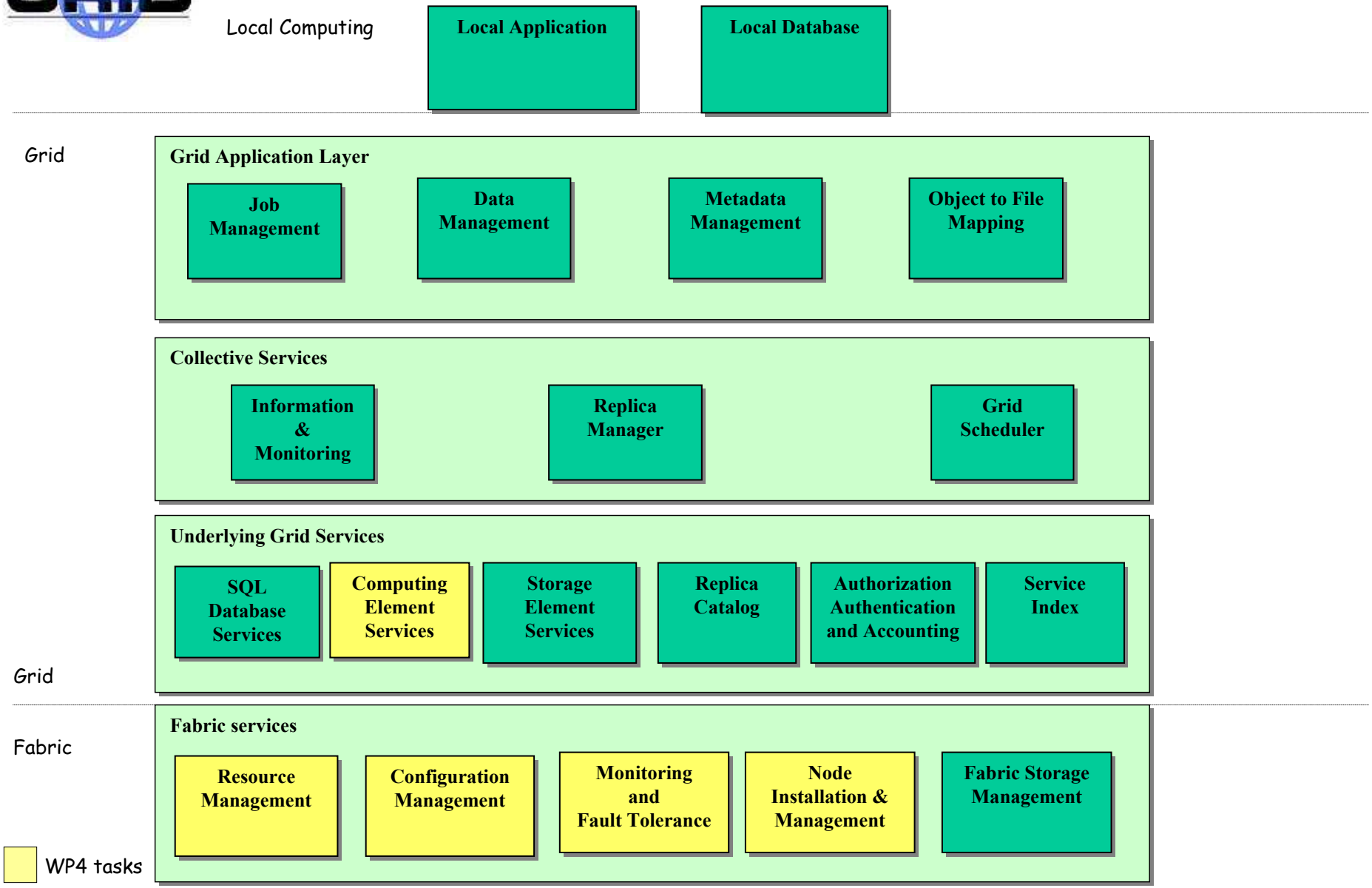
Not everything worked smoothly

- Architecture design: had to reach consensus between partners with different agendas and motivations.
- Delivered software: we learned some lessons and had taken some uncomfortable decisions

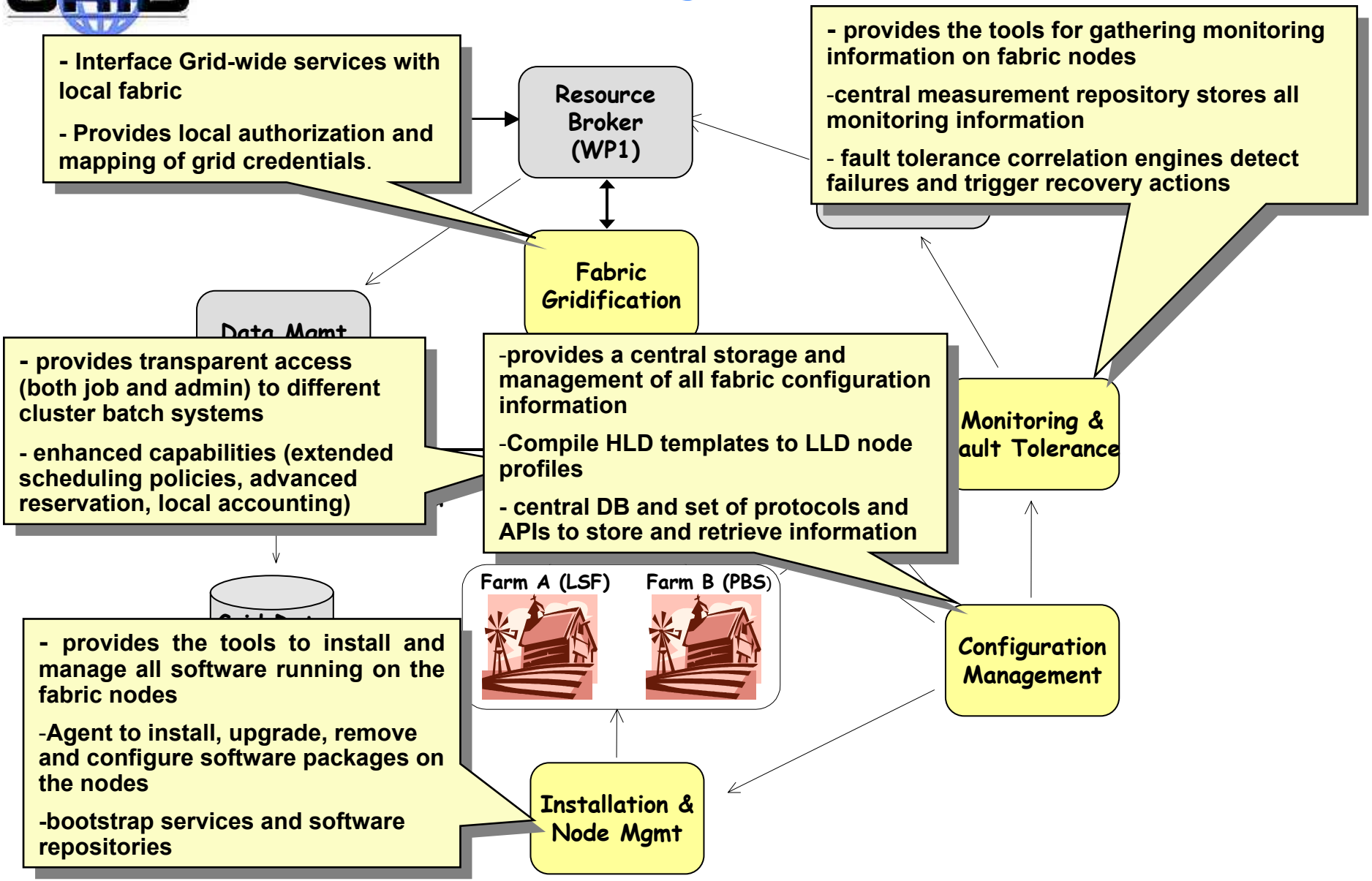
Architecture design and the ideas behind

- Information model. Configuration is distinct from monitoring
  - Configuration == desired state (what we want)
  - Monitoring == actual state (what we have)
- Aggregation of configuration information
  - Good experience with LCFG concepts with central configuration template hierarchies
- Node autonomy. Resolve local problems locally if possible
  - Cache node configuration profile and local monitoring buffer
- Scheduling of intrusive actions
- Plug-in authorization and credential mapping

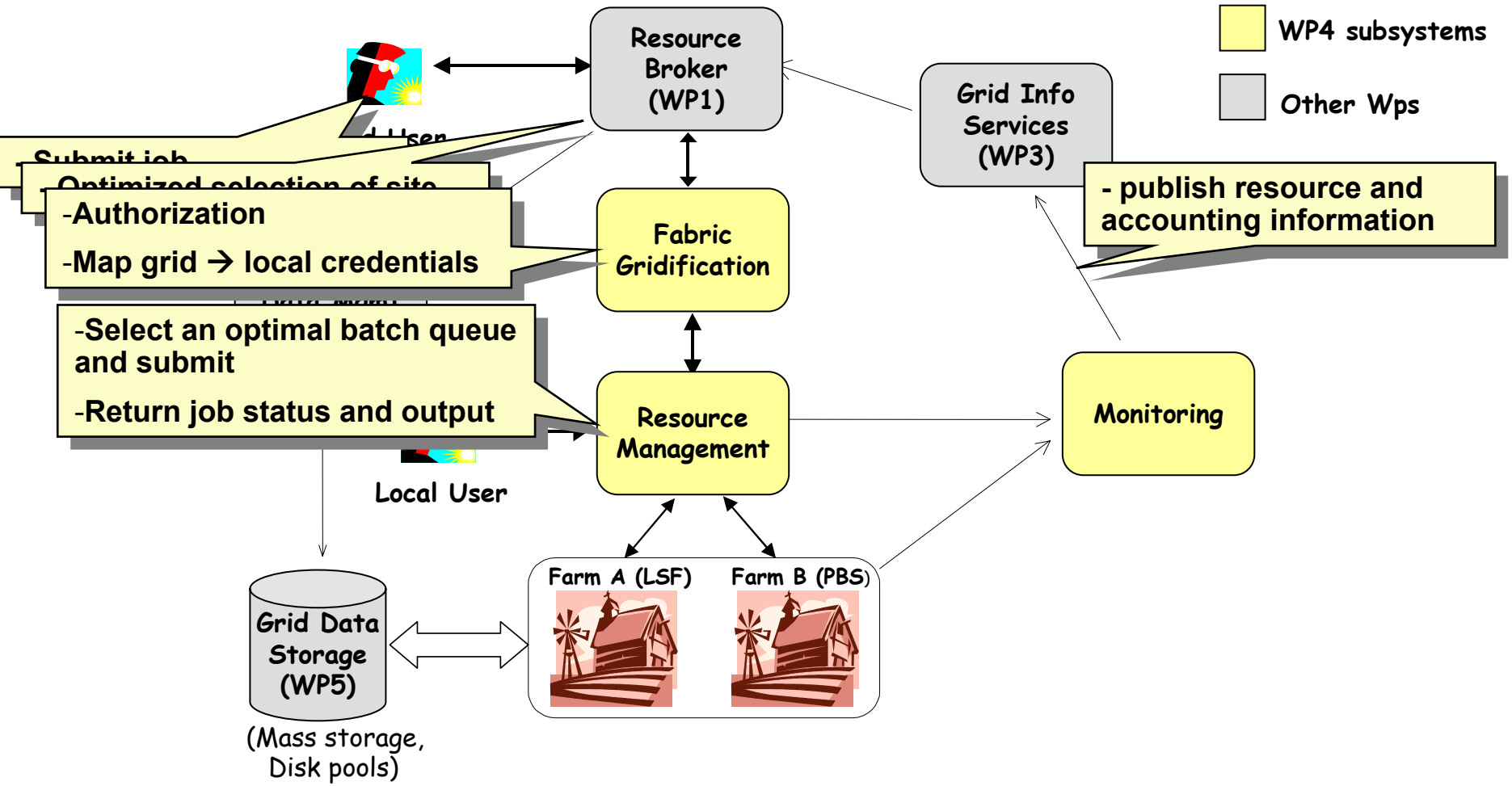
DataGrid Architecture



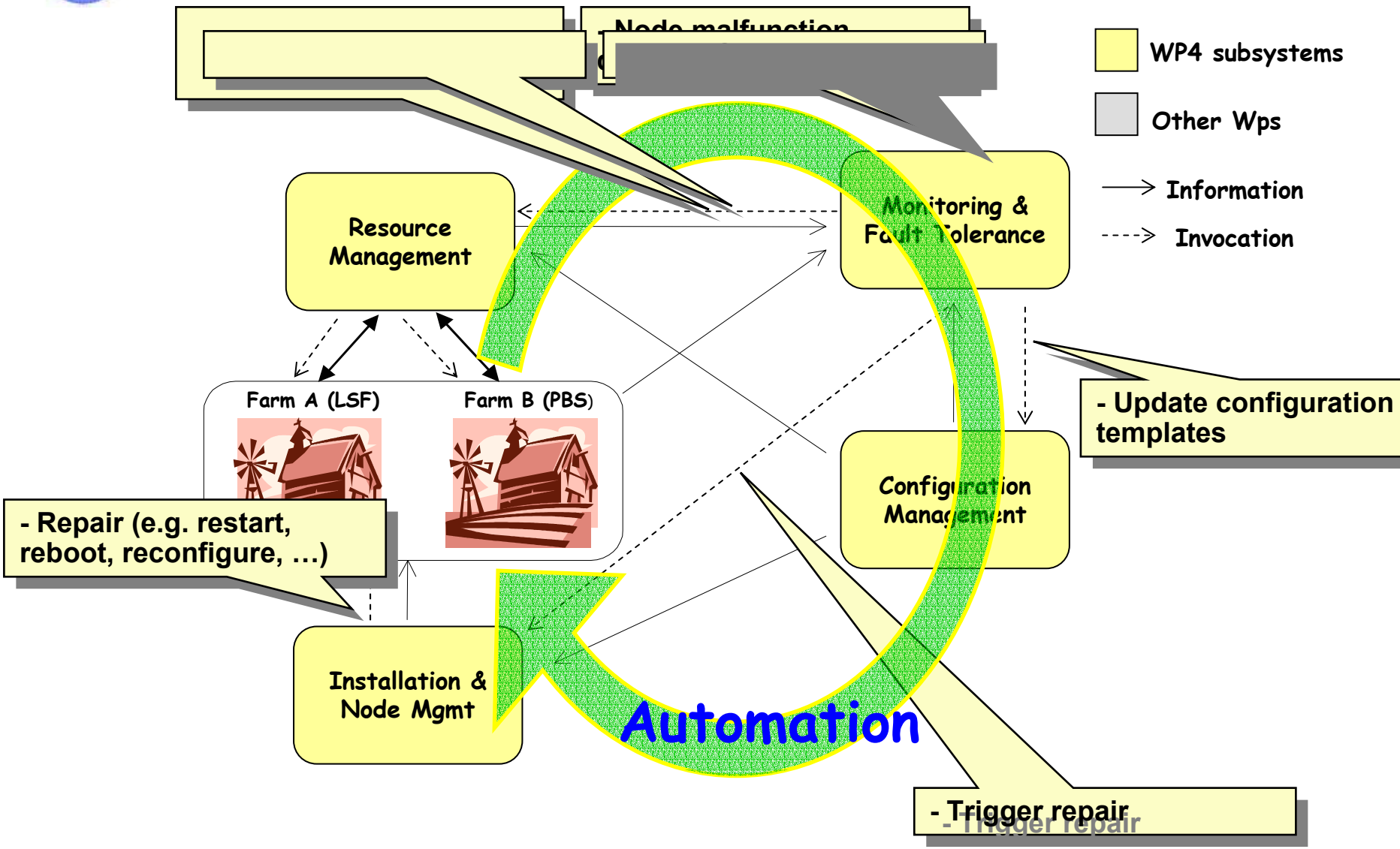
WP4 Architecture logical overview



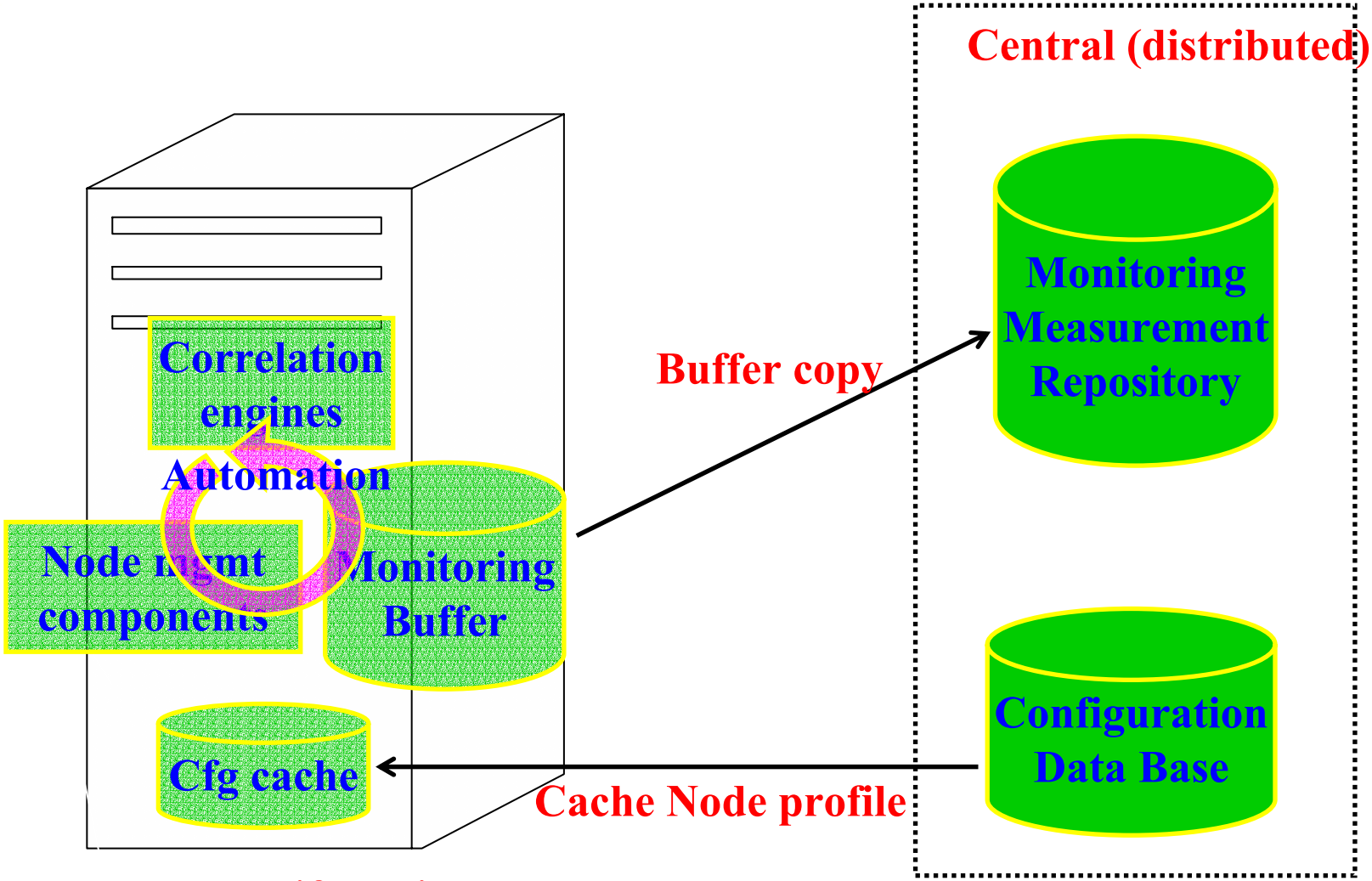
User job management (Grid and local)



Automated management of large clusters

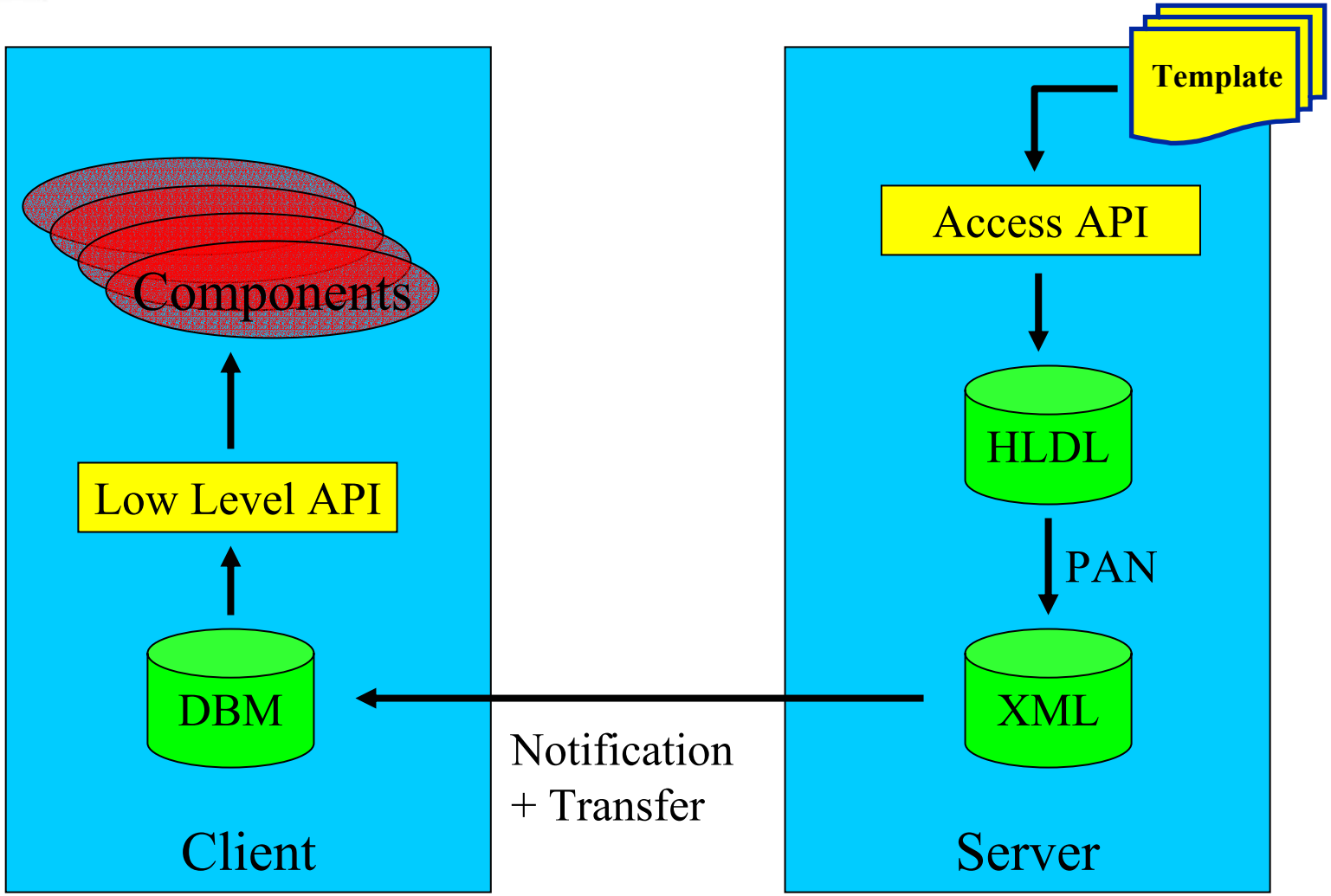


Node autonomy



Local recover if possible (e.g. restarting daemons)

Subtasks: configuration management



Configuration templates like this …

```
# TEST Linux system
#####################################
object template TEST_i386_rh72;
"/system/platform" = "i386_rh72";
"/system/network/interfaces/0/ip" = "192.168.0.1";
"/system/network/hostname" = "myhost";
include node_profile;
```

```
# Default node profile
#####################################
template node_profile;

# Include validation functions
##############################
include functions;

# Include basic type definitions
################################
include hardware_types;
include system_types;
include software_types;

# Include default configuration data
####################################
include default_hardware;
include default_system;
include default_software;
```

```
# SYSTEM: Default configuration
#########################
template default_system;

# Include default system configuration
#####################################
include default_users;
include default_network;
include default_filesystems;
```

```
# SYSTEM: Default network configuration
#####################################
template default_network;
"/system/network" = value("//network_" +
                    value("/system/platform") +
                    "/network");
```

… generate XML profile like this

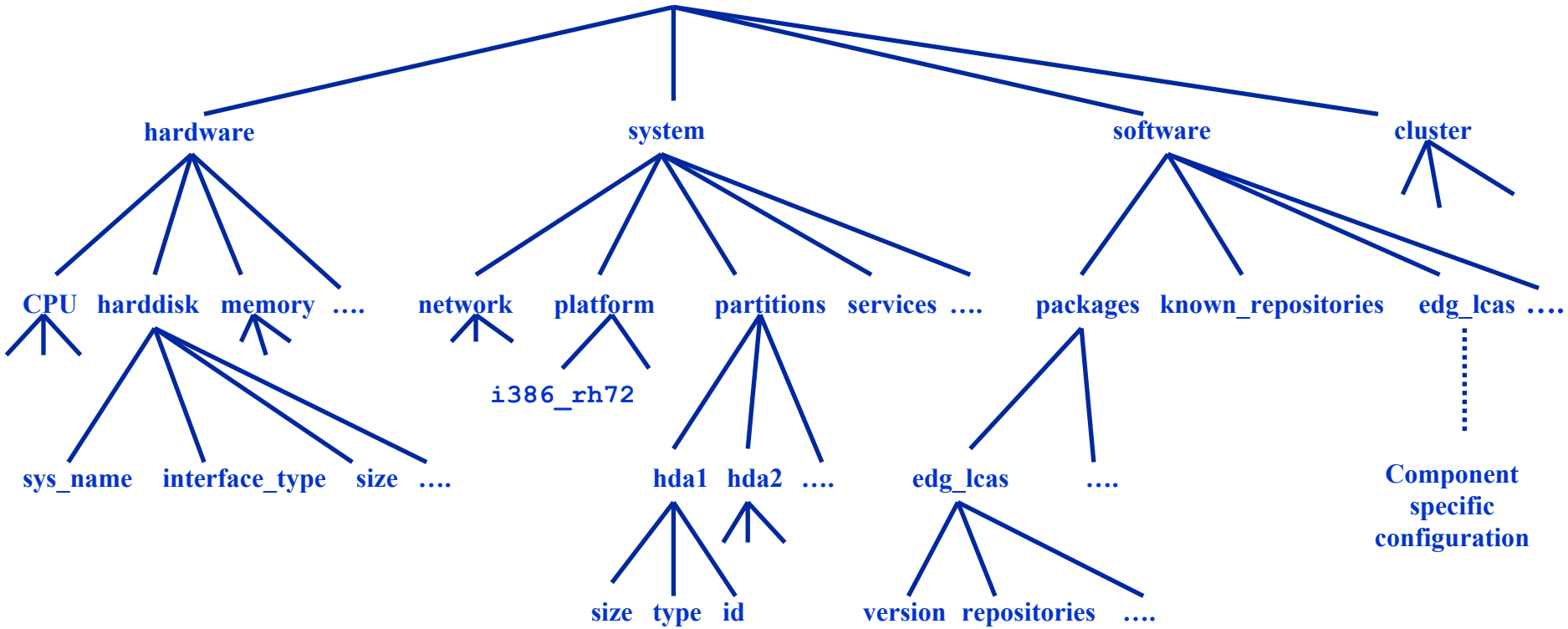
```
<?xml version="1.0" encoding="utf-8" ?>
- <nlist name="profile" derivation="TEST_i386_rh72,node_profile,functions,hardware_types,…

  …..

  - <nlist name="system" derivation="TEST_i386_rh72" type="record">
      <string name="platform" derivation="TEST_i386_rh72">i386_rh72</string>
    - <nlist name="network“
        derivation="TEST_i386_rh72,default_network,network_i386_rh72,std_network“
        type="record">
          <string name="hostname" derivation="functions,std_network">myhost</string>
        - <list name="interfaces" derivation="std_network">
          - <nlist name="0" derivation="std_network_interface,std_network" type="record">
              - <string name="name" derivation="std_network_interface">eth0</string>
              - <string name="ip" derivation="functions,std_network_interface">192.168.0.1</string>
              - <boolean name="onboot" derivation="std_network_interface">true</boolean>
            </nlist>
          </list>
  …..
```

- Description of the High Level Definition Language (HLDL), the compiler and the Low Level Definition Language (LLDL) can be found at: http://cern.ch/hep-proj-grid-fabric-config

Global configuration schema tree

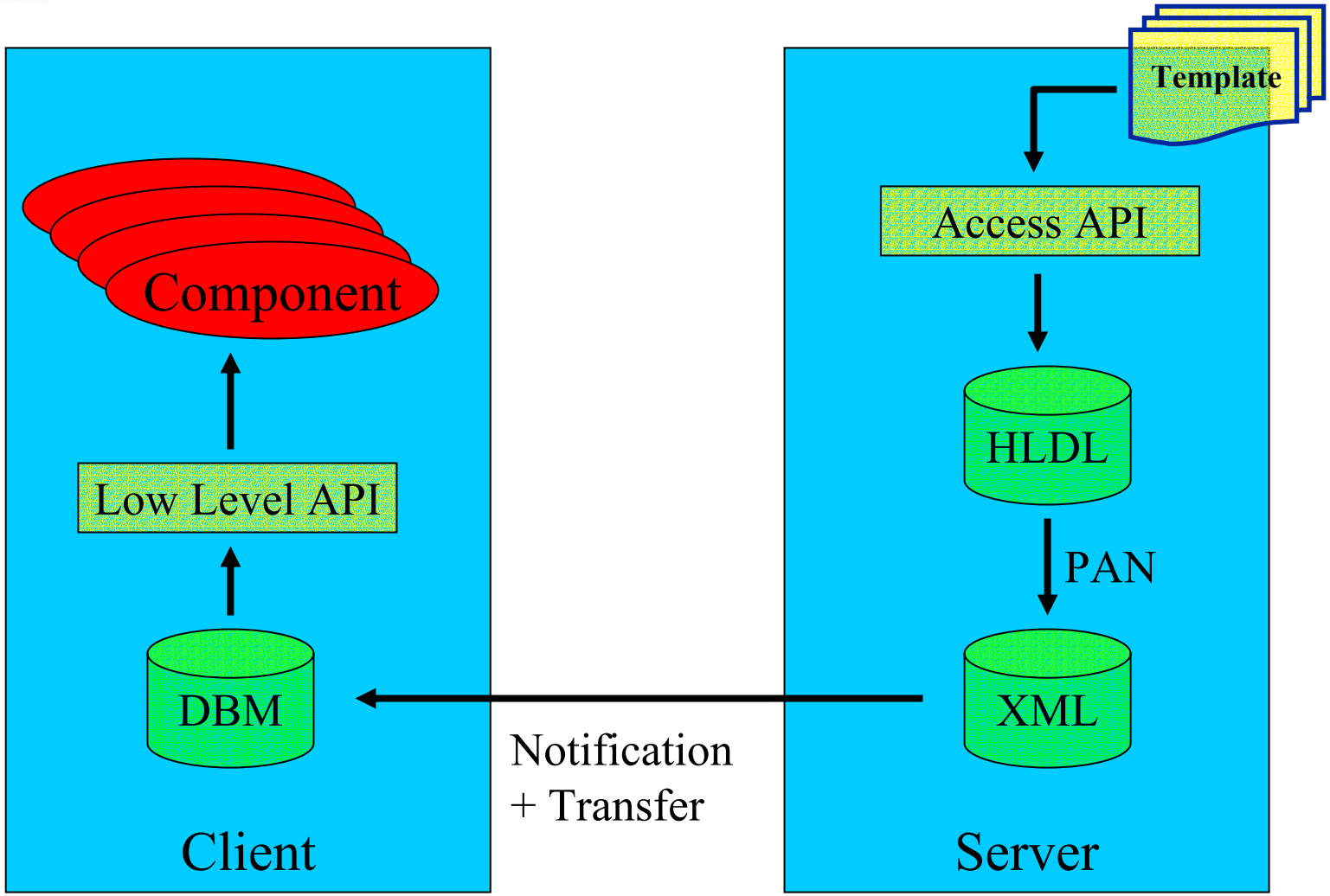


The population of the global schema is an ongoing activity
http://edms.cern.ch/document/352656/1

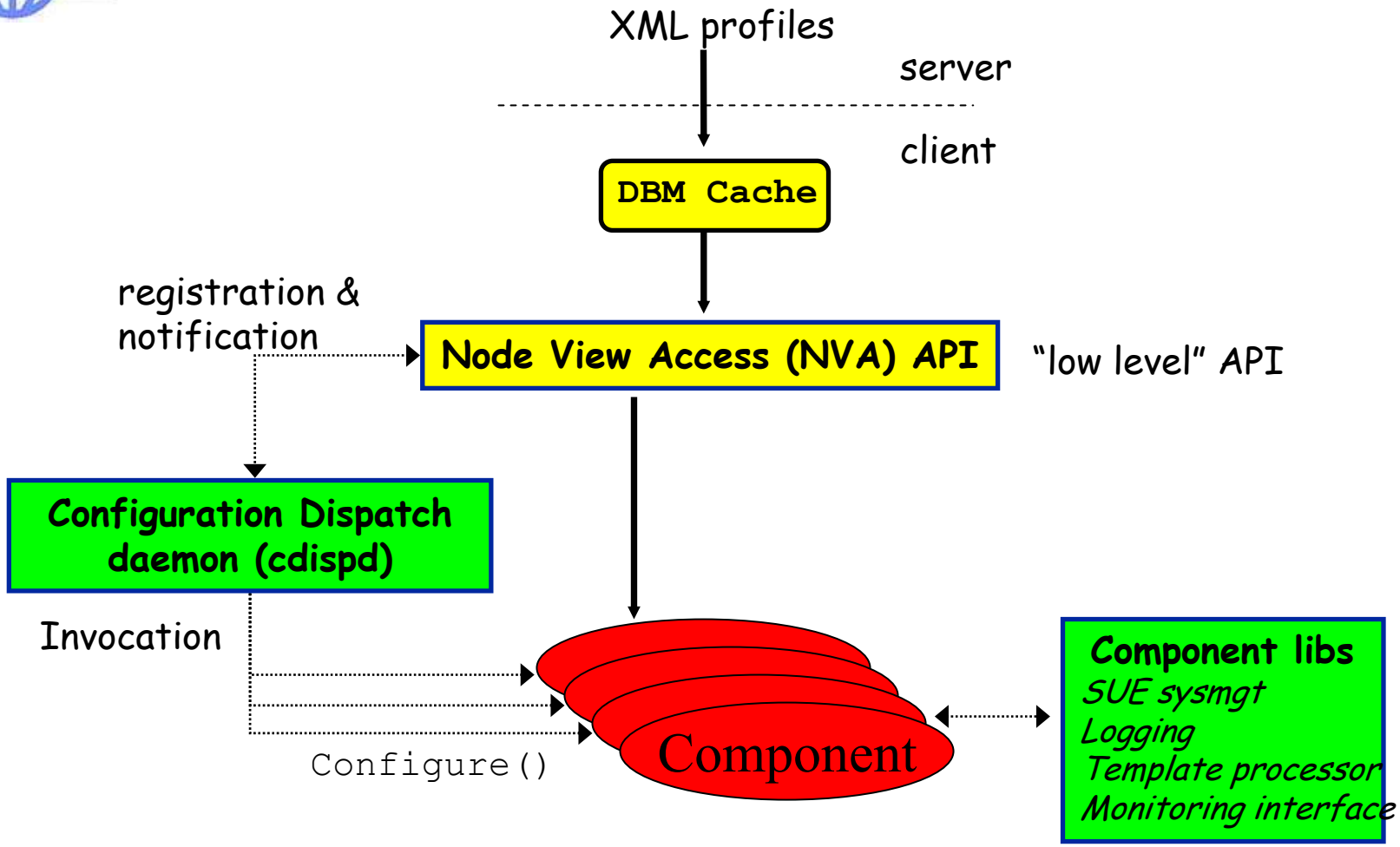
Subtask: installation management

- Node Configuration Deployment
- Base system installation
- Software Package Management

Node configuration deployment



Node configuration deployment infrastructure



Component example

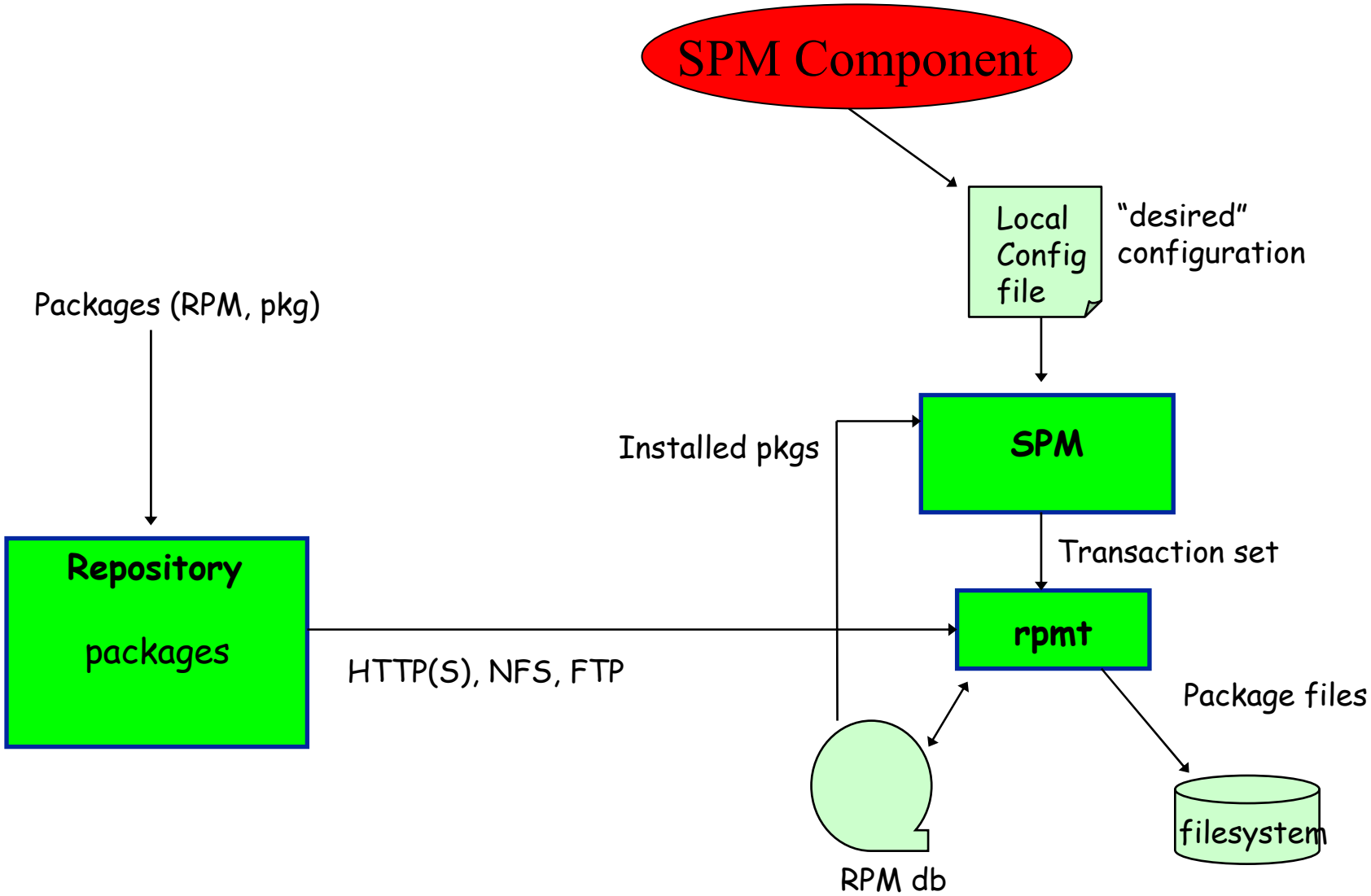
```
sub Configure {
  my ($self) = @_;
  # access configuration information
  my $config=NVA::Config->new();
  my $arch=$config->getValue('/system/platform'); # low-level API
  $self->Fail ("not supported") unless ($arch eq 'i386_rh72');
  # (re)generate and/or update local config file(s)
  open (myconfig,'/etc/myconfig');

      …
  # notify affected (SysV) services if required
  if ($changed) {
    system('/sbin/service myservice reload'); …
  }
}
```

Base Installation and Software Package management

- Use of standard tools
- Base installation
  - Generation of kickstart or jumpstart files from node profile
- Software package management
  - Framework with pluggable packager
    - rpm
    - pkg
    - ??
  - It can be configured to respect locally installed packages, ie. it can be used for managing only a subset of packages on the node (useful for desktops)

Software Package Management (SPM)



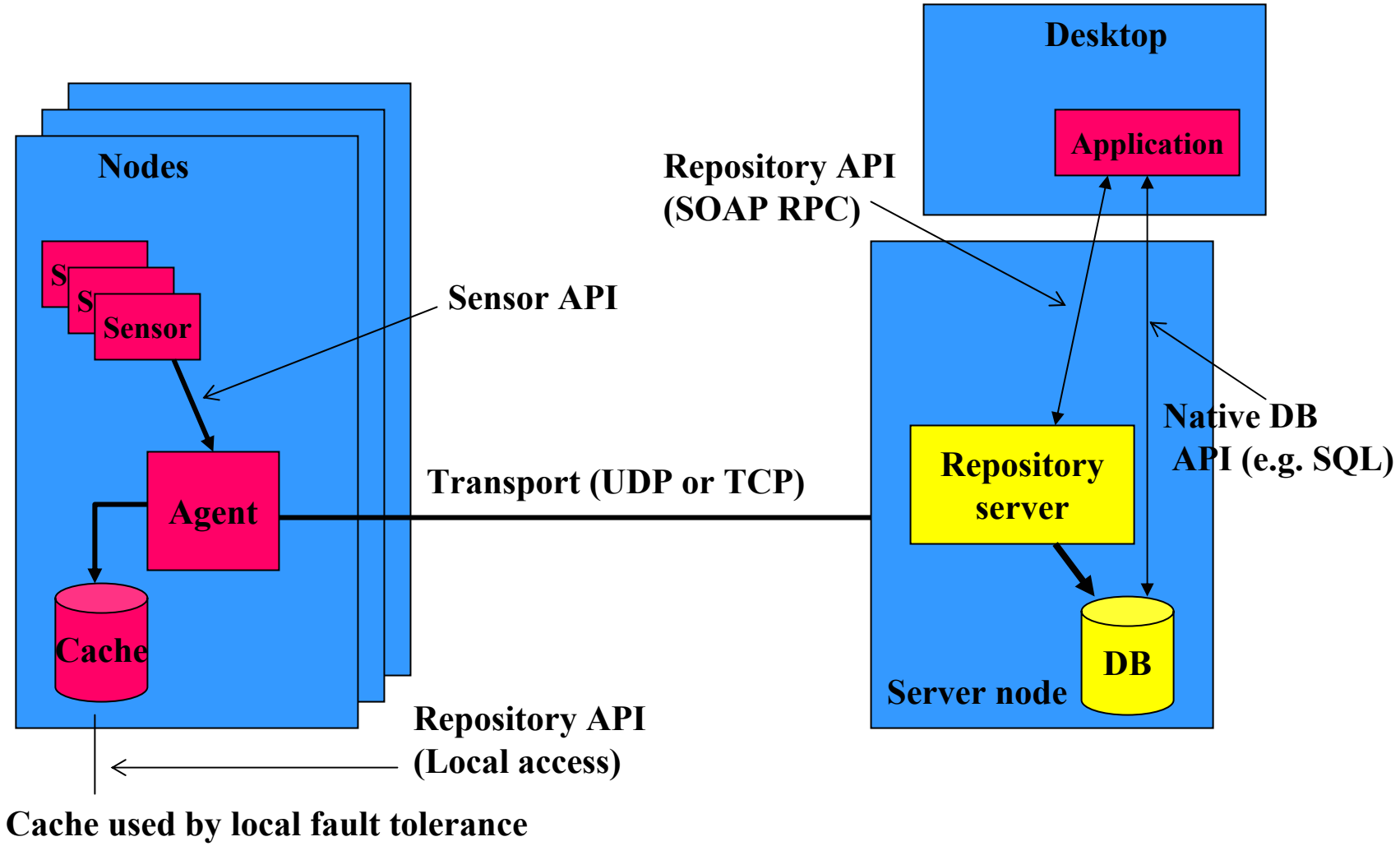
Installation (&configuration): status

- LCFG (Local Configuration) tool from Univ. of Edinburgh has been in production at the EDG testbed since more than 12 months
  - Learned a lot from it to understand what we really want
  - Used at almost all EDG testbed sites → very valuable feedback from a large O(5-10) group of site admins
- Disadvantages with LCFG
  - Enforces a private per component configuration schema
  - High level language lacks possibilities to attach compile time validation
  - Maintains propriety solutions where standards exist (e.g. base installation)
- New developments progress well and complete running system is expected by April 2003

Subtask: fabric monitoring

- Framework for
  - Collecting monitoring information from sensors running on the nodes
  - Store the information in a local buffer
    - Assures that data is collected and stored even if network is down
    - Allows for local fault tolerance
  - Transports the data to a central repository database
    - Allows for global correlations and fault tolerance
    - Facilitate generation of periodic resource utilisation reports
- Status: framework deployed on EDG testbed. Enhancements will come
  - Oracle DB repository backend. MySQL and/or PostgreSQL also planned
  - GUIs: alarm display and data analysis

Fabric monitoring



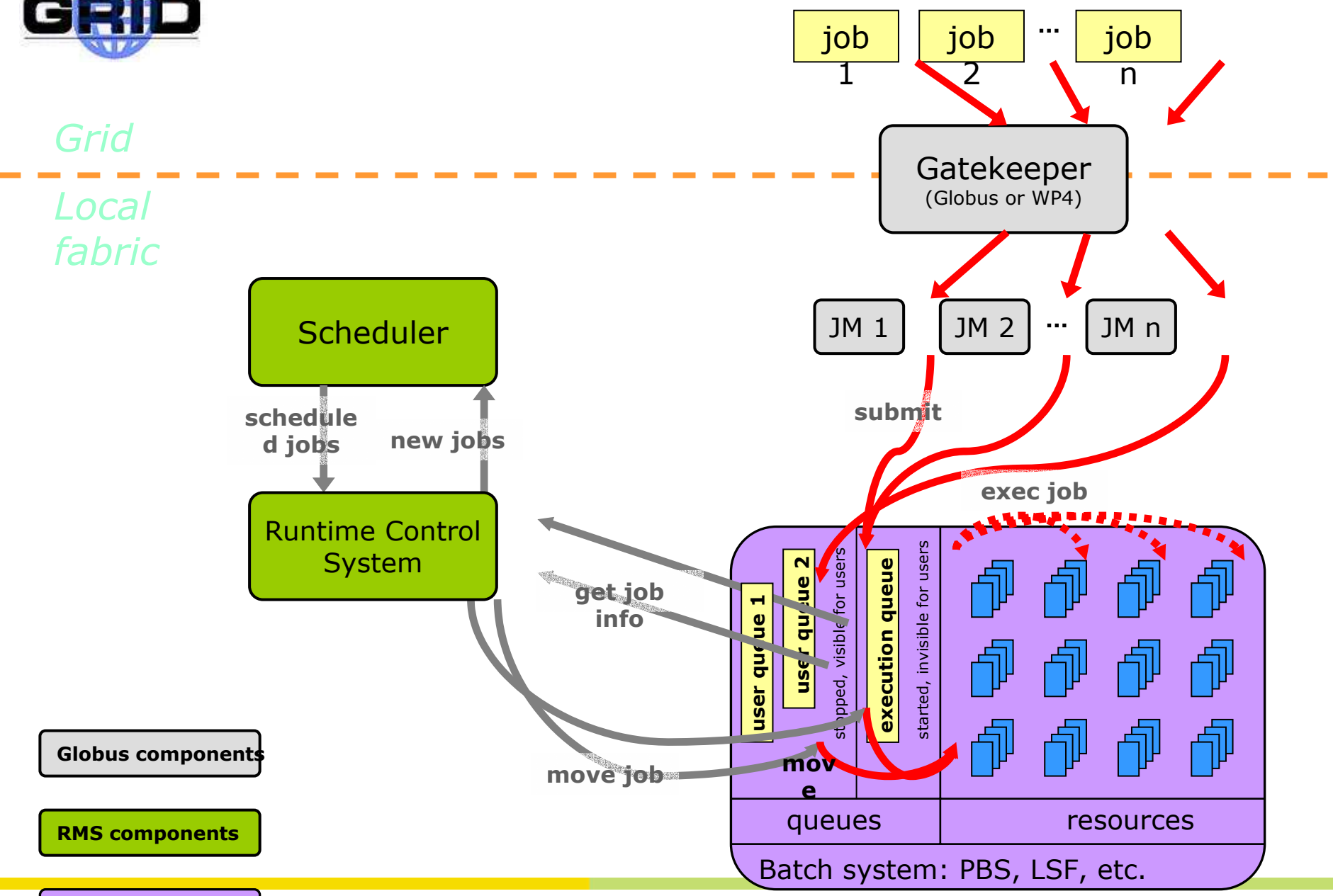
Subtask: fault tolerance

- Framework consists of
  - Rule editor
    - Enter metric correlation algorithms and bind them to actions (actuators)
  - Correlation engines implements the rules
    - Subscribe to the defined set of input metrics
    - Detect exception conditions determined by the correlation algorithms and report to the monitoring system (exception metric)
    - Try out the action(s) and report back the success/failure to the monitoring system (action metric)
  - Actuators
    - Plug-in modules (scripts/programs) implementing the actions
  - Status: first prototype expected by mid-November 2002

Subtask: resource management

- Manage grid jobs and local jobs. Layer between grid scheduler and local batch system. Allows for enhancing scheduling capabilities if necessary
  - Advanced reservations
  - Priorities
- Provides common API for administrating underlying batch system
  - Scheduling of maintenance jobs
  - Draining node/queues from batch jobs
- Status: prototype exists since a couple of months. Not yet deployed on EDG testbed.

Resource management prototype (R1.3)



Subtask: gridification

- Layer between local fabric and the grid
  - Local Centre Authorisation Service, LCAS
    - Framework for local authorisation based on grid certificate and resource specification (job description)
    - Allows for authorisation plug-ins to extend the basic set of authorisation policies (gridmap file, user ban lists, wall-clock time)
  - Local Credential Mapping Service, LCMAPS
    - Framework for mapping authorised user's grid certificates onto local credentials
    - Allows for credential mapping plug-ins. Basic set should include uid mapping and AFS token mapping
  - Job repository
  - Status: LCAS deployed in May 2002. LCMAPS and job repository expected 1Q03.

Conclusions

- Since last LCCWS we have learned a lot
  - We do have an architecture and a plan to implement it
  - Development work is progressing well
  - Adopting LCFG as interim solution was a good thing
    - Experience and feedback with a real tool helps in digging out what people really want
    - Forces middleware providers and users to respect some rules when delivering software
    - Automated configuration has become an important for implementing quality assurance in EDG
  - Internal and external coordination with other WPs and projects result in significant overhead
  - Sociology is an issue (see next 30 slides…)