Authorization and R-GMA

Linda Cornwall

WP3 workshop 2-4 April 2003

What is Authorization?

- Whereby a principal is allowed to or prevented from carrying out an action.
- In edg there are requirements to carry out an authorization decision based on
  - Specific DN
  - VO membership(s)
  - Role within VO
  - Group within VO
  - By allowing anyone with an acceptable certificate to carry out an action
  - Allowing anyone to carry out an action

Authorization – EDG Principle

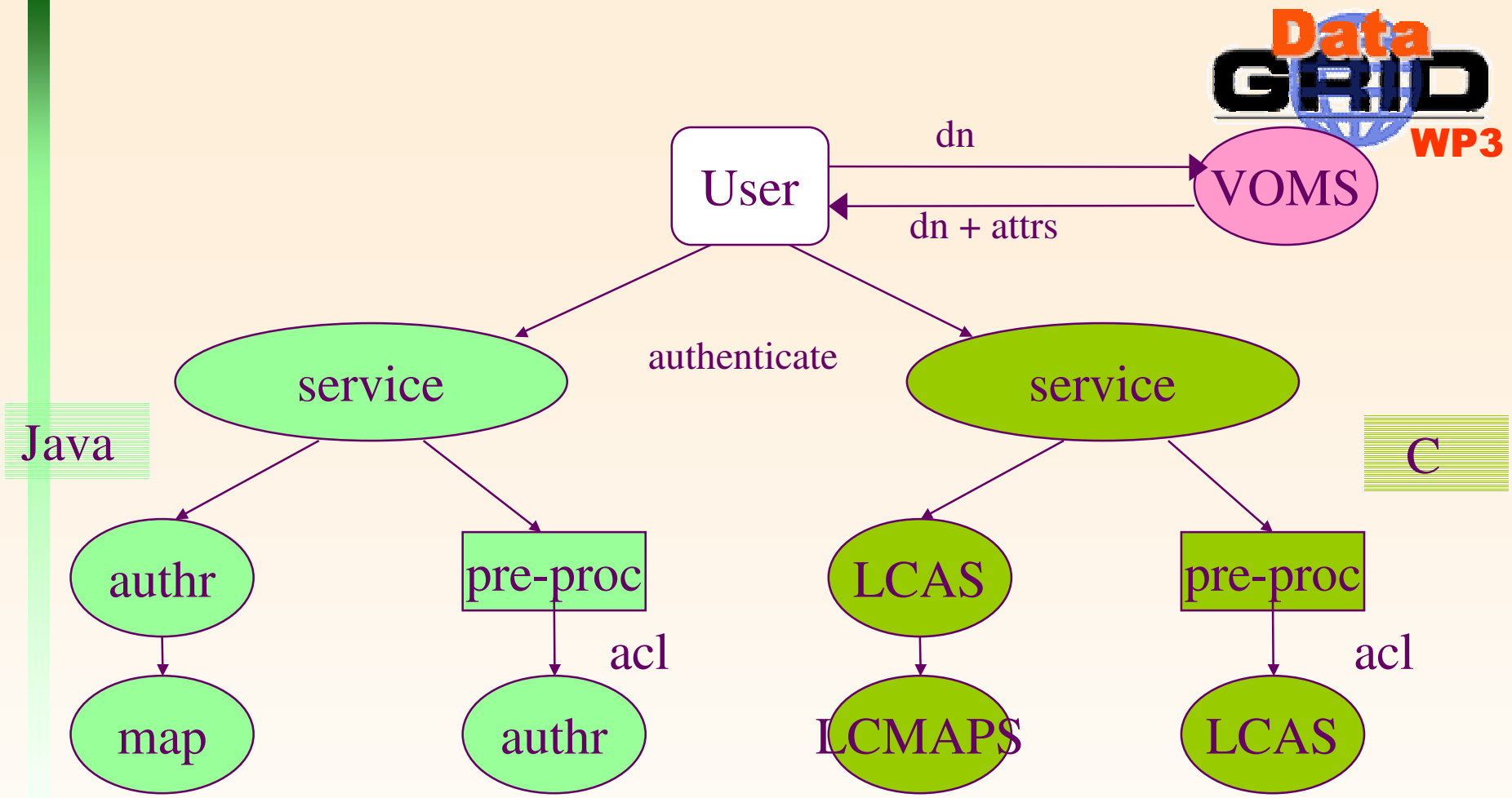
- Principle within EDG is that Authorization information goes with the resource or data.
- The authorization decision is made close to the resource or data, based on a combination of local Authorization information and attributes from the user
- This enables e.g. resource owner or administrator, or a file owner or administrator to keep control over it's access.
- Hence e.g. a producer of information must make the decision as to whether or not the requestor of that information can have that information.
- Hence the end producer of the information must have information on the DN, VO membership, role(s) and groups of the end client in order to make an authorization decision.

VOMS certificate

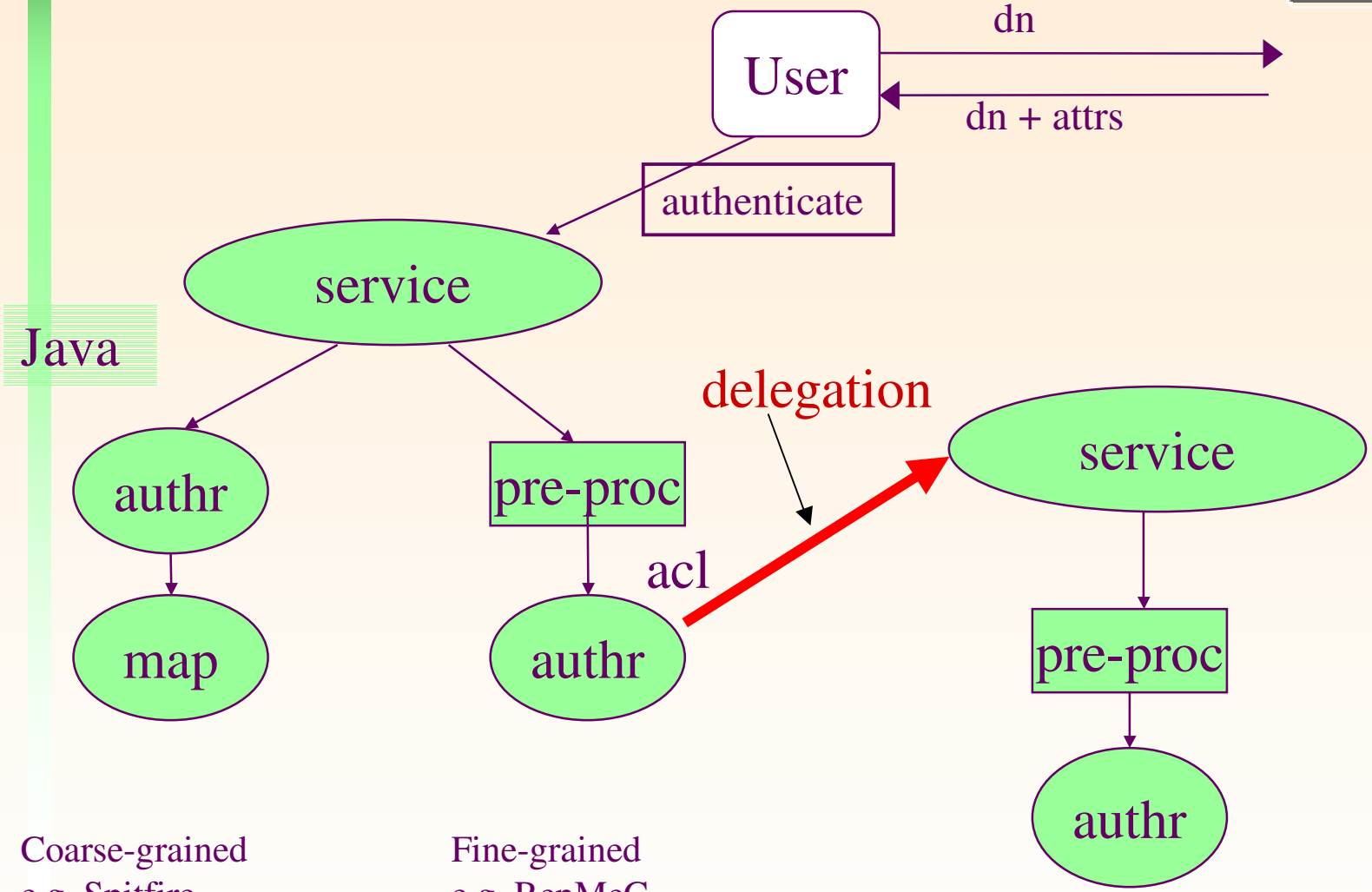
- Voms certificate contains proof that a user is a member of a VO, is a member of certain groups in a VO, and has certain roles within a VO.
- A delegated VOMS certificate means that any servlet that is connected onto has proof of an entity's VO membership, groups and roles as well as DN.
- Then an authorization decision can be made.
- Without a delegated VOMS certificate – authorization is not really secure – any hacked consumer can do what they like.

Course grained vs fine grained authorization

- Course grained – authorization on front of service (I.e. y/n can the person connect).
  - "Is the user allowed to use this service – if so – what role"
- Fine grained – authorization takes place within a service.
  - E.g. can this user read this file?
- R-GMA could have services which decide whether or not a connection is allowed, as well as services which decide whether to satisfy the request within the service.
- I favour ALL authorization decisions being made within services for R-GMA – a combination will be rather cumbersome.



Coarse-grained e.g. Spitfire	Fine-grained e.g. RepMeC	Coarse-grained e.g. CE, Gatekeeper	Fine-grained e.g. SE, /grid
WP2	WP2/WP3	WP4	WP5

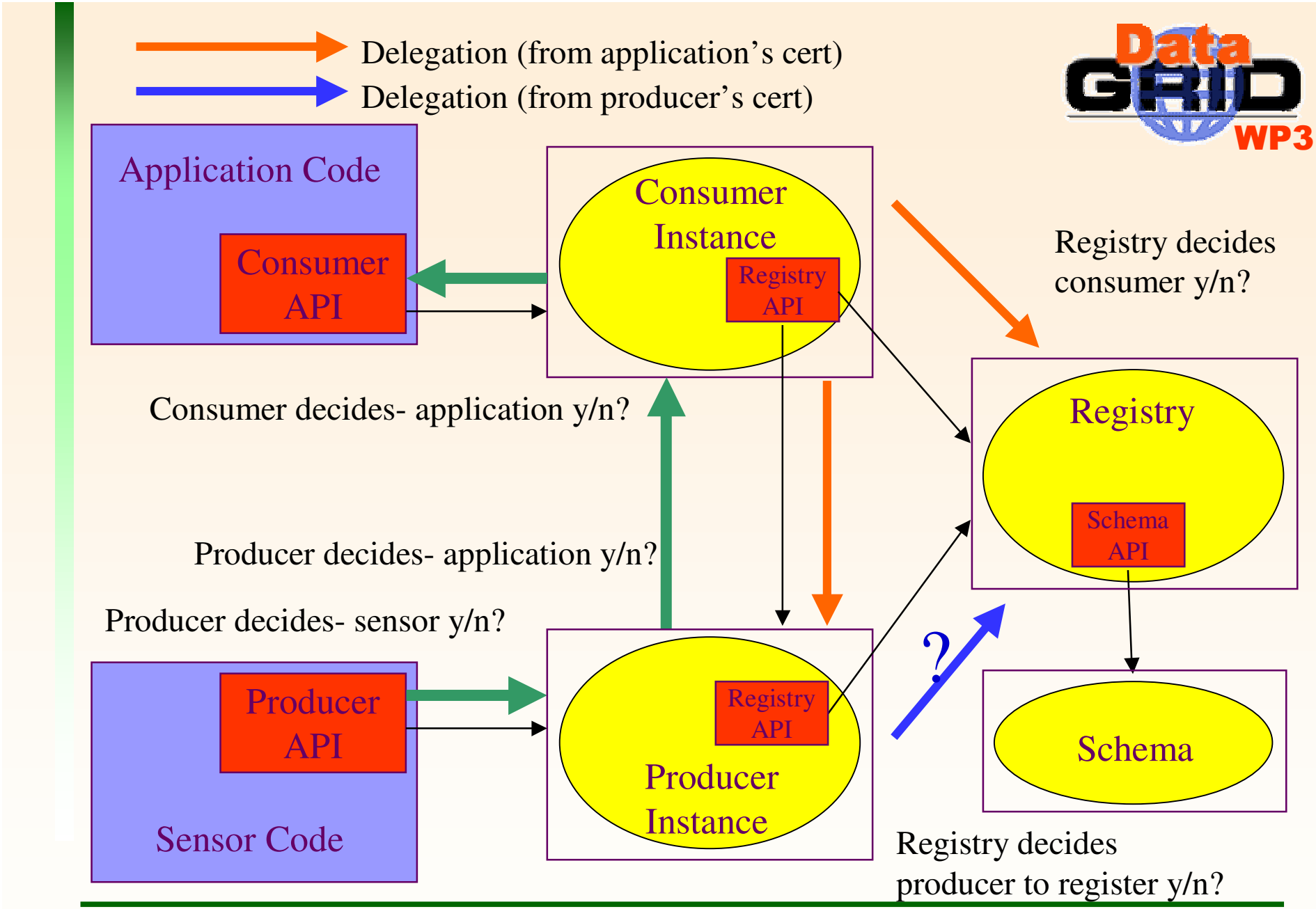


Coarse-grained
e.g. Spitfire

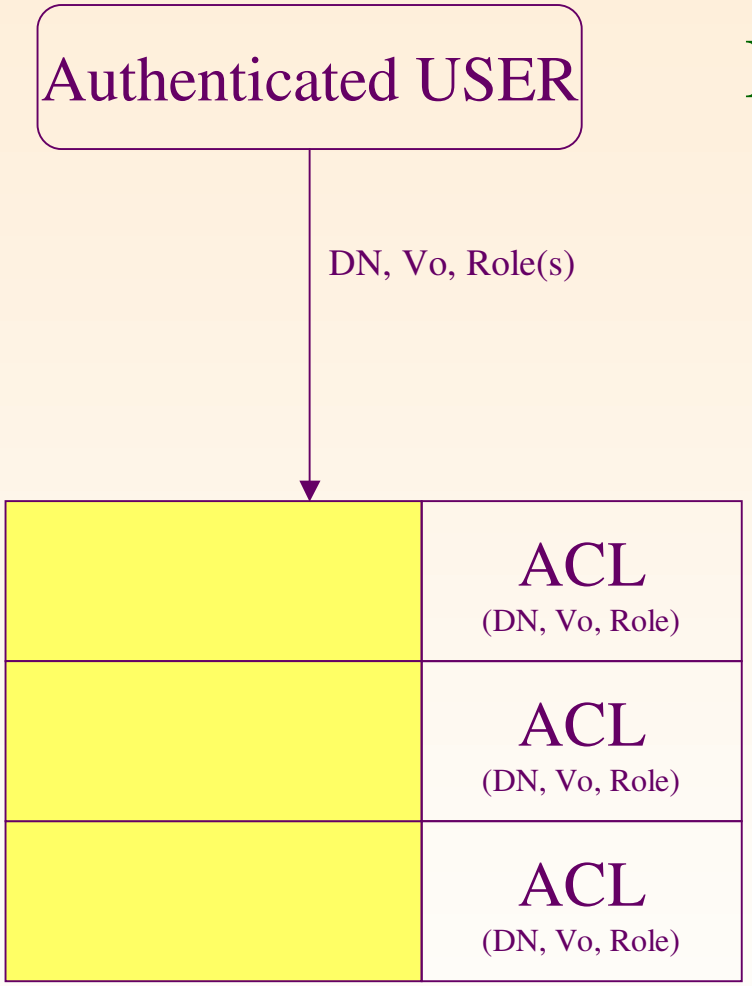
WP2

Fine-grained
e.g. RepMeC

WP2/WP3


Delegation (from application's cert)
Delegation (from producer's cert)
DataGRID WP3
Application Code
Consumer API
Consumer Instance
Registry API
Registry decides consumer y/n?
Consumer decides- application y/n?
Registry
Schema API
Producer decides- application y/n?
Producer decides- sensor y/n?
Producer API
Sensor Code
Registry API
Producer Instance
?
Schema
Registry decides producer to register y/n?

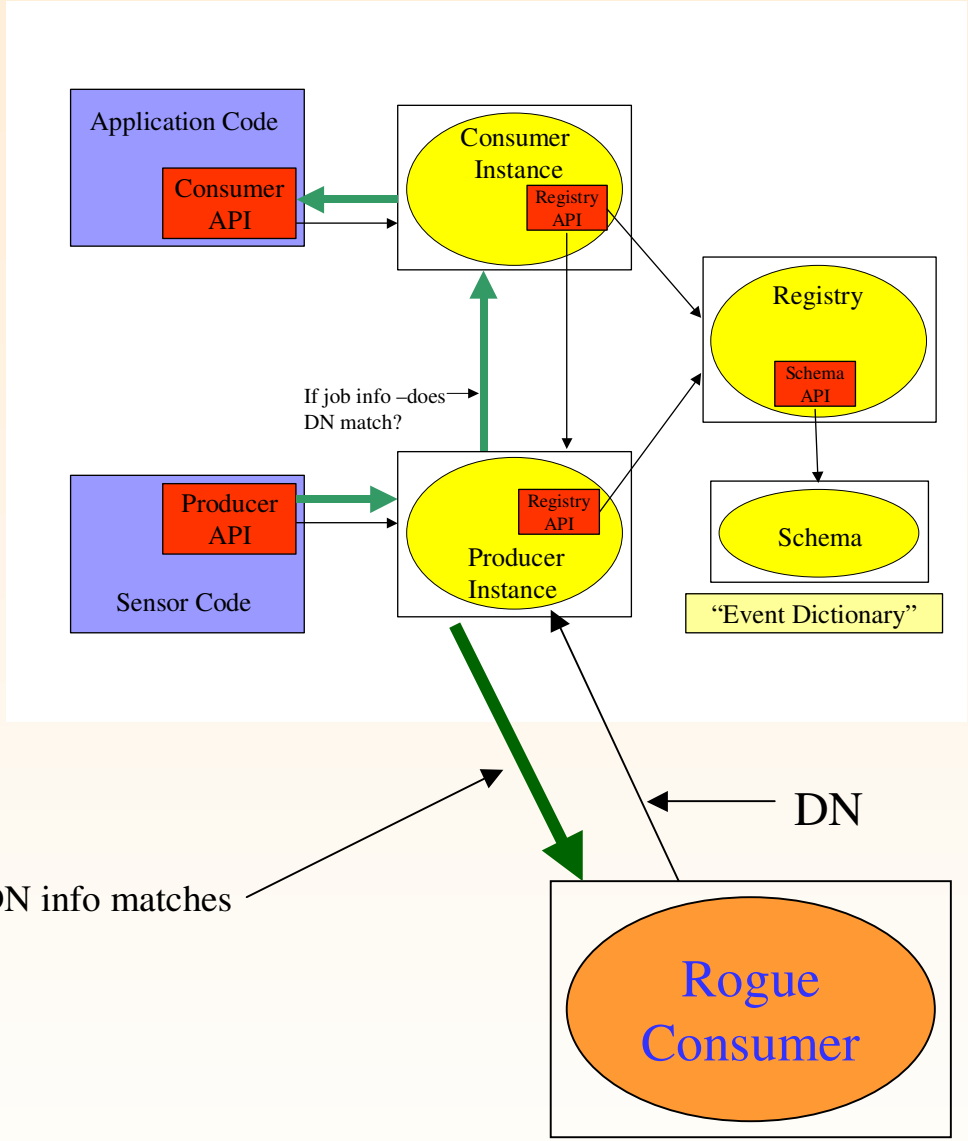
Fine grained R-GMA



In fine grained Authorization a decision must be made within the producer. Each table, even each row of a table will have it's own ACL

Confidentiality

- There are certain requirements on confidentiality. To satisfy these an authorization decision at the source of info AND a delegated VOMS proxy is needed.
- If a third party can say ‘tell me if Linda is banned’ without the use of a delegated certificate – then the fact Linda is banned can be found out without Linda’s permission.
- Similarly for any info – a hacked or rogue R-GMA can get any info they want. Can only make things difficult.



Without Delegation it is possible to obtain info one is not authorized to see. But it requires a consumer to be hacked or written.

Rogue Consumer has acceptable Certificate.

Some WP3 specific requirements

- It must be possible to restrict knowledge of the existence of producers of information to specific authorized users
  - Solution – authorization information goes into the registry – only those authorized are granted info on the producer
- A producer must be able to restrict the publishing of information to specific authorized users.
  - Solution – each time a user attempts to publish – check against a list of users.

Specific WP3 requirements - contd

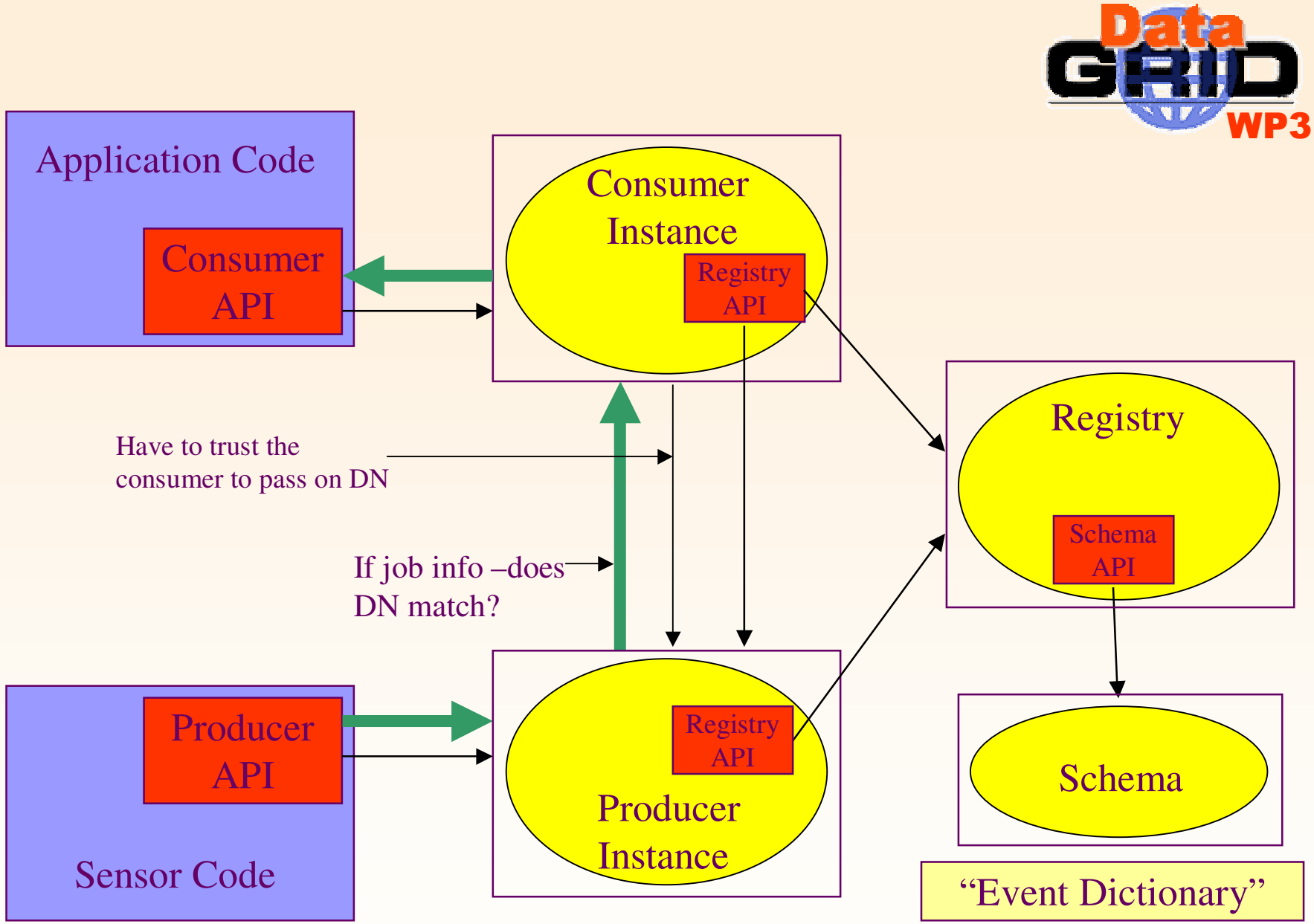
- A user can only see information on their own job
  - Solution – when job control info is requested, the producer only passes the info back if dn matches dn.
- A producer must be able to restrict read access to information to specific authorized users.
  - Solution – add authorization info into the table.
  - This could be a generalization of above

Suggested implementation solution

- Each Servlet has associated with it a file which defines authorization for each type of operation.
- Each table has an indicator as to whether authorization is per table or per row.
- If per table – indicate the name of the authorization file for this table – possibly indicate DN must match.
- If per row, each row contains authorization info – possibly a pointer to a table

Authorization without delegation

- First just have user's only seeing their own jobs.
- Only way to do this in the absence of delegation in the trustmanager is
  - extract the DN of the user when they connect to the consumer
  - Pass this onto the producer
  - Add to job info table that DN must match
  - Only allow the user to see the job if DN matches
- Not secure – anyone with an acceptable certificate could get the info – but they would have to write their own consumer code

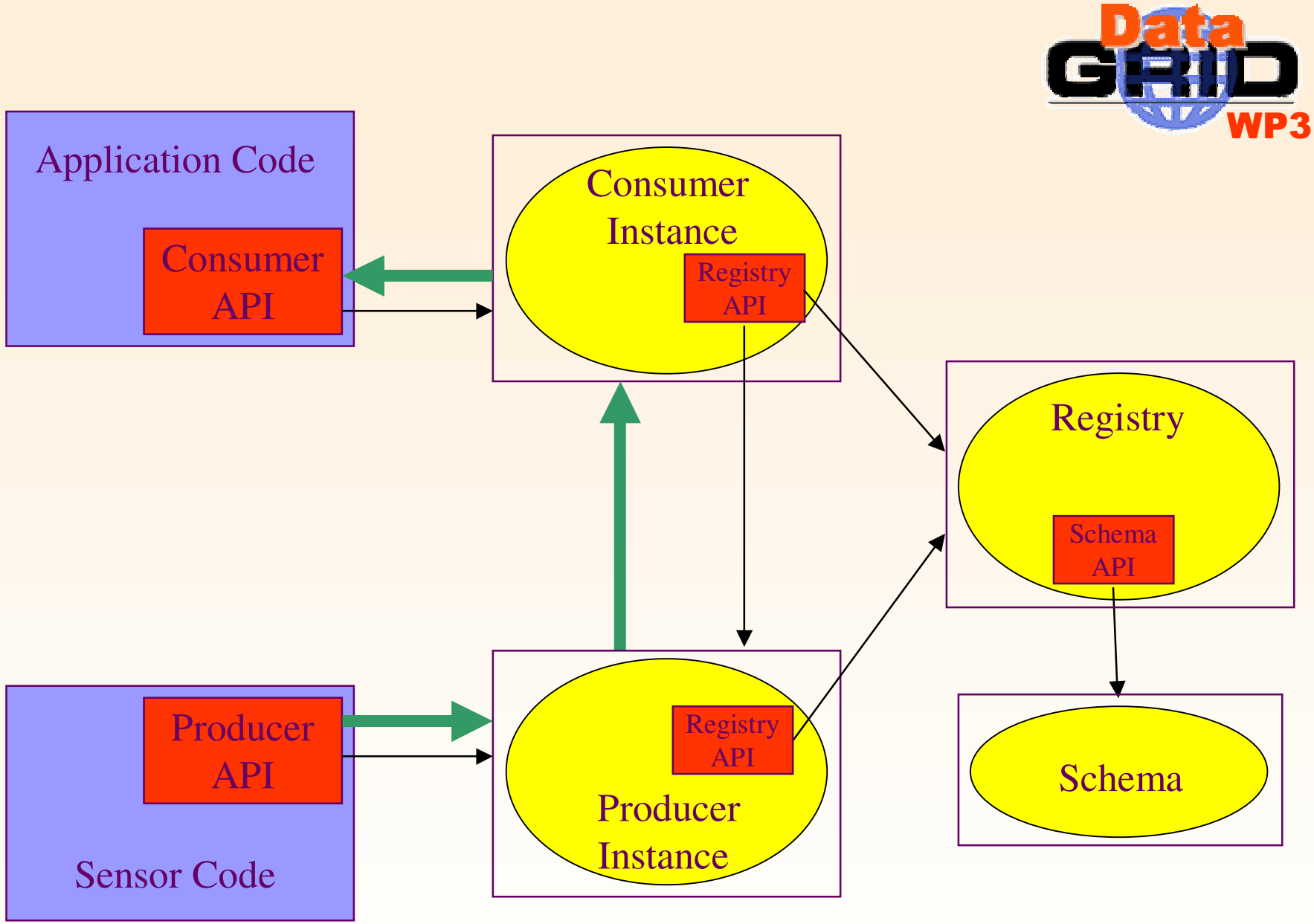

Application Code
Consumer API
Consumer Instance
Registry API
Have to trust the consumer to pass on DN
If job info –does DN match?
Registry
Schema API
Producer API
Sensor Code
Registry API
Producer Instance
Schema
"Event Dictionary"

Evolutionary approach

- 1st carry out a very crude system for authorization without delegation – until delegation is available. This will allow to try out the DN extracting and matching, having a first stage of adding to table.
- Replace with delegation as soon as available
- Then start looking at how to formulate what are effectively the ACL's. Possibly use GACL.
- Look at what to add to the schema tables – pointer such as no Authorization, DN match, pointer to table.
- Steve Fisher has ideas on details.

Mixing secure and non-secure access

- Allowing different access rules after authentication fits quite well into R-GMA.
- Allowing a mixture of secure and non secure access is more difficult.
  - Authentication happens before connection
  - Authorization happens within the servlet
  - So for non-secure access – would need a different copy of R-GMA on a different port.
  - Problem is for us – 1 R-gma does everything.
  - Ideally, would need different service for non-secure access, I.e. a cut down R-GMA which just allows certain things.


Application Code
Consumer API
Consumer Instance
Registry API
Registry
Schema API
Schema
Producer API
Sensor Code
Registry API
Producer Instance