European Commission

Information Society Technologies

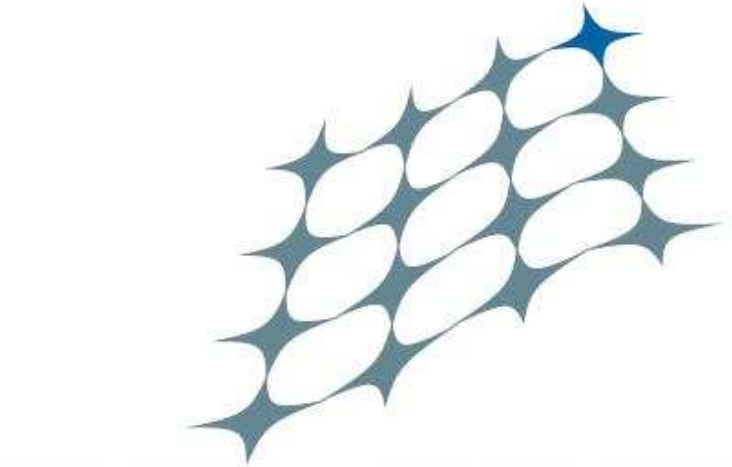
BEinGRID components: Methodology and SLA example

Igor Rosenberg

Barcelona, September 23rd, 2009

Structure of the presentation

- **The BEinGRID project**
- **The methodology involving Bes**
- **The SLA results**



European Commission

Information Society
Technologies

The BEinGRID project

BEinGrid Project Data Sheet

- **Type of project:** Integrated Project
- **Project coordinator:** Mr. Santi Ristol
  santi.ristol@atosorigin.com
  (ATOS ORIGIN)
- **Project start date:** 1st June 2006
- **Duration:** 42 months (until Nov 2009)
- **Budget:** 24.7 M Euros
- **Max EC contribution:** 15.7 M Euros (63%)
- **Consortium:** 75 + 23 partners
- **Effort:** 2713 PM (226 PY,65 P,360.000h)

The mission of BEINGRID is to Exploit European Grid middleware by creating a *toolset repository* of Grid services from across the Grid research domain and to use these services to deliver a set of *successful business experiments* that stimulate the early adoption of Grid technologies across the European Union.

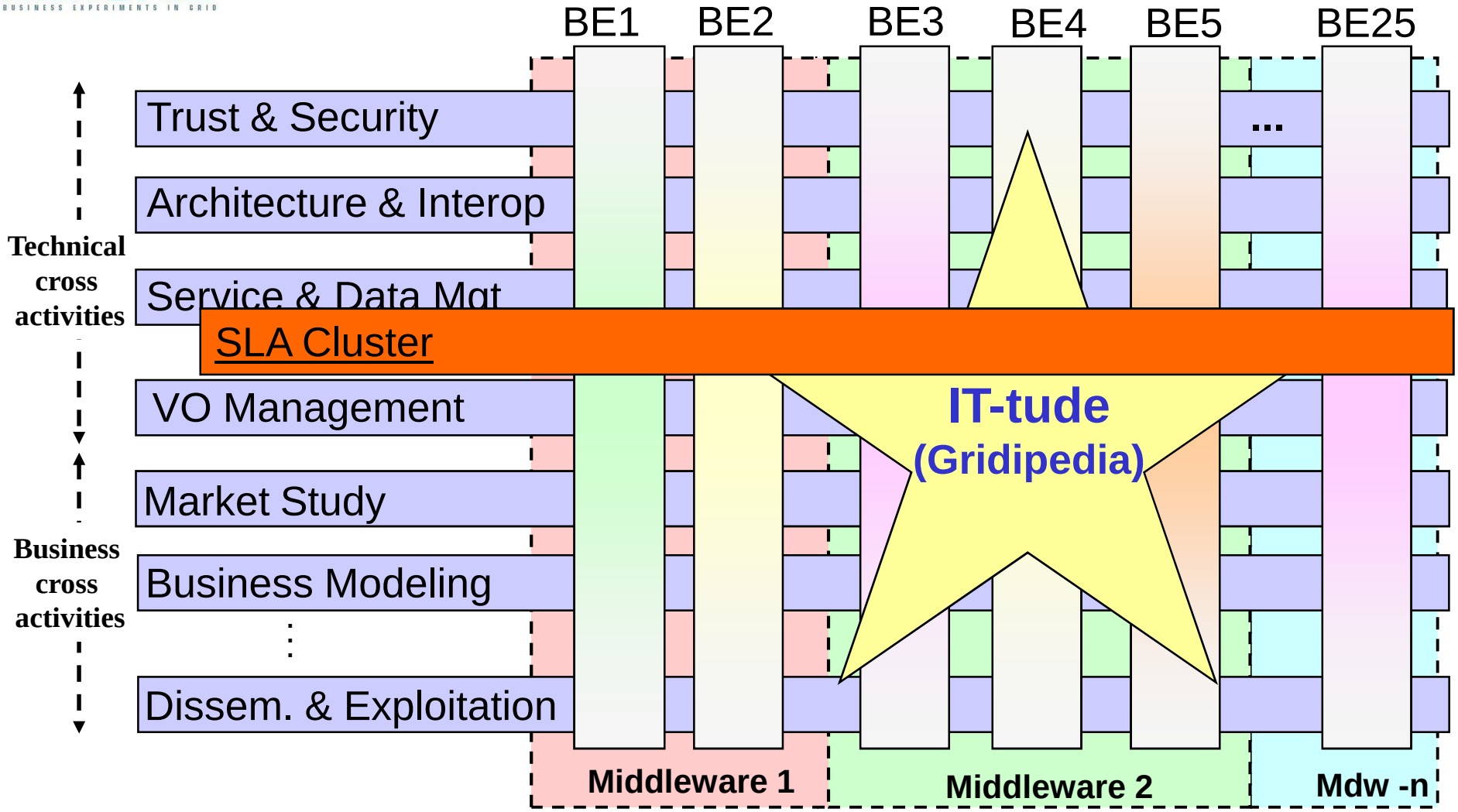
BEinGrid at a glance

- **18 + 6 Business experiences** **Real world!**
  - **Service Provider + Integrator + End User**
  - **12 different sectors : retailing, architecture, textile, finance, …**
  - **Two BE waves: first 06/2006-06/2008, second 03/2008-03/2009**

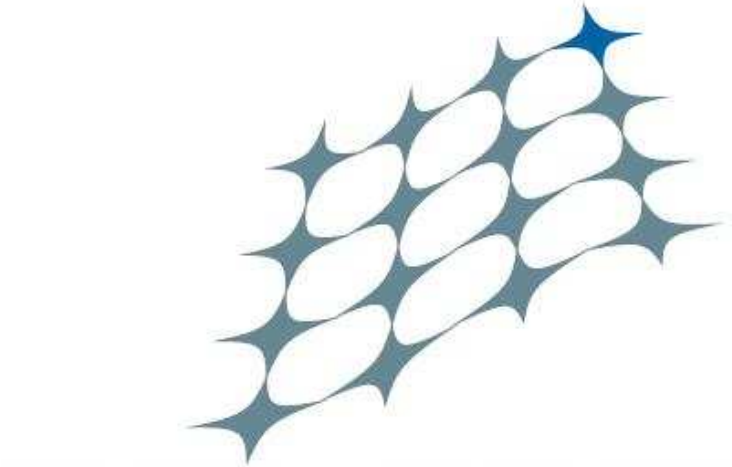
- **Cross activities**
  - **Analysis of the ongoing experiments**
  - **Technical “clusters” (security, portals, VO, SLA, …)**
  - **Business**

- **Gridipedia (General Repository)** ***Now called IT-tude!***
  - **Documents (designs, howtos, success stories, …)**
  - **Software marketplace (generic components, middleware, …)**

BEinGRID Approach



Selected branches: GTv4, UNICORE/GS, g-Lite, GRIA, WS-*



European Commission

Information Society Technologies

The methodology, (focus on the BEs)

Methodology (1/4)

- **A set of technical requirements defining needs expressed by the BEs, abstracted from the exact Use Case**
  - Capture the essence of several requirements (two or more BEs)
  - Abstract away the specific application context
  - Seek to meet generic challenges that underpin common technical challenges across multiple BEs.

- **Prioritise the requirements:**
  - Popularity (number of BEs covered)
  - Technical innovation
  - Business value (usefulness to achieve BE goals)
  - Dependencies with other requirements

Methodology (2/4)

- **Identification & analysis of common problems**
  - Common problem matches to common requirements among a BE cluster
  - Common problem matches the scope, expertise & interests of the corresponding technical people performing the analysis
  - Dependencies between common problem descriptions are made explicit

- **A set of design patterns sketching a solution to each problem**
  - Scope of the design pattern, identification of common components, specification interactions between them
  - Explain how design pattern relates to the problem
  - Explain how it can be embodied in 2-3 representative BEs

- **Guidance of how to apply the solution to the context of BEs**
  - Explain how the design enhances the BE
  - Explain what the BE needs to adapt/extend
  - Explain what the BE needs to avoid doing

Methodology (3/4)

- **A set of implementation patterns sketching an implementation of selected design patterns**
  - Depend on the execution platform selected
  - Explain how the platform specific components selected match the design pattern
  - Explain how the problem description is addressed
  - Detail interoperability issues

- **A set of ready-to-deploy components per implementation pattern**
  - Detail installation guidelines
  - Detail deployment configuration
  - Use common code repository

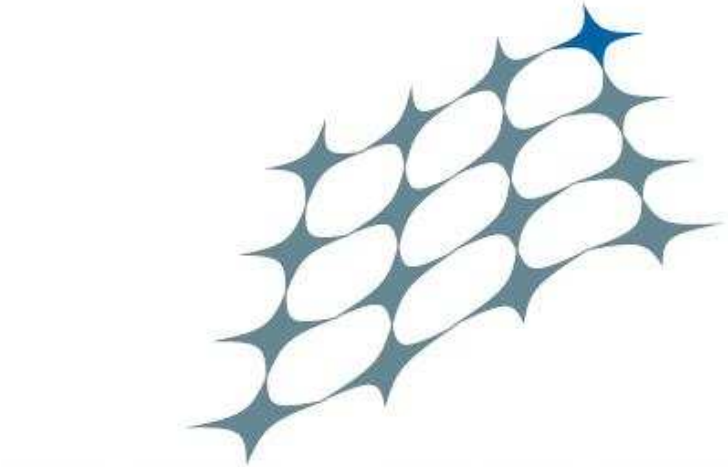
- **Consistent and visible documentation**
  - Use common documentation mechanism
  - Include simplified unit testing & integration testing information

Methodology (4/4)

- **A validation of the components with new BEs**
  - Second wave BEs must validate at least 2 components
  - Should report how the component adapts to the new Use Case, validating its genericity
  - Can extend the component to create extra functionality.

- **A generic component exploitation plan**
  - Can encompass several components
  - Based on collaboration between business analysts and software developers
  - Broad market analysis

- **A per-company exploitation plan**
  - Integration into a company product
  - Confidential
  - Focused market analysis



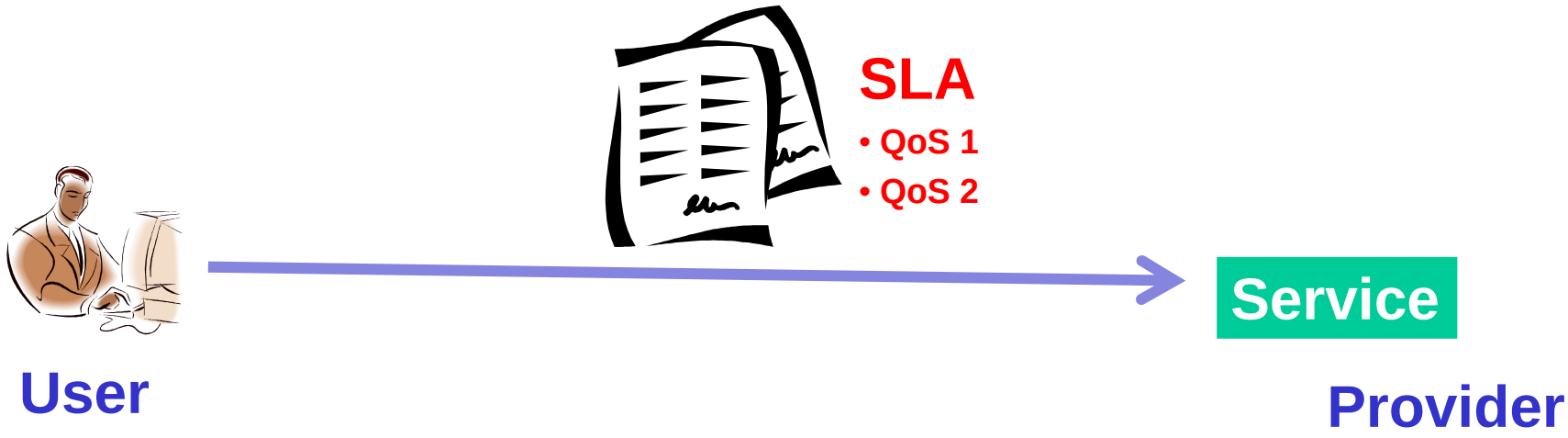
European Commission

Information Society Technologies

Application of the methodology to the SLA cluster

SLA definitions

- **QoS: Quality of Service.**
  - **A set of metrics to be achieved during the service provision**
- **SLA: Service Level Agreement.**
  - **A contract which defines the QoS of the service provided. Allows all actors to narrow the expectations**

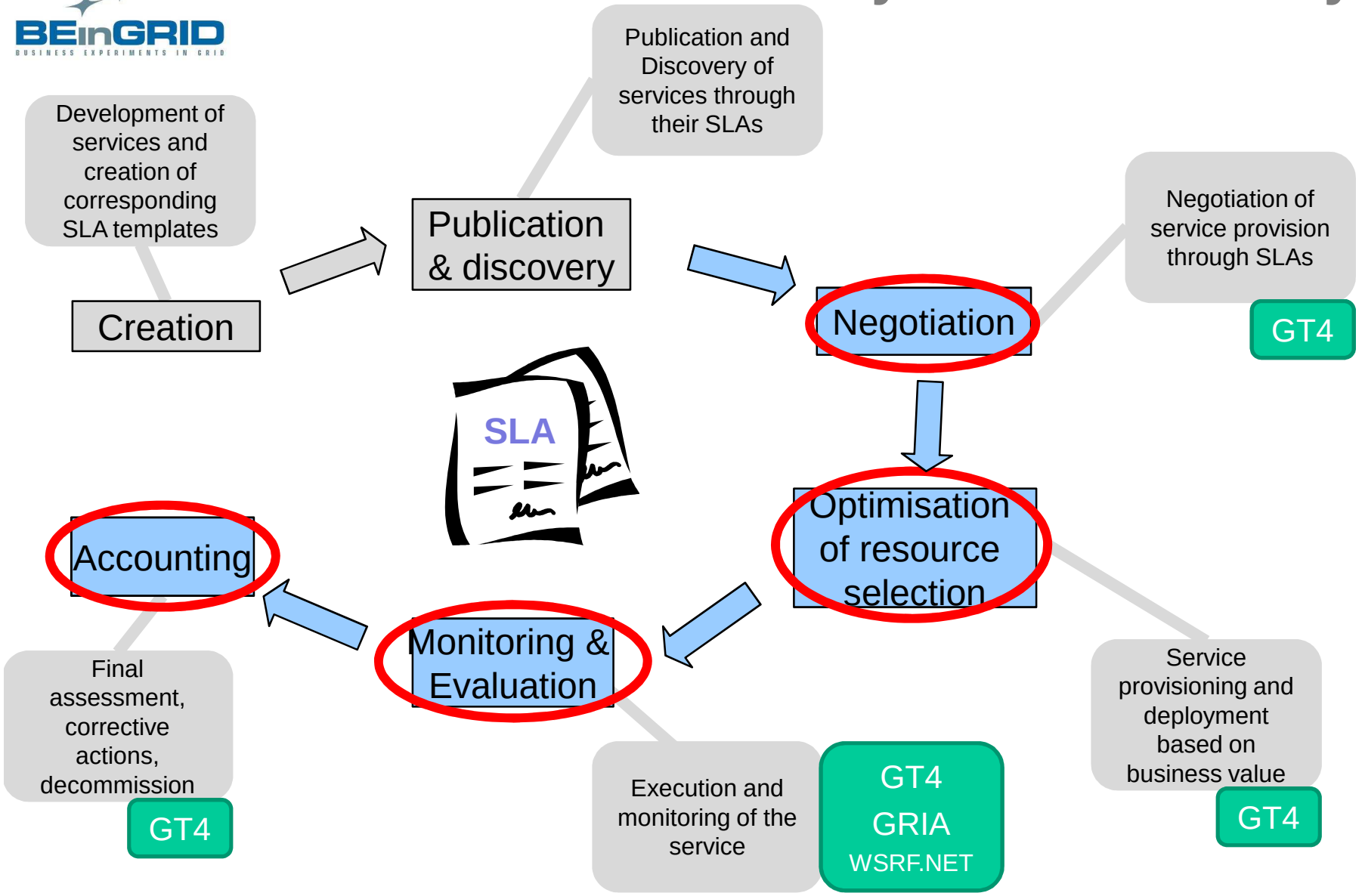


Use of SLAs in Business Experiments

Very dependent on the use case.

- **BE09 : online game platform**
  - **SLA to govern game creation and execution**
  - **network speed, RAM, CPU, and nb players**
- **BE25 : computation for eHealth**
  - **SLA for peak demand outsourced to external provider**
  - **Resource availability and job execution time**
- **BE20 : Telecom Fraud**
  - **SLA for data rate exchange**
  - **Data exchange frequency**
- **BE19&BE21 : relying on the GRIA SLA framework**
  - **SLAs as a service measure**
  - **control service access**

Overall Functionality: the SLA Life Cycle



SLA Cluster General Presentation

The Service Level Agreement Cluster considers the typical SLA lifecycle:

- Stage 1: Development of a service and creation of SLA templates for this service
- Stage 2: Discovery and negotiation of an SLA
- Stage 3: Service provisioning and deployment
- Stage 4: Execution of the service
- Stage 5: Assessment and corrective actions (when necessary)
- Stage 6: Decommission of the service

Aim of the cluster:

- Discover requirements √
- Sort by importance √
- Produce design patterns √
- Produce Best Practices Report √
- Describe exploitation scenario √
- Produce generic components:
  - Negotiation, √
  - Optimisation √
  - Eval&Monitoring, √
  - Accounting, √
  - SLA Framework, √

SLA Cluster Main Results

Justification

- SLAs allow finer service control
- But functionality was unavailable for BEs at start of project.

4 software components, directly following the Design patterns, developed entirely inside BEinGRID

- SLA Negotiation
- SLA Evaluator
- SLA Resource Selection Optimizer
- SLA Accounting

Lessons learned

- Renegotiation is still not clearly needed
- Dynamic SLAs for services are not widely accepted, still in the “early adopters” phase. End-Users are still reluctant to sign dynamic SLAs, but on-demand resource provision based on SLAs can be done between providers to accommodate peak demand.

Best Practices

- Plan your SLA usage in the early stages: generating completely dynamic contracts is too difficult, or cumbersome. A consensus pre-SLA can help limit the bounds for new SLAs.
- SLA must be provided as an architecture, for example lifecycle management for GRIA or GT4 (a BEinGRID development)

Business Benefits and Innovation Impact

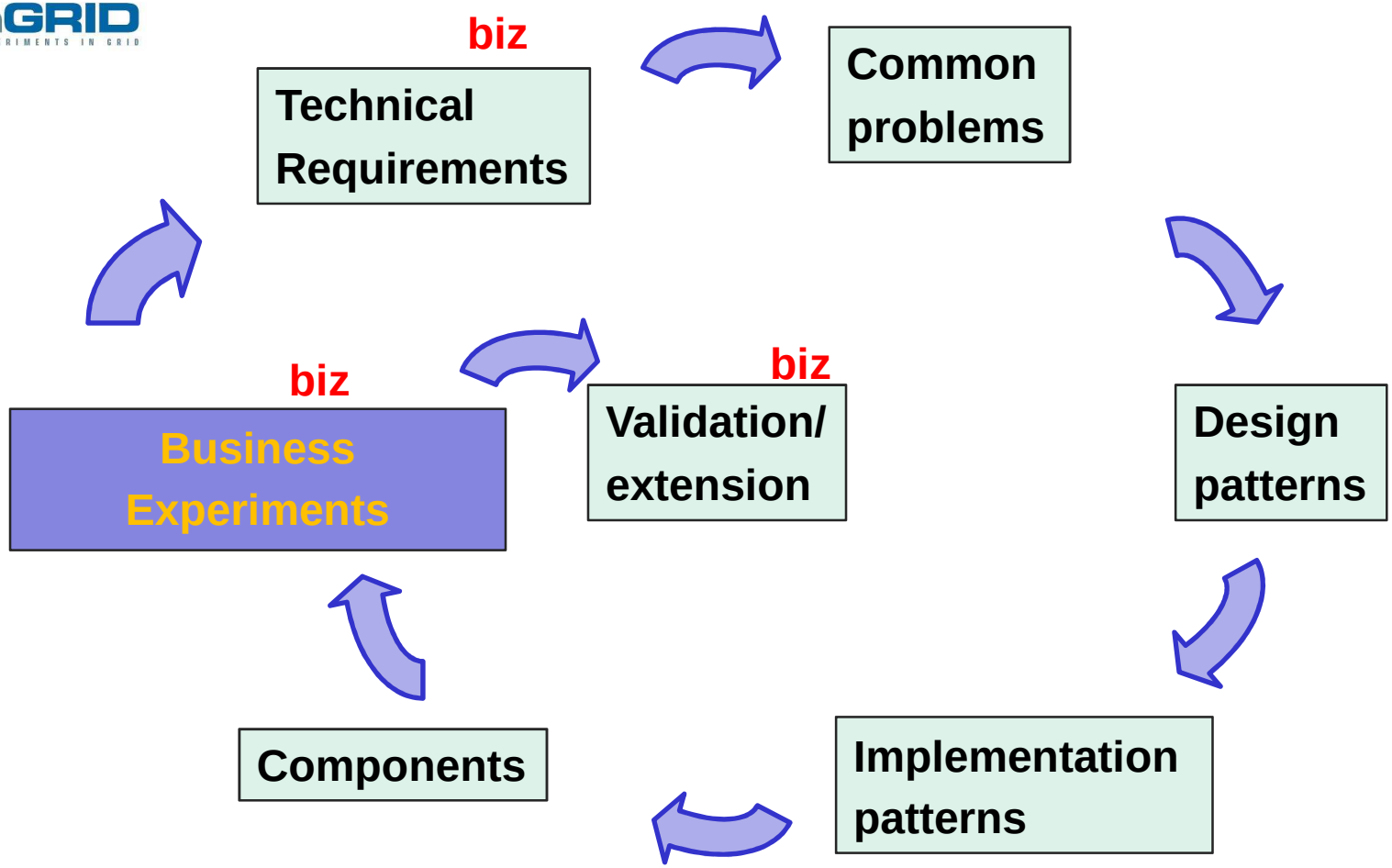
Business Benefits

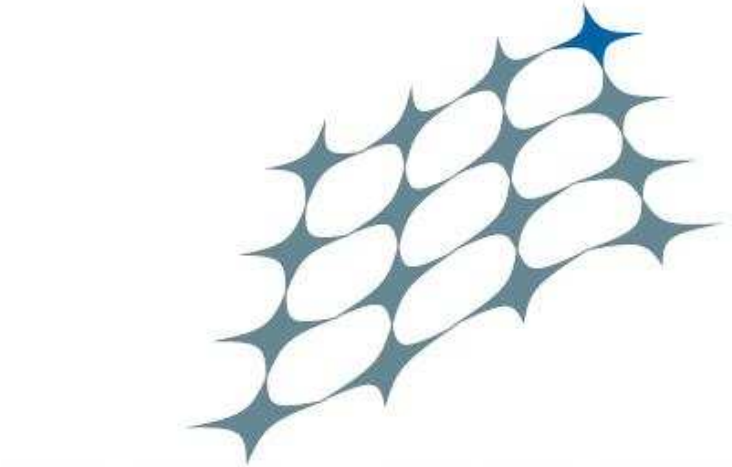
- Enables services to be offered with a contract specifying QoS (not just best effort) – customize the service to customer profile
- Generic – applies to any type of services (Grid or Cloud or SaaS)
- Easy progressive deployment: just a new management functionality
- Prioritise service provision for higher value contracts
- Increased client's confidence through transparency
- Prices adapted to demand to augment asset utilisation

Innovation Impact

- Implementing latest SLA specification (**WS-Agreement, 2007**) : interoperability
- First integrated framework for WS-Agreement on GT4
- Application of the dynamic SLA concept
  - on service instances (possibly more clients through finer control of low utilisation)
  - not just on global service provision (current scheme – which guarantees fixed revenues)

Conclusion


biz
Technical Requirements
Common problems
Design patterns
Implementation patterns
Components
biz
Business Experiments
biz
Validation/ extension



European Commission

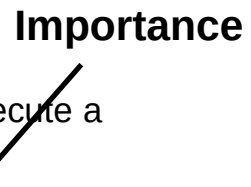
Information Society Technologies

Thank you!!

igor.rosenberg@ atosorigin.com

Classification of the initial topics by the first BEs

- **T1: SLA Template** Guidance to the write-up of the SLA template offered by providers: how-tos, skeleton, GUI, traps to avoid. **Medium**
- **T2: Publication and Discovery** Mechanism to allow an efficient management of distributed resources is proposed (good-use rules, to facilitate the work of a matcher). Low
- **T3: Negotiation** Tools easing the negotiation (bargain) of an SLA **Medium**
- **T4: Optimisation of Resource Selection** Selection of the most suitable host (to deploy and execute a service), optimise a predefined measure of system efficiency. **Medium**
- **T5: Monitoring** SLA Monitor system that checks the status of the SLA is proposed **High**
- **T6: Re-negotiation** Changing an already accepted SLA. Novelty is the existence of a previous contract providing initial values, and possibly running jobs (migration). Low
- **T7: Evaluation** Comparing predicting all the terms of the agreed SLA with the current situation (gained through monitoring), to discover potential violations to the agreement. **High**
- **T8: Accounting** Calculate the price for a given service (related to the SLA-metrics) Low



	T1	T2	T3	T4	T5	T6	T7	T8
BE03	X	X	X					
BE06	X	X	X	X	X	X	X	
BE07	X			X				
BE08	X		X	X	X		X	
BE09	X	X	X	X	X		X	X
BE10	X							
BE16	X			X				

Most relevant BEs (presenting major interest on SLAs)

SLA Cluster Exploitation Opportunities

- **Possible follow-ups in other European research projects**
  - **SLA management is still a hot research topic.**
- **BE25: SLA negotiation & evaluation in e-Health.**
  - **Potential for more consultancy and support work based on integrated framework.**
  - **eHealth is a sector requiring variable service provision, where QoS is critical**
- **Cloud hype slowly starting to introduce SLAs, the model adapts perfectly.**

SLA cluster components

SLA architecture for GT4

Common Capability	Component	Middleware	Release date	Version number	Operating System	License
Negotiation	SLA Negotiator	GT4 Java WS Core	03/11/2008	V0.4	Any	Apache V2
	AssessGrid Negotiation Manager	GT4 Java WS Core	16/04/2008	0.8.239	Any	Apache V2
Runtime monitoring	SLA-Evaluator for GT4	GT4 Java WS Core	16/12/2008	0.96	Any	Apache V2
	SLA-Evaluator for .NET	GRASP/.NET	29/07/2008	1.0	Windows XP, Windows 2003 Server	Atos Origin Dual License
	SLA Violation Notifier	GRIA 5.0	20/12/2008	2.0	Any	NTUA License
SLA Accounting	SLA Accounting for GT4	GT4 Java WS Core	07/05/2009	V0.7	Any	Apache V2
Optimisation of resource selection	Resource Selection Optimization	GT4	02/06/2008	1.0	Linux	LGPL