Dynamic Service Generation: 
Agent interactions for service exchange on the Grid

Clement Jonquet
PhD defence
Thursday November 16, 2006
Speech overview

1. Introduction to Dynamic Service Generation (DSG)
2. GRID and Service Oriented Computing (SOC) key concepts
3. Multi-Agent Systems (MAS) and the STROBE model
4. Service based integration of GRID and MAS (AGIL)
5. Conclusion
Speech overview

1. Introduction to Dynamic Service Generation (DSG)

2. GRID and Service Oriented Computing (SOC) key concepts

3. Multi-Agent Systems (MAS) and the STROBE model

4. Service based integration of GRID and MAS (AGIL)

5. Conclusion
Example: ‘looking for a job’ scenario

1. Introduction to DSG

- Complex wish to explain
- Long & dynamic interactive conversation
- Stateful & asynchronous
- Collaborative (other services)
- Generation of service

- Precise request with parameters
- Remote procedure call
- Short, one-shot interaction
- Stateless & synchronous
- Delivery of product

- Job seeker
- Precise project manager position

- Job agency
- <search_offer> service

- Road and map guide
- <distance> service

Clement Jonquet - PhD defence
Context

- **WHAT**
  - Modelling dynamic service exchange interaction in computer mediated contexts for both human and artificial entities

- **WHY**
  - Enhancing the way these distributed entities work in collaboration to solve the problem of one of them

- **HOW**
  - Proposing models and tools inspired from 3 different domains of Informatics: SOC, GRID and MAS

⇒ What kind of services do we want for the Informatics of tomorrow?
Thesis statement and objective

- A service exchange is not a simple delivery of product
  - It is based on conversation

- Tools that enable to provide and use services by means of conversations
  - Importance of the concept of state

- Going towards a new vision of the concept of service
  - Dynamic service generation
Dynamic Service Generation (DSG)

- A solution, identified and chosen among many possible ones, offered to the problem of someone

- Services
  - Imply creation of something ‘new’
  - Are associated with processes
  - Are constructed by means of conversations
  - Have a learning dimension (knowledge creation)
  - Create relationships between members of communities

> Computerization of the concept of service is not easy
DSG vs. Product delivery

- **Product delivery approach**
  - One-shot interaction process between a pair
    - User
    - Provider
  - ex: buying ready-to-wear clothes
  - ex: asking to MAPPY a distance

- **DSG approach**
  - Result of the activation and management of a process defined by the triplet
    - User
    - Conversational process
    - Provider
  - ex: having clothes made by a tailor
  - ex: finding a job thanks to JOBWINER
Method adopted

- Characterization process
  - List of DSG characteristics

- Try to address some of these characteristics
  - Concrete tools and models
  - Experimentations on simple scenarios
  - Re-usability of concrete principles

- Motivation
  - To formalize the convergence of 3 important domains for DSG: SOC, GRID and MAS

- Integration approach
Why SOC, GRID and MAS?

---

**SOC**
- Web oriented
- Use registries
- Standardization & interoperation

**MAS**
- Business process management
- Semantics
- State management
- Social structures
- Negotiation
- Conversation modelling
- Learning & reasoning

**GRID**
- Trust & security
- State management

---

1. Introduction to DSG
Speech overview

1. Introduction to Dynamic Service Generation (DSG)

2. GRID and Service Oriented Computing (SOC) key concepts

3. Multi-Agent Systems (MAS) and the STROBE model

4. Service based integration of GRID and MAS (AGIL)

5. Conclusion
What is GRID?

- **Foundation**
  - Flexible, secure, coordinated resource sharing among Virtual Organizations (VO) [Foster et al., 1999, Blueprint] & [Foster et al., 2001, Anatomy]

- **Originally**
  - Environment with a large number of networked computer systems where computing and storage resources could be shared as needed and on demand

- **Extended**
  - Virtualization of resources and assignment to stateful and dynamic services [Globus alliance, 2002, Physiology (OGSA)]

- **Last standard**
  - GRID-SOC convergence
  - Grid service = stateless service + stateful resource
Grid service

- Compliant with Web service and SOA standards [W3C]
  - Describable, discoverable component
  - Message based communication
  - Perform some function

- 2 major new aspects
  - State management (stateful/stateless)
  - Lifetime management (transient/persistent)

- Dynamic assignment of resources to a service
  - Instantiation mechanism
2. GRID & SOC key concepts

3. Invocation (SOAP)

5. Identification (GSR/GSH)

Grid service life cycle

1. Publication (WSDL)

2. Discovery (WSDL)

3. Execution (SOAP)

4. Instantiation

6. Execution (SOAP)

GRID SERVICE FACTORY

GRID SERVICE INSTANCE

USER
GRID key concepts

2. GRID & SOC key concepts
Speech overview

1. Introduction to Dynamic Service Generation (DSG)
2. GRID and Service Oriented Computing (SOC) key concepts
3. Multi-Agent Systems (MAS) and the STROBE model
4. Service based integration of GRID and MAS (AGIL)
5. Conclusion
What are agents and MAS?

- **Definition** [Ferber, 1995] & [Jennings, 2001]:
  - Physical or virtual autonomous entities:
    - Situated in a particular environment
    - Capable of perceiving and acting in that environment
    - Designed to fulfil a specific role
    - Communicate directly with other agents
    - Possess their own state (and controls it) and skills
    - Offer services
    - Have a behaviour that tends to satisfy their objectives

- **Service oriented characteristics**
  - Reactive, proactive, and adaptive
  - Know about themselves, and have a memory and a persistent state
  - Interact and work in collaboration
  - Able to learn and reason in order to evolve
  - Deal with semantics associated to concepts by processing ontologies
Why a new architecture?

- Agent communication requirements
  - To allow dynamic language evolution
  - Strong interlocutor model

- No dedicated conversation context
  - To develop a dedicated language
  - To adapt interlocutor’s specific aspects

- Composed of set of modules
  - Separate the interaction module and the service execution module
STROBE proposition [Cerri, 1996 & 1999]

- **Object**
  - To represent agents
  - Encapsulation of state
  - Message passing

- **Stream**
  - Flow of messages exchanged
  - Lazy evaluation

- **Environment**
  - To interpret messages
  - Multiples

- 3 first-class primitives

- Agents as interpreters
  - Read-Eval-Print-Listen loop

"Shifting the focus from control to communication" [Hewitt, 1977]
The STROBE model [Jonquet & Cerri, AAI journal, 2005]

- Agent representation and communication model

- Include an interpreter in each environment
  - Dedicated to interlocutors

- STROBE agents build their own dedicated languages while communicating
  - Language = environment + interpreter

- Language evolution done dynamically at:
  - The data and control level
  - The interpreter level (using reflection and meta-programming techniques)

- Formalized, implemented and experimented
  - Scheme & Java/Kawa in MadKit
STROBE agent representation

- **Brain**
  - Set of modules
  - e.g., learning & reasoning

- **Cognitive Environment**
  - Set of bindings (data level)
  - e.g., [a 3]

- **Capabilities**
  - Functions/procedures (control level)
  - e.g., [square (lambda (x) (* x x))]

- **Cognitive Interpreter**
  - Specific capability (interpreter level)
  - [INT (lambda (exp) (eval exp env))]
Cognitive Environment

- Conversation context
  - Keeps the state of a conversation
  - Context of evaluation of messages
  - Interlocutor model
  - Evolves dynamically at the data, control and interpreter levels

- Dedicated to an interlocutor or a group of interlocutors
  - Agents develop a communication language for each interlocutor (environment + interpreter)
  - Agents have dedicated capabilities

- A STROBE agent has only one CE dedicated to a given interlocutor

- When an agent meets a new interlocutor, it:
  - Instantiates a new CE by copying an existing one
  - Shares an already existing CE
Message interpretation

- Done:
  - in a given environment
  - with a given interpreter

- Both dedicated to the interlocutor (or group of interlocutors)

- Both able to change.
Speech overview

1. Introduction to Dynamic Service Generation (DSG)

2. GRID and Service Oriented Computing (SOC) key concepts

3. Multi-Agent Systems (MAS) and the STROBE model

4. **Service based integration of GRID and MAS (AGIL)**

5. Conclusion
Motivation

- Early suggested for the Computational Grid 
  [Rana & Moreau, 2000]

- Agents as a key element of the Semantic Grid 
  [DeRoure, Jennings et al., 2001]

- MAS and GRID need each others: brain meets brawn 
  [Foster, Jennings & Kesselman, 2004]

- Significant complementarities
  - GRID is secure but interaction poor
  - GRID manage raw data without semantics
  - MAS need interoperation and standardisation

- Service-oriented MAS 
  [Huhns et al. 2005]
GRID-MAS analogies

- Direct message passing based communication
- Service interoperation
- Orchestration and choreography of services
  - Business process management
- Service state and lifetime

- Idem
- Agent interaction
- Interaction protocol and agent conversation
  - Collaboration scenario
- Agent intelligence and autonomy
GRID-MAS analogies

4. Service based integration

- **Grid user**
  - Member of VOs
  - Uses services
  - *Offers services*
    [Cerri et al., OGSHA, 2004]

- **VO**
  - Context of service exchanges
  - Exchanges inside
  - Services publication

- **Service**
  - Functional position
  - CAS
  - Services are local to VO

- **Agent**
  - Member of groups
  - Holds roles
  - Delegates tasks

- **Group**
  - Context of activities
  - Communications inside
  - Capabilities become roles

- **Role**
  - Functional position
  - Role management
  - Roles are local to groups

[Foster et al. OGSA, 2002] [Ferber et al. 2003]
State of the art of current ‘integration’ activities

- **Agents and Web services (WS)**
  - Distinct/uniform view of agents and WS
    - e.g., transform SOAP call into FIPA ACL message [Greenwood et al, 2004]
  - MAS based Service Oriented Architecture
    - e.g., agents for WS selection [Singh, 2003]
  - MAS based Business Process Management
    - e.g., workflow approaches [Bulher & Vidal, 2003]

- **MAS to improve core GRID functionalities**
  - Resource management [ARMS, 2001][AgentScape, 2002]
  - VO management [Conoise G2005]

➤ *Interesting approaches, but not really interested in integrating the 3 domains*
Mapping of GRID and MAS concepts

- **Agent**
  - Unifies AA, HA, Grid user
  - Active entities involved in service exchange
  - Autonomous, intelligent and interactive
  - Grid users as potential artificial entity

- **VO (= Group = Community)**
  - Dynamic social group (virtual or not)
  - Context of service exchanges

- **Service-Capability relationship**
  - Virtualization of an agent capability
  - A service is an *interface* of a capability available for a VO

- **Instantiation**
  - Process of creating a new service-capability couple
  - Instantiating a new service means to instantiate a new CE containing the new capability
Agent-Grid Integration Language

[Jonquet, Degenie & Cerri, MAGS journal, 2007]

- 3 elements:
  - Set concepts
  - Set of relations between concepts
  - Set of integration rules

- Graphical description language
  - Kind of UML for GRID-MAS integrated systems

- Set-theory formalization
  - Example: holding relation

\[
\text{holding}: X \rightarrow A \cup H \text{ (application)}
\]

**Rule 23** All agents members of a VO hold a X509 certificate.

\[
\forall a \in A, \forall o \in O, a \in o \Rightarrow \exists x \in X, \text{holding}(x) = a
\]
AGIL’s integration model

4. Service based integration

- Virtual Organization
- Service container service
- Community Authorization Service
- Service instance
- Authorizing relation
- Handling relation
- X509 certificate
- Holding relation
- Membership relation
- Agent
- Brain
- Cognitive Environment
- Capability
- Interfacing relation
- Instantiating relation
- Interacting relation
AGIL discussion (1/2)

- Integrates both GRID and MAS properties
  - Bottom-up vision of service in GRID
  - Top-down vision of service in MAS

- Not restrictive neither for MAS nor GRID
  - Today, but tomorrow?

- Includes some of the MAS based GRID approaches
  - Meta GRID core mechanism are themselves Grid services
AGIL discussion (2/2)

- Both a description language and a integration model
  - Allows to represent both the meta-model and its instances (i.e., future integrated systems)
  - Rigorously fix the concepts, relations and rules

- STROBE is adequate for AGIL
  - WSRF: stateful resource + stateless service
    - evolution only at the resource level
  - AGIL: CE + capability
    - evolution of the CE and capability levels

- A service is an interface of a capability executed with Grid resources but managed by an intelligent, autonomous and interactive agent
Speech overview

1. Introduction to Dynamic Service Generation (DSG)

2. GRID and Service Oriented Computing (SOC) key concepts

3. Multi-Agent Systems (MAS) and the STROBE model

4. Service based integration of GRID and MAS (AGIL)

5. Conclusion
Conclusion (1/2)

○ We tried to address the question of service exchange modelling in computing context

○ Dynamic Service Generation
  ● A reflection about the concept of service that defends an integration of SOC, MAS and GRID
  ● Conversation based view of services

○ 3 concretes contributions
  ● STROBE
  ● i-dialogue (not presented today)
  ● AGIL
Conclusion (2/2)

- We adopted an integration approach

- AGIL is a formalization of agent interactions for service exchange on the Grid

  An answer to the problem of service exchange modelling
  - Contributes to go towards future DSG systems
The ‘looking for a job’ scenario in AGIL
Thank you!
Perspectives

- **Short term ones**
  - Learning rules on CEs in the STROBE model
  - Integrate first-class continuations in CE
  - Add to AGIL other concepts, relations and rules
  - Implement AGIL as an ontology [Duvert & Jonquet et al., AweSOMe workshop, 2006]

- **Long term ones**
  - Integrate new aspects and characteristics of DSG (specially coming from SOC [Singh & Huhns, 2005])
  - Continue the DSG characterization process
  - Validate the AGIL integration model on a large scale project
  - Integration with Semantic Web Services approaches (service container as a semantic platform) [Domingue & Motta, IRS and WSMO, 2005]
  - Provenance of dynamically generated services [Moreau et al., 2005]
Publications

www.lirmm.fr/~jonquet/Publications

- **Journal**

- **International conference**

- **National conference**
I-dialogue

[Jonquet & Cerri, International Lisp Conference, 2005]

- An computational abstraction to model agent multi-party conversations
  - Inspired by the *dialogue* abstraction proposed by [O’Donnel, 1985] to model process interactions
  - Uses first-class procedures, streams and lazy evaluation

- Enables to manage the entire conversation dynamically (not pre-determined)

- Adequate for intertwined dialogues
  - Executed simultaneously
  - Inputs and outputs depend on each other
  - Service composition
The dialogue abstraction

- Interactive session between 2 agents, which take turns sending messages to each other:

- Each agent computes a new state and a new output from its previous state and the last input it received from the other agent, using its transition function:

\[
\begin{align*}
  f^A_B : [\alpha_{j+k}, I^A_B] & \rightarrow [\alpha_{j+k+1}, O^A_B] \\
  f^B_A : [\beta_k, I^B_A] & \rightarrow [\beta_{k+1}, O^B_A]
\end{align*}
\]
The *i-dialogue* abstraction

- Agent B should consume 2 input streams and produce 2 output streams.

- Transition functions of B, do not produce respectively an output stream for A and B but the opposite.
Evaluation & experimentations

○ STROBE
  ● 2 implementations (Scheme & Java/Kawa in MadKit)
  ● 2 main experimentations
    ○ *Meta-level learning by communicating* (teacher – student dialogue for the learning of a new performative)
    ○ *Dynamic specification of a problem* (client – service provider dialogue to construct an train ticket reservation. Use of non-deterministic interpreters (constraints specification))

○ I-dialogue
  ● Implemented in Scheme
  ● Integration with the STROBE implementation in progress

○ AGIL
  ● Implementation under the form of an ontology started
STROBE agent in MadKit

- MadKit: Multi-Agent platform developed at LIRMM [Ferber, Gutknecht & Michel, 2000]
  - www.madkit.org

- Based on the Agent/Group/Role model

- Java agents but also Scheme, Python etc.

- Scheme – Java link with Kawa
STROBE communication language

- **Message structure:**

  $$MSG = \{AGENT_S, AGENT_R, PERFORM, CONTENT\}$$

  with $$PERFORM = \{assertion, ack, request, answer, order, executed, broadcast\}$$

- **Example of exchanges:**

<table>
<thead>
<tr>
<th>Teacher ($A_T$)</th>
<th>Student ($A_S$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>${A_T, A_S, request, square }$</td>
<td>${A_S, A_T, answer, undefined }$</td>
</tr>
<tr>
<td>${A_T, A_S, assertion, (define (square x) (* x x)) }$</td>
<td>${A_S, A_T, ack, (* . *) }$</td>
</tr>
<tr>
<td>${A_T, A_S, order, (square 3) }$</td>
<td>${A_S, A_T, executed, 9 }$</td>
</tr>
</tbody>
</table>
Creation of a new CE

- 2 types of CE
  - A global one (private)
  - Several local ones (dedicated)

- An agent has only one CE dedicated to a given interlocutor

- When an agent meets a new one, local CE are instantiated by:
  1. Copying the global CE
  2. Copying a local CE
  3. Sharing a local CE
Learning by communicating

- Every languages propose 3 levels of abstraction

  Data
  (set! a 3)

  Control
  (define (square x) (* x x))

  Interpreter
  add a special form

- STROBE enables ‘learning-by-being told’ at the 3 levels
  - Reflective interpreters and reifying procedures [Jefferson et al., 1992]
  - First class interpreters [Simmons et al., 1992]
  - 2 levels of evaluation using the `eval` function in the language
A ‘counter’ example in AGIL

(S1) \(<\text{incr\_count\_factory}> : cc:=cc+1;\)
(S2) \(<\text{decr\_count\_factory}> : cc:=cc-1;\)
(S3) \(<\text{incr\_count\_inst1}> : cc:=cc+1;\)
(S4) \(<\text{incr\_count\_inst2}> : cc:=cc+1; \text{print}(cc);\)
(S5) \(<\text{decr\_count\_inst1}> : cc:=cc-1; \text{print}(cc);\)

\(E^D_D\) D’s global cognitive environment
\(E^D_C\) D’s local cognitive environment dedicated to C
\(E^D_{A,B}\) D’s local cognitive environment dedicated to A and B
CC CC local variable binding
Comparison with WSRF

(S1) <incr_count_factory>

(S2) <decr_count_factory>

(S3) <incr_count_inst1>

cc := cc + 1;
?? print(cc);

(S4) <incr_count_inst2>

cc := cc + 1;

(S5) <decr_count_inst1>

cc := cc - 1;
print(cc);
PD vs. DSG (1/2)

- User exactly knows:
  - what he wants (clearly defined problem)
  - what the system can offer him (clearly defined product)
  - how to express his request (adaptation to provider’s language)

- Same type of deliveries
- No history

- Cannot realise DSG
- Pre-developed by the provider (clearly defined goal)

- User:
  - has unclear wish (bootstrapping situation)
  - elicits and understands progressively the provider’s capabilities
  - the provider adapts to the user’s language

- Unique generated services (conversation is unique)
- Depend from previous DSG and history
- Can realise PD
- Offered realised within a service domain and constructed dynamically (user’s specific objectives)
PD vs. DSG (2/2)

- Long lifetime
- Slow evolution
- No reasoning
- No knowledge creation
- Same satisfaction for each delivery
- No possible retraction
- No emotion or psychological impacts
- Easily valuable an billable
- Able to announce the result

- Passive

- Ephemeral life-cycle
- Dynamic and natural evolution
- Static and dynamic reasoning
- Pedagogical perspective
- Satisfaction increases with each generation
- Anytime mind changing
- Implies (+ or -) emotions

- Hardly valuable and billable
- Gain the user’s trust (not announce or guarantee a final result)
- Perpetually evolving, learning on their previous generation to improve the next ones

- Pro-active
Service taxonomy

- **SERVICE**
  - (1) **SYNCHRONISM**
    - synchrone
    - asynchrone
  - (2) **STATE MANAGEMENT**
    - stateful
    - stateless
  - (3) **ARCHITECTURE**
    - system oriented
    - service oriented
  - (4) **STANDARDIZATION / PUBLISHABILITY**
    - standardized
    - not standardized
    - dynamic/transient
    - static/persistent
  - (5) **DYNAMICITY / LIFETIME**
    - synchrone
    - asynchrone
  - (6) **TOPOLOGY**
    - point-to-point
    - multipoint
  - (7) **EXECUTION**
    - one shot interaction
    - conversation
  - (8) **SEMANTIC**
    - semantic description
    - no semantic description
  - (9) **USAGE**
    - human or artificial only
    - human and artificial
  - (10) **COMPOSITION**
    - simple
    - aggregated/composed
Economic taxonomy extension

- **Good**: physical, tangible object (natural or man-made) used to satisfy people’s identified wants and that upon consumption, increases utility.

- **Service**: non-material equivalent of a good. (e.g., information, entertainment, healthcare and education).

- **Product**: Output of any production process (tangible good or intangible service).
Elements of SOC

matchmaking between goals and functionalities
→ semantics

virtualization dissemination
→ aggregation
→ generation

orchestration/workflow conversation/choreography
→ dynamic and based on agent conversation

agents identification, functional description, interface specification, operational description, contract life cycle, exchange description
→ based on agent negotiation

message passing based synchronous point-to-point
→ asynchronous multi-party

stateless and internal stateful
→ separation between stateless services and stateful resources
Elements of Service Oriented Architecture

- Historically:
  - software component based approaches (DCE, CORBA, COM, RMI)
  - to standardize invocation mechanisms

- Framework:
  - Web services [W3C]
    - describable, discoverable
    - message based
    - perform some function
  - interoperability and standardization
  - identifies 3 components

- Evolution:
  - simple service invocations, to business processes (orchestration, choreography, composition)

- Technologies:
  - WSDL, SOAP, UDDI, WSCL, WSFL, BPEL4WS, PSL...
Web services limits

- RPC like computing
- Object-oriented behaviour
- No user adaptation
- No memory (stateless)

- No conversation
- Synchronous communication
- No lifetime management
- Passive
- No semantics

➤ Web services are typical PDS

A service is seen as a standardized and interoperable interface of a specific function (accessed remotely)