Agent based languages and architectures for web service integration

Stanislaw Ambroszkiewicz

(www.ipipan.waw.pl/mas)

Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland

and

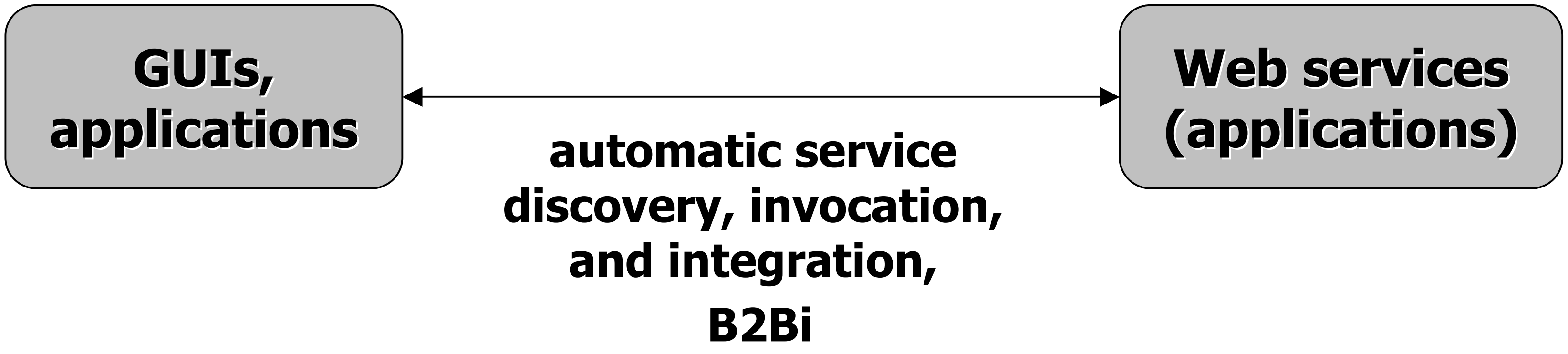
Institute of Informatics, University of Podlasie, Siedlce, Poland

Radical Agent Concepts 16-18.I.2002, NASA, Washington

The challenge for agent technology:

Web service integration

- **Internet (TCP/IP)**
  --> simple and ubiquitous computer networks
- **WWW (HTTP)**
  --> simple and ubiquitous access to data
- **Web services (SOAP + WSDL + UDDI + ???)**
  --> simple and ubiquitous access to applications


GUIs, applications
Web services (applications)
automatic service discovery, invocation, and integration,
B2Bi

Web services?

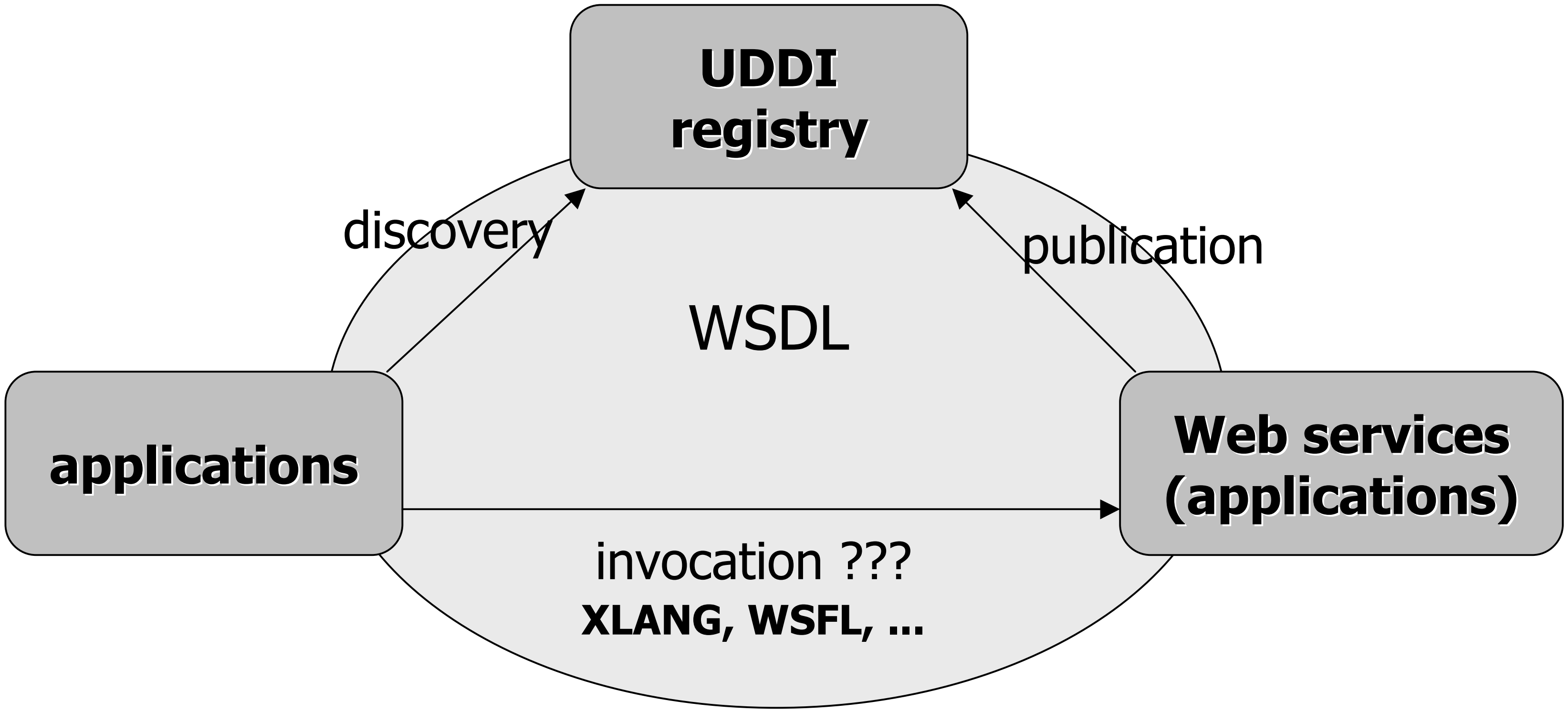
⌘ ***Web services*** are self-contained, self - describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions that can be anything from simple requests to complicated business processes ...

Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service (in an automatic way!).

⌘ From a service provider's point of view, if they can setup a web site they can join global community. From a client's point of view, if you can click, you can access services.

Industrial standards:

Web service integration - IBM, Microsoft, HP, SUN, ...



⌘ **Once service is discovered, a dedicated interface must be implemented to interact.**

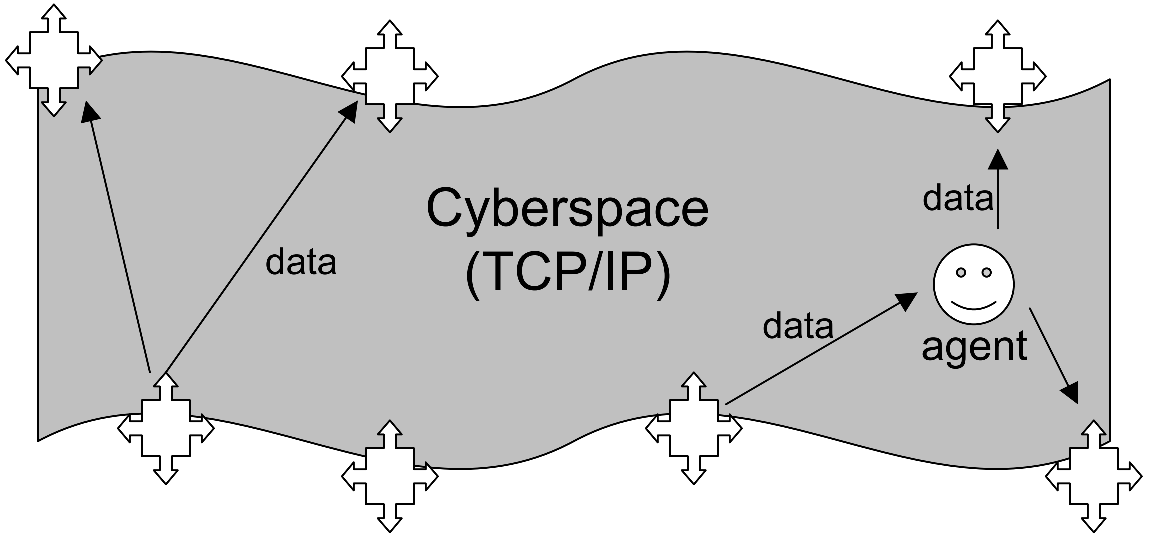
Related efforts

- **Microsoft .Net, Sun ONE, E-speak (HP), …** strategies.
- WSDL + UDDI - success or failure?
- XLANG, WSFL, BTP, ebXML,, …
  - partial (complex?) solutions
  - one simple protocol is needed!
- Web Services Activity of W3C (extended XMLP)
- **DAML-S (DARPA)** project aims at a complete solution based on Semantic Web concept (initial stage).

Agent based service integration:

How to realize it?

- **Web Services** - the places where data are processed and stored.
  - applications, GUIs, devices, e-commerce, e-business, …
- **First of all:** A generic language for describing data processing controlled by agents in networked environment (**cyberspace**) is needed!
- Let's design such language!



Language

What do we want to describe?

- **resources** - data collected in types, e.g., ***Typ1, Typ2,*** …
- **services** - applications where the resources are stored and processed:
  - type of operation performed by the service:
    - precondition ***form_in***
    - postcondition ***form_out***
- **functions** implemented by operations, e.g., ***f;*** parameter ***a*** is of type ***Typ1***, the value ***f(a)*** is of type ***Typ2***

Language

What do we want to describe?

- **tasks** specifying <u>what</u>, is to be processed, <u>how</u>, and <u>when</u>, and <u>where</u> the result is to be stored:
  - <u>when</u> - timeout: ***(leq, gmt(), date),*** i.e., the current GMT time is less or equal to ***date***
  - <u>where</u> - relation: ***(is_in , res, ser)*** , i.e, a resource ***res*** is in service ***ser***
- **task example:**
  - "resource ***res1*** is processed by function ***f*** and the result is stored in service ***ser1*** by the time ***date1";*** formally:
    - ***(is_in ,f(res1), ser1) and (leq, gmt(), date1)***

Language

What do we want to describe?

- **Service attributes:**
  - ***operation_type(service)*** is a pair of atomic formulas: ***form_in*** and ***form_out***
  - ***commitments(service)*** is a pair of atomic formulas : ***form_in*** and ***form_out***
- **Agent** a processes dedicated to a single task realization
- **Agent attributes:**
  - ***intentions(agent)*** is an atomic formula
  - ***knows(agent)*** is an atomic formula
  - ***goals(agent)*** is an atomic formula
  - ***commitments(agent)*** is a pair of atomic formulas: ***form_in*** and ***form_out***

Language:

Term and formula construction

- Terms are constructed in the standard way
- Composite formulas are constructed using only conjunction, disjunction and implication;
  no quantifiers and no negation!

Our idea of service integration

Service

- **service description**:
  - unique name and communication address - URI, e.g., service ***name* = *pegaz://ii.ap.siedlce.pl/uslugi/moj-service***
  - operation type: the pair of formulas
    - ***form_in(operation_type(name))***
    - ***form_out(operation_type(name))***
  - the service is invoked if ***form_in*** is satisfied
  - ***form_out*** describes the result of operation performed by the service

Our idea of service integration

Service invocation

- **Six steps of service invocation:**
  - ***agent*** sends to the ***service***: "my intention is ***φ***"
    - ***φ --> intentions(agent)***
  - service responds: "I commit to realize ***φ*** if ***ψ*** is satisfied"
    - ***ψ --> form_in(commitments(service))***
    - ***and***
    - ***form_out (commitments(service)) --> φ***
  - ***ψ*** is satisfied
  - operation is performed by the service
  - ***φ*** is satisfied
  - confirmation is sent to the agent

Our idea of service integration

Service composition into workflow

- A TASK is created by a user and delegated to an agent.
- The TASK becomes the ***GOAL*** *of the* ***agent.***
- ***Agent's GOAL*** becomes its first intention ***φ0*** (with a timeout!)
- Service ***SER-0*** agrees to realize ***φ0*** if ***φ1*** is satisfied
- ***φ1*** becomes the next agent's intention
- Service ***SER-1*** agrees to realize ***φ1*** if ***φ2*** is satisfied
- ***φ2*** becomes the next agent's intention
- Service ***SER-2*** agrees to realize ***φ2*** if ***φ3*** is satisfied
- (continued on the next slide)

Our idea of service integration

Service composition into workflow

- … and so on
- Finally, ***φN*** becomes the next agent’s intention.
- Agent is able to satisfy the formula ***φN***
- **Workflow for realizing agent’s goal is constructed!**
- Any formula includes a timeout
- The timeouts synchronize the workflow execution
- **Workflow execution:** domino effect
  - ***φN --> … --> φ3 --> φ2 --> φ1 --> φ0 = GOAL***

Language Entish

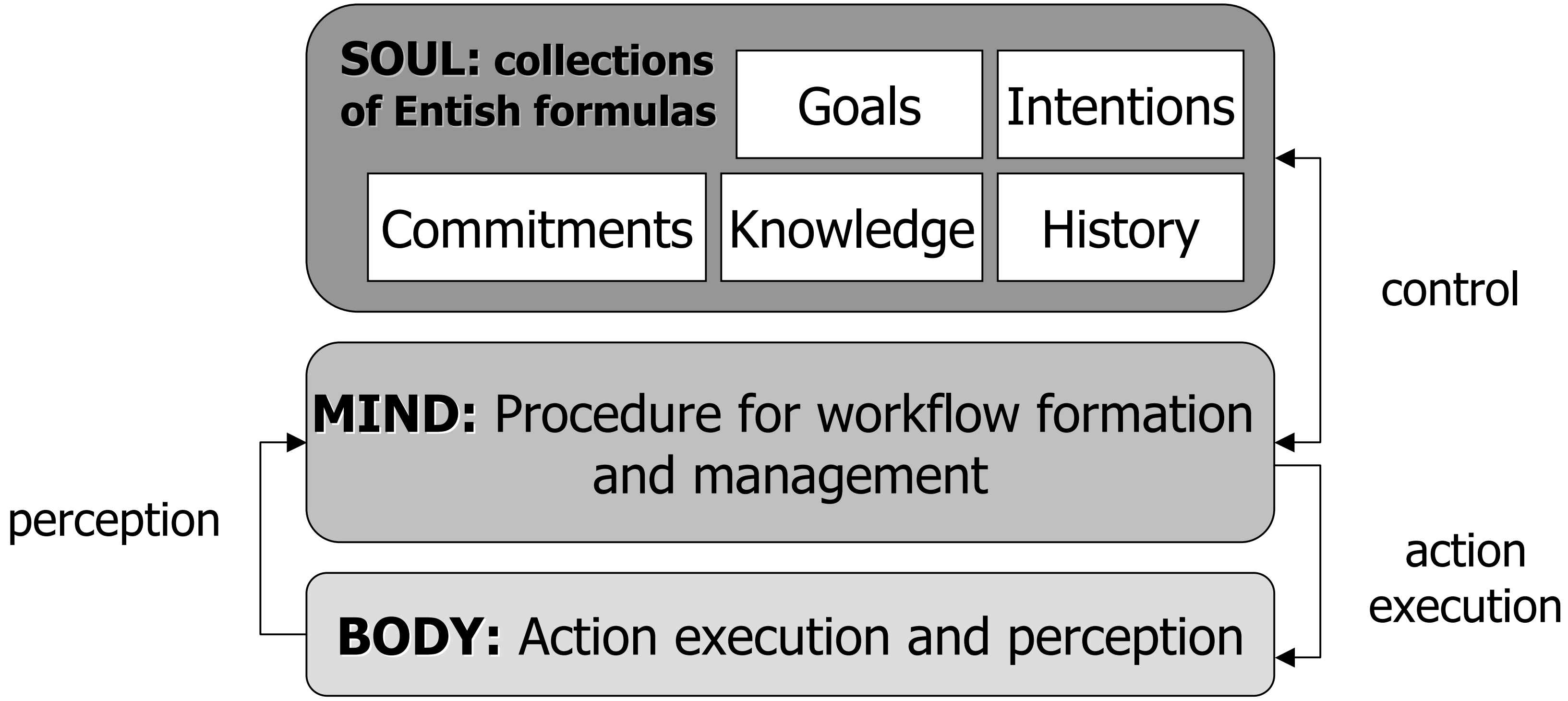
Don't ask what it means, but rather how it is used.
- L. Wittgenstein

- **Entish is design as a minimum necessary to construct protocols for service integration by agents.**
- A simple version of the language of first order logic with types.
- Describes <u>only static</u> relations between agents, services, and resources; no actions - fully declarative language.
- Ability to express agent / service mental attributes: intentions, goals, commitments, knowledge.
- The idea of webizing language (TBL) is applied - elements have unique names URI. Entish can be used and developed in a distributed way: users can introduce new definitions, and new primitive notions to the language.
- **Do we need formal meaning provided by ontologies ?**
  The answer: NO!

Agent architecture:

the idea of soul migration

- **The consequences of our language:**
  - new (?) agent architecture,
  - soul as a universal data format for storing essential data of agent process



SOUL: collections of Entish formulas
Goals
Intentions
Commitments
Knowledge
History
control
MIND: Procedure for workflow formation and management
perception
action execution
BODY: Action execution and perception

Soul migration

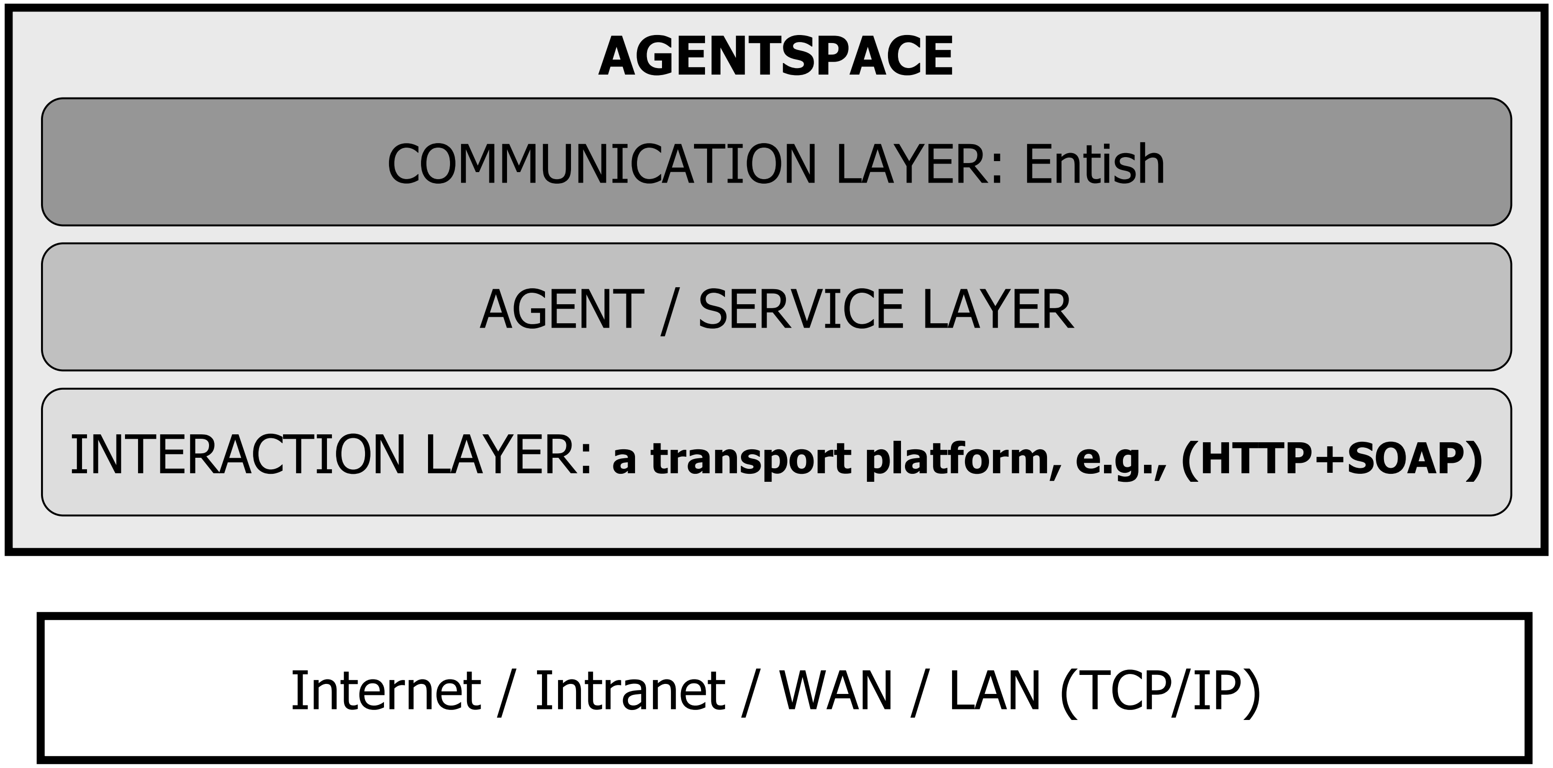
- Soul - minimal data necessary to assure continuation of agent process (closing and then restoring) in a heterogeneous environment
- Soul data (mental (**BDI?**) attributes) are expressed in Entish
- Soul format (in XML) is independent from mind and body
- Soul is design to be universal agent data that can inter-operate with any mind and body implemented according to the format
- **The idea of soul and the problem of agent persistence:**
  - soul is designed to be a complete data necessary to recover agent process
- **Soul migration and the problem of security of hosts open for strange agents:**
  - soul is only data, not a binary code to be executed

From language to implementation

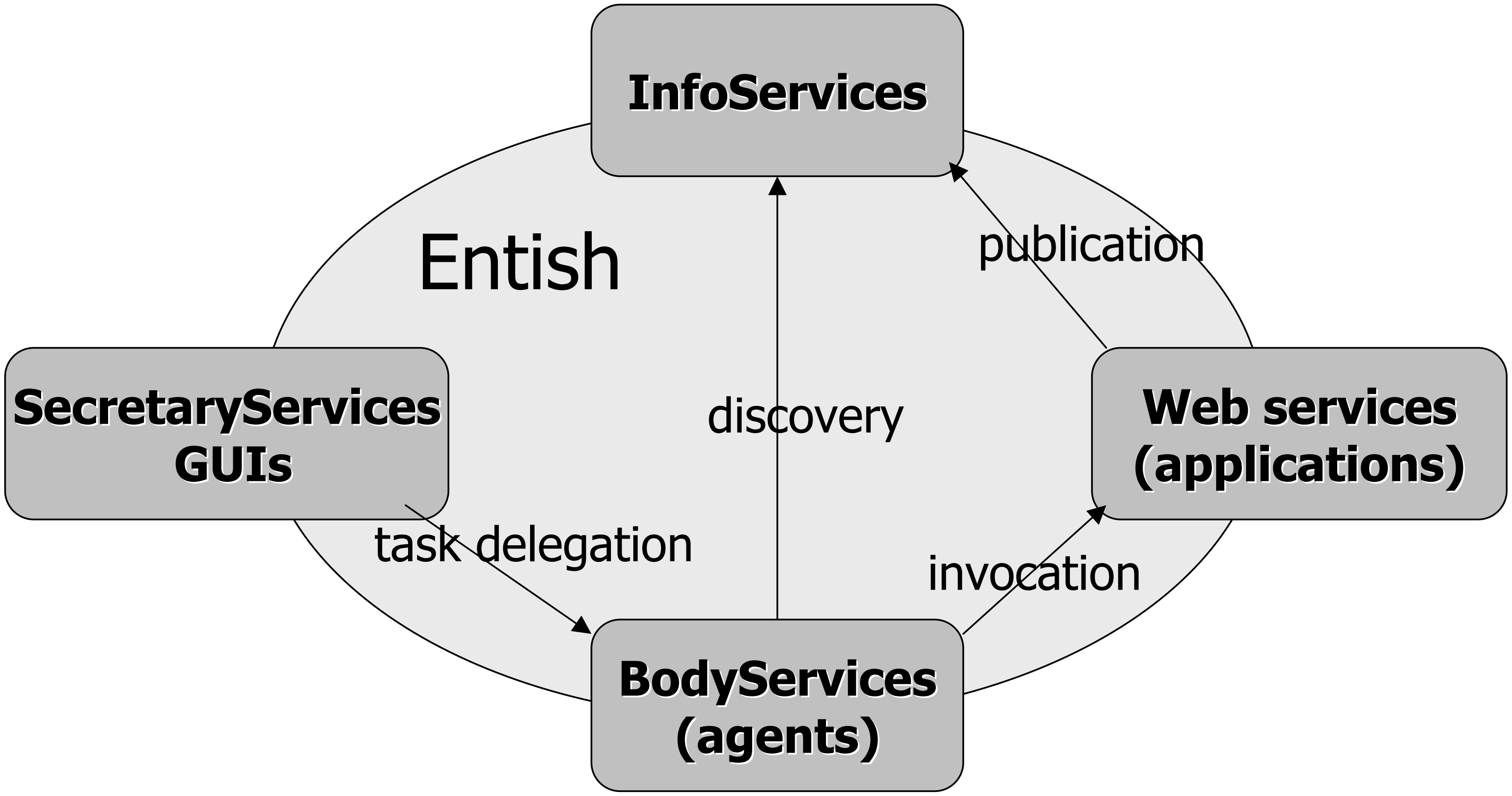
- ***Language*** *--> formal model (semantics) --> abstract architecture --> implementation*

- ***Entish*** *--> prime event structure (spec. of agent / service behavior) --> agentspace architecture -->* ***agentspace*** *= infrastructure for web service integration by agents*

Agentspace architecture:
a generic layered view


AGENTSPACE
COMMUNICATION LAYER: Entish
AGENT / SERVICE LAYER
INTERACTION LAYER: a transport platform, e.g., (HTTP+SOAP)
Internet / Intranet / WAN / LAN (TCP/IP)

A specific agentspace architecture



- Entish is a communication language for automatic service integration

Services
- functioning of agentspace

- **SecretaryService** - User GUI to agentspace.
  - Helps user to formulate his/her task in Entish.
  - Creates agent soul and sends it to BodyService.
  - Presents the result of task performance to the user.
- **BodyService**
  - Implements mind and body layers of our agent architecture.
  - Once an agent soul is delivered to BodyService,
    the agent process is created.

Services

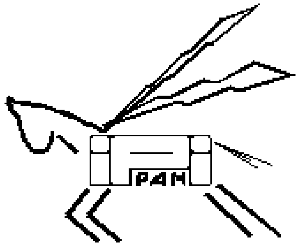
- functioning of agentspace

- **InfoService** - distributed and open knowledge base
  - web services publish their operation types in InfoServices.
  - agents request for services which can realize their intentions.
  - agent experiences are collected and processes in InfoServices.
- **other (web) services**
  - any application with well specified input and output can be joint as a service to agentspace.
  - only simple communication interface must be implemented.

Agentspace:

a minimum infrastructure for service integration

- We do not impose any implementations details.
  - Different implementations of agentspace architecture should interoperate.
  - No system services.
- System is open and distributed.
  - Agentspace can be implemented on any transport.
  - Inside a specific agentspace; InfoService, SecretaryService, BodyService and other (web) services can be implemented by different programmers.
- **The only requirement:** they must be able to communicate in Entish, i.e. implement Entish communication interface!



What is new in our approach

- No formal ontologies (versus DAML+OIL).
  Don't ask what it means, but rather how it is used.
- Declarative (no actions) language Entish (versus DAML-S, XLANG, WSFL, …)
- Soul concept - minimum data necessary for restoring the agent process (versus weak migration)
- Agent as temporal process dedicated to a particular task (versus agent as permanent object)

Conclusion

- Entish is a simple agent communication language for web service integration.
- Formal specification of Entish is completed, and published in Proc. of ESAW’01, Springer LNAI 2203, December 2001.
- Prototype of Agentspace based on Pegaz (our MAP) already implemented.
- Testing and collecting experiences.
- Details on our web site:
  - www.ipipan.waw.pl/mas/