Service Oriented Information Systems

Jaroslav Ráček, Jiří Hřebíček
Masaryk University, Brno
Specific features of environmental IS (EIS)

The EIS should support environment related processes (some features common with business processes)

- Legislative processes or license processes (greenhouse gases licenses, environmental laws development)
- Emergency processes (actions during environmental catastrophes and their prevention)
- Education
- Information for public

⇒ Collaboration of the actors must be easy, user oriented, often permanent, and flexible
Types of actors

- **Providers.** Actors providing information (and sometimes services and actions). Necessity to use their existing (legacy) software systems
  - Can be involved in complex networks of action (emergency management, licence processes)
  - Have responsibilities, it must often permanently know each other

- **Consumers.** Actors asking for informations (services).
Such a collaboration types are possible if EIS is a service oriented software system (SOSS)
Service oriented software systems (SOSS)

- Virtual networks of autonomous software components (services) behaving like real-world services in mass services systems.
  - Must be peer-to-peer (p2p, no central service)
  - Services must be operated via command (message) queues or via more sophisticated devices (data stores)
  - Must have a specific service oriented architecture (SOA). SOA need not be web based, services need not be web services (often however are)
  - The service discusses up to now are called in the sequel *application services*
Communication of providers

- Software systems of providers should communicate with the SW systems of other providers
- Providers can use the fact, that they permanently know other providers (it does not hold for e-commerce)
- We call such systems *confederations*.
  - The details of communications can be tuned in advance and it need not use e.g. SOAP protocol; the situation in *e-commerce* is different
Confederations are quite common, examples:

- e-government, municipal SW systems, providers are information systems of offices
- Decentralized enterprises
- Systems supporting collaboration of health services
- Purchase coalition of car vendors
- Systems integrating many legacy systems and third party products
- Environmental information/control systems
- *Soft real-time (process control) system (decades of experience), some systems written in COBOL*
Confederations for small systems

- Decomposition of systems to achieve desirable software engineering properties:
  - Flexibility
  - Reusability
  - Openness
  - User oriented interfaces
  - Development effort reduction (not only due to reusability)

- **Consequence:** The necessity for EnIS to be SOSS can bring many SWING advantages
Environmental information systems are confederations

- We can (and sometimes must) use in EIS the methods, tools, and results from practice, e.g. e-government, soft real time systems
- We need not use SOAP and like standards being user unfriendly, too universal, quickly changing, and cumbersome
SOSS must have a specific service oriented architecture (SOA)
Architecture of SOSS

It is good to design portals again as services

System interface 1
consumer 1

System interface 2

Old interface of A3

2nd Summer School of Computational Biology
Ai must behave as service

- It must asynchronously accept service requests, it must be possible that there are several unsettled message requests at a time.

- There must therefore be a (hidden) FIFO data structure (queue) of requests unsettled unsettled yet.
Internal structure of gates

- Input queue
- Commands transformer
- Responses transformer
- Hidden data store

Commands: G → A1
Responses: G → A1
Application: A1
Providers must be integrated almost „without any change”

- They are typically integrated as black boxes
  - Security
  - Minimal changes inside as well as outside services, see below
- Advantageous generally (implementation details hiding)
Principles of service orientation are known and used for decades but difficult for many people to be used properly.

Soft real-time systems were the first cases of service oriented systems. They have shown their advantages (are in use for decades).

One of authors wrote such systems for manufacturing control starting from seventies.
Roots and barriers

- Roots of SOA and service oriented software systems (SOSS) are in (soft)real-time systems and properly used there with very good results for decades, some features common with systems written in COBOL.

- Contemporary SOSSs still have some features and uses some attitudes from the philosophy of process control systems. *They usually are not only information systems. They have features of control (real-time) systems. It can be the main barrier of their use.*
Process control features in IS.

Issues:

- Integration of existing systems (IS of offices, environment monitoring technologies)
- People are integral parts of processes and must be responsible for them
- Processes can ask real world processes (human being, technologies ...) for data, the data therefore need not be timely. Processes can even include real world actions.
- Processes involving several services
- Methods of prototyping, etc......
Requirements on environmental processes

There should be Process owner(s) responsible for process actions consequences

- The interfaces of providing services involved in a business process must be understood
  - by process owners,
  - by the authorized bodies willing to see process progress or performing audit,
  - by people involved in lawsuits, etc.

Modifiability of process by process owners

The interfaces of (application) services must be user oriented
Interfaces of services should be user oriented, reasons

- Process owner must understand the function of services, it is good to specify the functions via understandable service interface, it should therefore be:
  - Declarative (what rather than how),
  - Semantically rich
  - Based on formats and semantics inspired the user knowledge domains
We will introduce services of a new type

In the sequel we shall discuss so called infrastructure services behaving like enhancements of middleware
Interfaces of services should be user oriented

- Further advantages
  - Stability of the interface
  - Changes inside services tend to be invisible outside
  - Reduction of the communication channels load
Towards user oriented application services interfaces

If an application service A has an interface accessing all the function of A but in a user-unfriendly way, it is possible to develop a service *front end gate* (FEG) transforming user oriented commands into existing commands acceptable by A.
Front-end gate (FEG) is a service used as an enhancement of G.

A service can have none, one, or more FEGs.

FEG is technically and conceptually similar to portals.
SOSS with FEGs

System interface 1

Old interface of A3

System interface 2
Issues

- User oriented message formats cannot be until now standardized to the SOAP level
- To a limited extend only applicable in e-commerce where partners are looked for all over the world – proprietary standards are then difficult to apply
- Our proposals (user oriented interfaces, FEG, ...) are fully applicable for *confederations*, it is in the systemes where partners are known
Similar solutions

Enterprise Service Bus (ESB)

- IBM, BEA Systems, Sonic Software ....
- Main principles identical with the ones discussed above
- ESB is more closed,
- Expensive
- Tuned for the use inside of enterprises, where the autonomy of actors is limited
Processes in SOSS - integration and orchestration of services

- Possibilities of the implementation of processes
  - Business processes coded inside processes
    - Services often integrated as black boxes, the processes difficult to modify and made user friendly
  - Business processes coded inside messages (distribution lists)
    - Still inflexible and too dependent on changes in application services
    - User unfriendly, problems with enactment
  - Process managers (see below)
  - One or more services commanded via a portal at a time, no sequencing support,
    - Business process controlled manually

- Ways of implementation can be combined
Environmental process life cycle

- **Process model** $M$
- **Process control** $C$
- **Process actions**

**Repositories**
- Repository of models
- Repository of control structures or a set of control services called process managers (preferred)

**Key Points**
- Modeling
- Process owner
- Process enactment
- Parameters
- Emergency responses/changes
- Process commands
- Modifications based on experience
Process manager PM

- A new specific service generated during process enactment as a process control device
  - every process can have its Process manager

- The generation of PM performed on request of future process owner(s) through a portal

- The Process manager is at first empty and can be filled manually or from a process models repository and parametrized (enactment or initialization)

- The Process manager communicates with applications using their interface formats
Generation of process managers

- Portal
  - Enactment request
  - Process manager generation

- Process owner
  - Parameters interactions modifications

- Repository of process models M
  - Generation of C control structure, from M (optional)

- Process manager
  - BPEL
  - User oriented services commands
  - Application services

BPDL
Intelligent communication, data quality and data stores

- Very often the simple FIFO (queue) strategy of accepting commands must be modified
  - Message passing must be controlled by operators as it is not possible to have necessary data of acceptable quality – timely, accurate, or complete and human intelligence and knowledge must therefore be involved
  - There can be several equivalent destination services (compare grid computing)

- Solution: service having main features of data stores known from structured design and data flow diagrams
The use of a data store

Process manager

Data store

Other requests
sources

A G

A1 G

A2 G

FEG

Middleware is not shown!
Infrastructure services

- Front-end gates
- Portals
- Process managers
- Routers
- Data store like services
Snapshot of a structured SOSS

A application  FEG front-end gate  P portal
D data store  PM process manager
IfS infrastructure service (router)

2nd Summer School of Computational Biology
SW engineering advantages of service oriented systems

- **Modifiability**
  - Changes tend to be local
  - New services easily integrated
  - Easy collaboration of SOSS
  - Reused services

- **Openness**
- Easy selective reengineering
SW engineering advantages of SO 2

- Support for decentralization and restructuring
- Simplified (environmental) process reengineering
- Effective implementation of environmental (business) processes
Requirements specification of SOSS

- Explicitly stated that SOSS is to be developed
- What services will be included and how
- The application service interfaces will be specified
- Infrastructure services specified
Inverse development

- Application services (A, often legacy systems or third party systems) are integrated mainly as black boxes.
- Newly developed services are in some sense enhancements of middleware functions.
- Formely the main effort was spent in applications, not in communication services.
Principles of SO are simple and difficult at the same time

- SO is based on long term practice
- Requires specific solutions (different from standard OO ones) and new marketing policies
- Has features of process control
- Dependence on data quality (especially for environmental systems)
Conclusions 1

- The barriers are connected with the fact, that
  - SO produces systems having real-time properties
  - SO is a paradigm being so far new for many developers, a paradigm requiring new development strategies, skills, and new marketing attitudes.
Conclusions 2

- Service orientation (SO) is after object orientation and databases the third leading paradigm of SW development.

- SO offers new prospects for environmental systems, it is in fact the only known feasible implementation paradigm for environmental software systems.
Acknowledgment

Supported by project MSM0021622412
INCHEMBIOL
Thank for your attention

racek@iba.muni.cz
hrebicek@iba.muni.cz