Mag. iur. Dr. techn. Michael Sonntag

Multi agent systems as web service providers

Distributing SOAP requests to agents by redirection

17th EMCSR - 2004, Vienna, 14.4.2004

Institute for Information Processing and
Microprocessor Technology (FIM)
Johannes Kepler University Linz, Austria

E-Mail: sonntag@fim.uni-linz.ac.at
http://www.fim.uni-linz.ac.at/staff/sonntag.htm

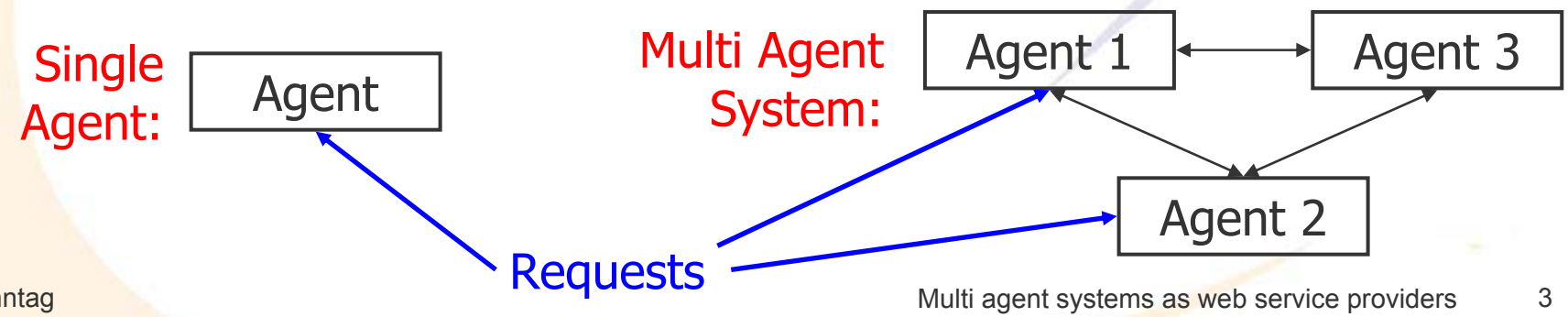
© Michael Sonntag 2004

Introduction

- Web services and agents are similar, but not equal, concepts
  - → Self contained
  - → Providing services according to strict specification
  - → Main focus on integration with other "components"
- Web services and agents can be combined in several ways
  - → Discovery: Search for matching web services
  - → Arranging chaining: Sequence of services to employ
  - → Agents as clients: Invoking web services for its own tasks
  - → Agents as servers: Providing more complex web services
- For really complex services, a system of agents (MAS) might be required
  - → An internal implementation issue (e.g. reusing agents)
  - → Should not be visible to the outside!

Single Agents vs. MAS

- Single agent: One agent providing a service
  - → Often rather similar to an application
- Multi Agent System (MAS): Several independent agents working together on a single goal
- Should a single agent be created or a MAS?
  - → Guidelines for this are rare
  - → AOSE usually provides roles, which must be aggregated
    - » Many implementations model agents as rather heavyweight
    - » Therefore several roles should be fulfilled by one



Single Agents vs. MAS: External restrictions

- Performance: Partitioning creates overhead
  - → Communication between agents (messaging, RPCs, etc.) requires much more resources, time, effort, ... compared to intra-agent communication (i.e. method calls)
  - → But better flexibility: Exchange/reuse of individual agents
  - → Enables easy (and even dynamic) exchange for a new implementation
- Security: A single agents is the only point of attack
  - → Only a single agent must be "fortified"
  - → "Internal" agents or subparts can then trust all the others
  - → No simultaneous attacks possible: hard to detect and avoid
  - → Specialisation on smaller/individual tasks with defined interface is more secure (less bugs, better testing, ...)

Single Agents vs. MAS: External restrictions

- Effort: Separate agents produce "implementation overhead"
  - → E.g. communication specification, synchronization, protocols
  - → But this is also a good idea for design (e.g. specifying communication protocols results in better understanding of tasks and their partitioning)
  - → Only rarely an issue (importance of coding in lifecycle is low!)
- Complexity of use: Single unified API is easier to understand
  - → But a huge list of methods/services is also complicated!
  - → Targeting/Finding the correct agent can be difficult
    - » Registry (MAS) and search functionality (client) required
- Commercial interests:
  - → Smaller parts are more difficult to sell and probably cheaper
  - → Customization according to actual needs better possible
    - » Configuration of agent system or even partly self-configuration

Single Agents vs. MAS

- As of this, a single huge agent would probably be best
  - → But this prevents all the other nice things:
    - » Reuse of whole agents or groups of them
    - » Easy change of implementation by replacing agents
    - » Clear separation of tasks and specialization for them
    - » ....
  - → Creates a performance bottleneck
    - » Or requires multiple threads with extensive synchronisation
  - → One step back from modularization and object-orientation
    - » AOSE should be a step forward; abstraction on higher level

- Combination of both approaches seems to be better!
  - → Allows balancing the advantages and drawbacks

Gateway Agent

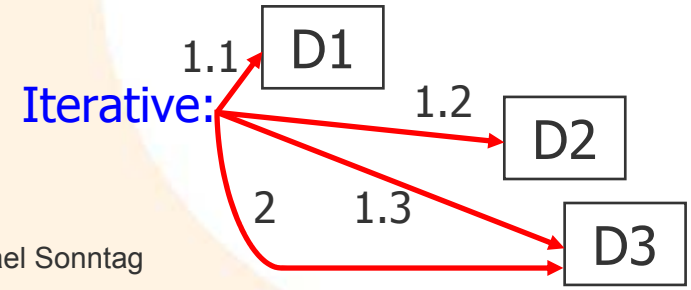
- Proposed solution: Single visible agent = Gateway
  - → Internal structure of MAS according to design
  - → Distributes requests to actual service agents by redirection
- Advantages:
  - → Single agent responsible for security, rather simple structure
  - → Unified service definition possible
  - → No public registry and search functionality needed
  - → Increased efficiency: Automatic load distribution possible
  - → Easy support for mobile agents with a fixed home base
- Problems:
  - → Performance: Everything must go through this agent
    - » But only once, and this agent only decides **who** will do the work
  - → Large and maybe complicated interface to the outside
  - → Single point of failure

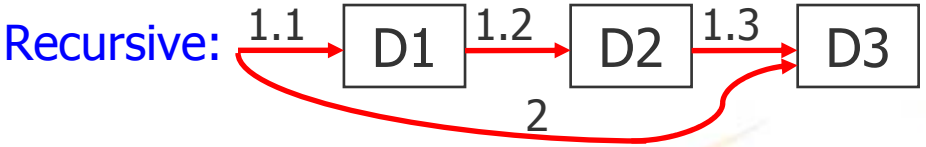
Redirects in SOAP

- SOAP (Simple Object Access Protocol) is a widespread protocol for platform-independent web services
  - → Uses mostly the HTTP protocol (e.g. SMTP also possible)
    - » Redirection available on transport level (for HTTP protocol only!)
  - → Supports intermediaries for the message/command/...
    - » Only at the sender or transparent at intermediate stage
      - – No propagation back to sender!
    - » Recursive redirects possible, but for each communication again

⇒ Support for redirection on protocol level needed

- → Allows iterative redirection as well
- → Supports recursive redirection with sender notification





Redirects in SOAP: Kinds of redirection

- Different transport or different endpoint
  - Fully automatic possible
  - → Replacing e.g. SMTP by HTTP, HTTP by SHTTP
    - » Can contain additional information: Timespan valid, address list,...
- Another method to invoke
  - Fully automatic possible
  - → Name has changed, but parameters stay the same
- Different WSDL (web service description)
  - Requires more knowledge → Agents as clients
  - → Name and parameters or other properties have changed
- Query a registry
  - Fully automatic possible
  - → That (and perhaps which) registry to query

Redirects in SOAP: Specification

- To ensure compatibility, SOAP extensions are used
  - → New header element
    - » Can be marked as optional or required ("mustUnderstand")
  - → To be used in responses (rec.) and fault messages (it.)
- Intended for SOAP RPCs (Remote Procedure Calls)
  - → Useful also for general messaging
  - → Other kinds of redirection perhaps more useful there:
    - » Which conversation it belongs to, target agent, ...
- Single XML element with several subelements
  - → Can itself be extended for custom requirements
- Several kinds of redirection can be combined
  - → Difficult for clients and perhaps confusing, but possible

Sample Implementation

- Connecting an E-Learning platform to an agent system
  - » Currently in development, SOAP already in place
  - → More complex services are provided asynchronously by agents, e.g. deriving keywords of interest from user actions
    - » Specialized agent for each user
- Gateway agent is the only one known to the platform
  - → User agents are created/found dynamically and on demand
    - » Initial request is forwarded to them
    - » Redirection for all further requests
  - → Any changes within the system are transparent to the outside
- Server modifications small: Deciding and setting the redirection target according to an internal "strategy"
- Client changes minor: Inheriting from a different base class
  - → Everything for redirection is done there automatically!

Evaluation

- Implementation of redirection has been completed
  - → Uses Apache AXIS as base & custom redirection extension
  - → Performance improvement compared to always going through the gateway agent for each request
  - → Gateway agent still some kind of bottleneck
    - » No problem, as it does no work at all, only distributing requests
- Benficial in the following cases:
  - → High volume of requests: E.g. providing a map of currently logged in students and their locations
  - → Many changes of implementation agents: Especially useful during development and extension
  - → Mobile service agents: "Portal" for agent communication
    - » Currently not used in this project!

Conclusions

- MAS should consist of two clearly separated parts
  - → Visible to the outside: Single agent probably better
  - → Internal working: System of agents for complex tasks
- Request redirection has several advantages over registries
  - → Clients need not know about and handle a registry
  - → Dynamic changes much easier
  - → Better performance: Instead of query immediately request
- SOAP has no redirection; integration through extension
  - → Server support: Deciding whether and where to redirect to
  - → Minimal client support: Can be "automated" by changing the code generated from interface definitions

Redirection allows simple and fast reconfiguration and is suitable and beneficial also for web services

Questions?

Thank you for your attention!

© Michael Sonntag 2004