Using Service-Oriented Architectures for Evolvable Software Systems

September 2006

Vincent Schmidt, Ph.D.
Research Computer Scientist
AFRL / Human Effectiveness Directorate
Vincent.Schmidt@wpafbase afs.mil
Comm 937-255-8363

Document approved for Public Release AFRL-WS-06-0282
Effective Enterprise Applications

- Must be designed to support the work environment (Work-centered)
- Easy to modify and upgrade as the work changes (Evolvable)
Perspectives of Evolvable Systems

- Self-Evolving SS
- Reconfigurable SS
- Progressive SS
- End-User Development
- Morphing Systems
- Interactive Evolutionary Computing
Progressive Software Systems

• Designed to support changes in the nature of work

• Able to dynamically gain new functionality

• Evolution keeps users from having to find alternative ways of accomplishing their work

• Users have more power and control over the software

• Spend less money and time upgrading software, more time using the software as a tool

• Some changes can be fielded remotely

• Extends lifetime of software
Progressive Software System Spectrum

- **Low risk**
  - Add visual field
  - Simple reformatting
  - No testing impact

- **Intermediate risk**
  - Combine existing data fields
  - Equations and limit comparisons
  - Some testing impact expected

- **High risk**
  - New database entries
  - Complex computations
  - Significant testing impact

Expected likelihood of occurrence

Anticipated Complexity

- Reconfigurable
- Progressive
- End-User Development
- Evolvable

Anticipated Complexity

Low

High
Technologies Implementing ESS

- RPC
- ConstellationNet
- AJAX
- Web Services
- HLA/DIS
- Java Applets
- XML
- CORBA
- Java RMI
- TBone
- MacroMedia
- XML XUL
Service-Oriented Architecture (SOA)

is about

providing functionality and data

as independent and remotely accessible

stateless services
Deploying SOA (distributed)
Deploying SOA (local)

- Enterprise Application
- Service Directory
- Web Server
- Web Services (1..n)
- Local PC
ESS Implemented as SOA
Benefits to Using SOA in ESS: System Design

SOA leverages software engineering principles:

• Enforced modular design
  – Services are stateless modules; each service provides a solution to a specific requirement

• Standardized interfaces
  – Accessed using common protocols and structure

• Language-independence
  – Able to use the most appropriate computing language for the given business logic
Benefits to Using SOA in ESS: Business Rules

SOA provides a clear and efficient path to evolving business processes:

• Service Reuse
  – Individual services are generic and stateless, can be reused and streamlined

• Ease of Upgrading
  – Modules can be replaced, extended, or ignored

• Greater control of data
  – Business logic is distributed among nodes
Benefits to Using SOA in ESS: Technology Management

SOA promotes good technology management:

• Dynamic upgrades
  – Web services can be upgraded without halting or restarting the application

• Redundancy reduces risk
  – Multiple similar services can be distributed or identified to provide the same business logic
Key Issues

• How do we do configuration management?
• How can we efficiently track evolutionary changes?
• When do we rebaseline?
• Who has authority to evolve the system?
• For SOA, how do web services changes impact each site?
Continuing Research Efforts / Tasks

• Research and document mechanisms for CM
• Testbed for SOA within AFRL/HECS
• Integrate ESS / SOA into appropriate experimental software

• Experiment with publicly available web services
• Develop and manage a local repository of services

• Continue to track and use most current SOA tools and methods
Conclusions

• The “ideas” behind ESS are the current trend of software design, allowing dynamic change of software to accommodate an ever changing nature of work

• ESS concepts have a real ability to help reduce cost/time supporting software maintenance

• ESS maximizes system versatility and useful life of software

• SOA may be a key technology supporting ESS