Case Study: Service-Oriented Architecture Using EGL at APIS IT

Robert Stanko
Division Manager, APIS IT
robert.stanko@apis-it.hr

Agenda

- Introduction
  - Croatia
  - APIS IT

- SOA at APIS IT
  - Business needs
  - Goals
  - Architecture

- Why EGL?
  - Key APIS IT challenges
  - EGL strengths
  - EGL adoption at APIS IT
  - EGL & Web Services
Croatia at a glance

Location: South-East Europe
Population: 4.5 million
Area: 56,538 km² (21,829 square miles)
Coastline: 5,835.3 km (3,626 miles) / 1,185 islands
Language: Croatian (Latin script)
Heritage: Croatian culture is based on a fourteen century-long history; six World Heritage sites and eight national parks
Currency: Kuna (1 USD = 4.8 kuna)
GNI per capita: 9,330.00 USD (2006)
Croatia – on the way to EU and NATO

- Currently, the negotiations are carried out for the accession to the EU and are expected to end in 2009
- Invitation to join NATO few months ago
- Non-permanent member to the UN Security Council 2008-2009

APIS IT

- 44 years in the process of creating key IS in the country
- understanding of customers' business processes
- the biggest and the most experienced IT integrator in the Republic of Croatia
APIS IT customers:

- Ministry of Finance
  - tax administration
  - customs administration
- Elections on all levels ([http://www.izbori.hr/](http://www.izbori.hr/))
- Government portal ([http://mojauprava.hr/](http://mojauprava.hr/))
- The City of Zagreb
  - general administration
  - public registers (citizenship, birth, marriage ...)
  - land register, GIS, address model
  - public utilities
  - finance, payroll, human resources
APIS IT goals

- Infrastructure reliability, flexibility, modularity
- Technological independence and cross-platform integration
- Legacy integration and reuse
- IT standards development
- Interoperability between heterogeneous systems
- New development process with support for business process modeling and process integration
- Ability to respond quickly to changes in customs and tax legislative

Agenda

- SOA at APIST IT
  - Business needs
  - Goals
  - APIS IT SOA Architecture
Business needs

Goals

- Define SOA reference architecture
- Define SOA governance standards
- Create SOA middleware runtime environment
- Create new development methodology (service oriented, SOMA based) and technical framework (asynchronous communication, XML processing, ESB, process integration, CICS integration)
- Create consolidated architecture which solves legacy integration and reusability issues and enables process integration on different platforms (Windows, zOS, zLinux)
- Kick-start development of major EU projects, based on SOA principles
APIS IT SOA Architecture - EU Application Integration

Agenda

- Why EGL?
  - Key APIS IT challenges
  - EGL strengths
  - EGL adoption at APIS IT
  - EGL & Web Services
APIS IT main challenges

- Time to market
- Eroding mainframe skills
- Leveraging legacy investments and adherence to new SOA

Time to market

- New projects enabling compliance to EU mandates for Croatian government
  - NCTS (New Computerized Transit System)
  - SEED (System for Exchange Excise Data)
  - EMCS
- System “online” dates NOT NEGOTIABLE
- Very compressed project schedules (less than a year!)
Eroding “mainframe” skills

- Business “services” have to run also on CICS
- Staffing expert CICS/PL/I developers are very challenging in Croatia
- New hires (fresh from school) have very different program expertise
- Young developers expect state-of-the-art development languages and tools
- Existing pool of CICS/PL/I experts busy “keeping the lights on”

Leveraging investments and adhere to SOA

- All development must fit within overall SOA
- New services must be deployable to CICS but also available as Web Services
- Significant business function already available within legacy applications
  - PL/I
  - VisualAge Generator
- Ability to reuse assets critical for time to market and costs
EGL Strengths

- APIS IT believes in Rapid Application Development technology
  - Successful track record using IBM VisualAge Generator
- EGL provides similar or better strength of VAGen and supports the modern architecture required by APIS IT
- Without an EGL-like technology skills ramp up may be too steep to meet deadlines
- EGL can also accommodate some conversion from existing legacy investments

VAGen at APIS IT – current state

- 60 + CICS DB2 applications
- 2000 + VAGen programs (TUI, called batch, main batch)
- cca 80% Customs IS
- cca 50% Tax IS
- the rest is PL/I and JAVA
1997 – why VAGen?

- productivity!
- internal (non-scientific) measurements
- 2-3 times faster development of traditional CICS applications
  - partly because of greater level of abstraction built in language itself
  - partly because of workstation modern IDE better than traditional mainframe based coding
  - partly of great debugger (ITF)
- the code generated to target environment is usually error free
- great debugger means great productivity

VAGen - ITF vs CEDF

Initial experience with EGL 6.0

- started evaluating EGL 6.0 in 2006
- great language in theory
- some problems in real life
- lack of seminars, lack of skilled consultants (in our part of the world)
- tool immaturity: unstable debugger
- dozen of PMRs opened
- IBM spoiled us with VAGen ITF (functional AND robust)
  - we are not satisfied with anything less
October 2007 – a new beginning

- part of RBD 7.1 Beta program
- Evaluating:
  - BIRT reports
  - JSF/Web development
  - Web Services Support
  - gave special attention to EGL editor AND EGL debugger
  - Experimented with VAGen conversion capability

RBD 7.1 – our experiences

- great product in real life scenarios
- finally stable & robust & fast
- well integrated & well documented
- lots of tools that simplify the development, testing and deployment of applications
- ready for serious production
What we said to IBM at the end of Beta

- “Finally a matured and stable EGL ! For us, it is version 1.0, the first version worth considering for any serious production. EGL 6.0 shouldn’t have "left the building", if you ask me, because of its notoriously unstable EGL debugger”

- “Finally a stable product for new development and a reliable upgrade path for VAGen users !”

2007 – why EGL ?

- usually phrases say something like “leveraging existing skills of business oriented developers while hiding complex J2EE and other runtime technologies to build first class applications …”
- we have business oriented developers with deep knowledge of existing business processes and applications and we will for sure exploit their skills, but …
- … we tried another scenario !
2007 – why EGL ?

- hired new developers with only (!) a basic knowledge of:
  - Java (very basic stuff)
  - some 3rd generation language (Pascal, Basic)
  - relational databases
  - Web services (in theory, but no practice)
- and no knowledge of VAGen, CICS, TSO, J2EE, WSDL, Eclipse, reporting tools or batch programs (!?)

Training

- how to train them ?
  - we don’t have years !
  - EU projects are approaching fast and in great number
  - we just can’t “on demand” produce enough Java/PL/I developers
- their training consisted of 2 parts:
  - SQL basic and advanced (self learning courses based on IBMs CF12 and CF13)
  - IBM EGL distance learning classes (EGL Foundation Training, EGL/JSF Web Development)
- after few weeks of training they were capable of developing real (not hello world) Java/J2EE applications, Web services and reports
Training

Lessons learned:
- EGL programming model is simple enough and easy to learn for developers with different backgrounds and different initial skills
- training costs are very low (compared to other technologies)
- in a very short time, developers can be trained for developing in very different technologies
- in a very short time, they are skilled enough for developing error free applications
- we can count on a very flexible pool of “business developers” and move people around projects

Training

Lessons learned:
- don’t need months (or even years) to produce a skilled professional
- developing in some older technologies, like CICS and batch programs running on mainframe, suddenly seems acceptable for new generations who have no clue about them (nor they care about mainframe !)
- IBM is offering 2 great new seminars on EGL (and they are free, for now) !
EGL & Web services

- high level of abstraction allows developers to concentrate on solving the business problem itself and not on middleware complexity when developing Web services
- becoming experts in WSDL and XML can take months, in EGL this knowledge is few clicks away
- exposing existing functionalities as Web services is very easy with EGL, e.g. existing PL/I (linked) programs or stored procedures
- it can also be done in many different ways, but EGL is very open for collaboration and integration
- since these services have been exposed through standard interface, reusing them by disparate clients (Java, .NET), is very easy

Productivity

- 10 years ago we had 2-3 times faster development
- now, with much more complex environments, it’s even faster
- internal research has been in progress, but the first results are very encouraging
- measuring: TBD
Our future requests to IBM

- Support of SOAP/JMS
  - corporate standard for intranet enterprise messaging, due to superior QoS needed for critical business applications
- Tools for automated creation of:
  - EGL records from Cobol and PL/I commareas (to allow easy mapping between EGL and Cobol/PL data structures)
  - EGL ExternalTypes from Java classes
- XML data type
  - Handling XML data is not an easy task, even using EGL XML libraries as described in EGL Tech corner

QUESTIONS