EWD:
Simplifying the VistA Web Application Architecture

Rob Tweed
M/Gateway Developments Ltd

http://www.mgateway.com
Twitter: @rtweed
A bit about me

• 1980s:
  – Lead Mumps developer at Royal Marsden Hospital, London: Hospital Information System

• Early 1990s: Management consultant at Touche Ross, London
  • Major NHS Networking project for UK Dept of Health

• Mid-1990s – present:
  – Caché/Mumps web app framework developer
  – Javascript expert
  – Internet technology industry watcher
A bit about me

• Pioneer of web development for Mumps & Caché
  – WebLink & WebLink Developer
    • Sold to InterSystems: 1996
  – Consultant for early CSP design
  – Assisted most of InterSystems' major customers worldwide on Web Application design & development
A bit about me

• Author of EWD
  – 2004 – present
  – Advanced web/ajax development framework

• EWD:
  – Tried and tested
  – Supports the largest Internet-facing Caché-based web applications
    • World's largest Caché-based web app:
      – Quest Diagnostics Care360 "EziOrder"
        » 30,000+ concurrent users
        » 500,000 test requests/day
Design Goals of EWD

• Conceptual Simplicity
• Architectural Simplicity
• Design-centric
• Speed of development
• Ease of maintenance

• Embodiment of nearly 20-years experience of developing Enterprise-grade web applications
• Addresses all areas of my dissatisfaction with IT industry's solutions
Conceptual Simplicity

Designing a dynamically generated page should be no different, conceptually, from designing a static one.
Architectural Simplicity

- As few moving parts as possible
Design-Centric

• Design, not programming, is what makes great applications
• Clear separation of Design from Programming
• Reduce programming to trivial levels
Speed of Development

• Development times should be orders of magnitude faster than normally expected
Ease of Maintenance

Software Life-Cycle Costs

Source: Digital Aggregates
The Importance of Maintenance

• 1 day of coding will cost 13 days of maintenance
  • ie less coding is good

• Fewer moving parts and layers means that one person can understand the entire application
Key features of EWD

- User Interface Tier (UI)
- Middle Tier
- Back-end Tier
UI Tier

- HTML5 + Javascript + CSS3 allows browsers to provide a very rich UI
- Which Javascript framework to use?
  - ExtJS
  - Sencha Touch
  - jQuery
  - YUI
  - Dojo
  - etc
Javascript Frameworks

• Steep learning curve
• Complex, non-intuitive code
• Describe how, not what
  – Difficult to maintain
• Programming
  – One developer's code is another's nightmare
  – Infinite degrees of freedom in coding
  – Cut & Paste Development
EWD: UI Tier

- Framework agnostic
  - "uber framework"
- Abstracts complex Javascript to XML Tags
  - Reduced learning curve
  - Simpler, much faster development
  - Easier to read, write and understand
  - Easy to maintain, even by a different developer
- Ajax "Fragments"
  - Complex UI broken down into small, easy to understand, pieces
- Custom Tags
  - Encapsulate widgets in re-usable tags
  - Avoids cut & paste development
Example EWD/ExtJS UI
Middle-tier

• Conceptual only
• Automated integration of client-side Javascript to:
  – Legacy Mumps
  – Caché/Ensemble
  – Node.js
    • As a bridgehead to:
      – EWD (for realtime web using websockets)
      – Other databases and languages
Middle Tier

- UI and Back-end are deliberately de-coupled, with "EWD Session" as middle ground

Diagram:
- EWD Page
  - `<?= myVar ?>`
- EWD Session
  - `myVar`
- EWD "Back-end"
  - `setSessionValue^%zewdAPI`
  - GT.M / Caché
  - `Pre-page script`
  - db
- UI Designer doesn't need back-end skills
- UI can be initially designed with a simple, spoofed back-end
- Back-end developer doesn't need UI skills
- Back-end can be changed without any impact on UI
Design-centric Development

• Designer is key, and stays in the loop, during and after development

• Programming is reduced to:
  – moving data between the database and Session
  – Invoking legacy functions to get/fetch/validate data
Conceptual Widget Integration

- Javascript UI widgets are mapped to EWD Session Arrays
  - Designer doesn't know or care how data is created
  - Programmer maps database information to simple, intuitive multi-dimensional arrays that map directly to JSON

<ext4:gridPanel title="Simpsons" height="200" width="700" sessionName="simpsons" />
Back-end Tier

- Mumps / Caché / Ensemble
- Physical Integration:
  - Traditional web server + passive gateway
    - IIS, Apache, nginx
    - WebLink, CSP, m_apache
  - Node.js
    - ewdGateway Module for access to Mumps / Caché / Ensemble
    - Websockets support for realtime web
    - Other Node.js modules for RDBMS and NoSQL access
Back-end Tier

• Seminal Paper, March 2010:
  – A Universal NoSQL Engine, Using a Tried and Tested Technology
    • Rob Tweed & George James
    • http://www.mgateway.com/docs/universalNoSQL.pdf
  – The Mumps database is an extremely high-performance NoSQL database that has the combined capabilities of all 4 categories of NoSQL Database:
    • Key/value
    • Document
    • Columnar
    • Graph
Back-end Tier

• Also mapped in Cache to:
  – RDBMS
  – Object Database

• EWD adds:
  – Native XML Database / DOM

• Conclusion: This is a greatly mis-understood and under-rated database technology that is ideally suited for underpinning web applications
Compare complexity of a J2EE architecture:
With the Simplicity of a Cache/Mumps web architecture

- Browser
- Web Server
- Passive gateway
- Web Application Management Environment
- Corporate Applications
- Session Storage
- Cache / Ensemble / Mumps
With EWD as the Framework:

These are all that need to be developed
EWD: Already in use with VistA

- WorldVistA web development framework of choice
- Used to develop VistACom
  - Small, agile development team
  - Key to attaining Meaningful Use Stage 1 accreditation
  - In daily use at Oroville Hospital
- Denver Acquisition & Logistics Center (DALC), a tenant unit of the Dept of Veterans Affairs (VA)
- Basis of SMART App integration for VistA
What about non-Mumps integration?

• Several alternatives:
  – Ensemble
    • Integration engine built on top of Cache
    • Adaptors for most other databases and languages
    • EWD is 100% compatible with Ensemble
  – Web Service access
    • EWD can consume external web services
      – Behave as an HTTP client
  – Node.js (server-side Javascript) opens up new alternatives
    • ewdGateway.js: Node.js gateway module for EWD
    • Use other Node.js modules for integrating other DBMS
Ensemble Example:

Browser

Web Server (Apache/IIS)

Gateway

Ensemble

Gateway

EWD Runtime

EWD Pages

EWD Session Store

Page Scripts

Adaptor

VistA

DoD
Node.js Example:

Browser

Node.js
- EWD Gateway Module
- Page Scripts (Javascript)
- Node MySQL interface
- mySQL-based system

Cache/Ensemble/GT.M
- EWD Runtime
- EWD Pages
- EWD Session Store
- Page Scripts
- VistA

EWD Pages
iEHR & EWD: Way forward?

- Try both current iEHR proposal & EWD
  - Agile, user-centered pilot development project
  - Compare results & re-assess
- Already performed a project:
  - 1 clinician (Tony Shannon)
  - 1 developer
  - EWD + ExtJS
  - 10 days
  - Outcome: successful creation of a clinical UI
- So we're not talking about an exercise that should take long to assess