

The Peloponnesian Project

Middleware Architecture For The Generation And Distribution of Entertainment Utility

© 2003

Copyrighted By Yow Keen
Mann, Released Under The GNU Free
Documentation License For The
Peloponnesian Project And In
Association with WorldForge.
No warranties implied or accepted.

Version 0.2

By K.M. Yow

Email: kmyow@yahoo.com

Contents

1. INTRODUCTION	3
1.1. Description.	3
1.2. Background.	4
1.3. Why have an Expanded Architecture?.	5
2. CONCEPT.	6
2.1. Pattern Usage.	6
2.1. Unique Features.	7
3. THE ELEMENTS	7
3.1. Software/Hardware Specification.	7
3.2..The Architecture	8
3.3. Integration	9

1.Introduction.

1.1. Description.

In the not too distant future there will be the Internet, the Commercial Internet and Transnational Conglomerate Intranets – all based on IPv4 and IPv6 with varying bandwidth, costs, security and technologies. The underlying transport medium, be it wireless, 3G networks or satellite or broadband powerlines is abstracted away by the layered nature of the TCP/IP protocol. In such an environment opportunities await and to take advantage of those opportunities you need an architecture that accounts for the shifting sands of technology.

The Internet is the internet you see it today, free, global, subject to D.O.S attacks and other security difficulties and in many places slow. The commercial internet is the same internet but with services accessible for a fee or acts as a business to customer portal but is vulnerable to the same storms. What is interesting is that open standards and new distributed programming technologies and hardware has made it possible and, learning from the difficulties of the existing Internet, viable, to create a fast, secure, low cost global reach intranet “add-ons” to the existing internet. When transaction information moves across a chain of nodes, it is only as secure as the weakest link in the chain. The same goes for delivery of copyrighted material and services.

The weakest node is currently the desktop/handheld. With technologies like palladium and laGrande, a board manufacturer can put secure bus/ports on the motherboard, a chip manufacturer can place secure chips in a cellular/wireless handset and a card manufacturer can create security cards for - access to the Global internet with DNS disruptions??? No! - rather- to act as a secure node as part of a chain of already secure nodes connecting suppliers and customers. Naturally, such security should be used on a “as needed” basis i.e can be turned on and off both for choice, cost, control, and ease of access reasons. So you can turn off security and access the Global internet as you have and you always will but have your crypto keys and passwords/credit card placed in secure caches and made inaccessible (blackbox) in non secure modes. In other words, turn the secure mode on for access to the new intranets which will be fast and disruption free and turn it off for accessing everything else (since that high level of security is redundant and leaves private information open). Future operating systems will most likely make such a procedure seamless and painless. The existing commercial internet will most likely merge some their functions with the new transnational intranets in a transparent fashion to the end user. Naturally the growth of free trade regions allows such transnational intranets to be taxed with some of the

proceeds going to upgrading the Global internet creating a business model for improving Global internet infrastructure. Ultimately the Global Internet, lower cost, riskier with more choice and variety will run side by side with secure and disruption free networks using identical technologies.

1.2. Background.

In the WorldForge Gaming System, an online paper, Bryce Harrington dreamt of creating immersive, persistent role playing worlds that are free to play, free to change and graphics rich. It has the following goals:

1. A framework for which a infinite variety of games can be built.
2. Uses GPL'd Software, Modularizes, and encourages Customization with Toolboxes and Mix Matching for stories, scripts, player level and various other game features.
3. 90 % of the game customization and administration can be performed while the game is running.
4. Be Graphically rich

Likewise I am proposing the 'Peloponnesian' Architecture because it incorporates three technologies from three huge rivals who compete aggressively with each other: Microsoft, Sun and OSS (backed by IBM). The architecture uses the latest iteration of evolutionary technologies like Sun One, .Net, and standards like UDDI, WSDL, XML to further WorldForge charter goals as well as embed it in a larger framework that allows interoperability with closed commercial systems, webservice and allows a variety of business models to be implemented and bootstrapped into viable existence. This has resulted in the following expanded goals:

5. Be commercially sound and free at the same time
6. Supports closed proprietary software
7. Bootstraps Game Development Enterprises
8. Supports Grid Computing
9. Integrates with the Changing Internet
10. Is Architecturally driven
11. Supports Product Software Model
12. Supports Service Software Model

.But an architecture is useless unless it connects up with what has been done so far as part of OSS efforts like the sourceforge projects and worldforge. The architecture should also be modular to allow anyone to mix and match the parts they desire as long as they observe the interfacing conventions (i.e. we want to avoid saying it's my way or the highway, in line with worldforge's charter to foster competition by having multiple projects). The interfacing is envisaged to be combination of SOAP/XML (across internet and firewall http port 80 and xml payload) and XML - RPC (internally between different applications on the Debian

Platform/Other OS platforms) with input/inspiration from ATLAS C++(the protocols need to support XML,UDDI,WSDL and mature further). Towards a living architecture concept, it is envisioned that new/current developers should be able to drill down via a browser based hyperlinked based tree structure from the blocky architecture diagram to the code at the leaf node.Each leaf would have one version of the source code and the different leaves can be collapsed to show only the latest leaf version.The html pages will be organized into a linux filesystem with different directory levels i.e /worldforge,/worldforge/pelopon-architecture,....., ./source-code-parser to allow interaction with CVS and ZOPE.Basically I can point my browser at the Peloponnesian architecture and the Zope server would allow me to drill down and see the source code at the node and update the latest code into the update directory with the merge being down by the coordinator who then issues it a version number.MAKE /ANT can be adapted to follow the tree to produce builds. The coordinator at the end of the day can then return the various components to the cvs repositories.

1.3. Why have an Expanded Architecture?

If you create a matrix of the licensing possibilities for server -client you get the following:

1. server - gpl , client -commercial close
2. client -gpl ,server - closed commercial
3. client - commercial ,server commercial
4. client - gpl ,server - gpl

My initial interest is 1. For example. a experimental commercial doom3 engine kept on the client communicating via SOAP/XML with a reformed stages server to implement a persistent MMORPG/Simulation in Grid Computing Arrangement ,done as part of the free doom3 mod programme i.e. i doubt any of us can afford to license the engine.Alternatively you can also form your own closed commercial engine team and retain your gpl servers for another online effort.the stages/xclient people /developers i.e people in worldforge, can reinvent themselves into consultants like ZOPE org.These are some of the possibilities.You could also incorporate JAVA to support wireless Nokia games.

Either way ,the opportunity opens up to learn about ,.NET, Sun One ,wireless and so on by working with technological trends and start building businesses ,enterprises or communities around them.

But FIRST some sort of architectural middleware framework must emerge.

2. Concept.

2.1. Pattern Usage

Middleware developers in the Closed and Open Source Communities must resolve a wide range of constraints: responsiveness, dependability, interoperability, portability, extensibility and accommodating legacy software. The severity of these constraints has resulted in the complexity of middleware patterns as opposed to those seen in smaller scale O-O applications and concurrent programming. Good Software engineering demands that patterns be created from solutions to these constraints by experienced developers who for years have struggled and incorporated into the solutions design considerations and engineering trade offs. The patterns known by names such as ProActor, Reactor, Wrapper Façade (to name just a few) have been applied to deal with I/O, threading, synchronization and event demultiplexing issues that arise in middleware development. Software like Apache, Zope, PHP are actually systems that implement these patterns both by design and by trial and error experience. Whereas a pattern separates an idea from its expression, the peloponnesian architecture now acts as a super pattern and pulls together the various patterns embodied in OS'es and Middleware both closed and open to achieve the vision of generating and distributing utility computing.

Patterns have to be independantly rediscovered and be well proven, something which the OSS process does very well culminating in the Linux platform. By pulling the various open and closed software which implement the patterns for middleware construction successfully over a period of many years, decades even in some cases, into various subsystems under the peloponnesian architecture, we have in effect the beginnings of a Super/Meta Pattern.

Like cells organizing themselves multicellularly into organ systems, a Super Pattern becomes greater than the sum of its parts and opens up new possibilities and new ways of solving complex problems and new horizons of doing business, entertainment and working.

It will be very much like standing on the shoulders of giants and looking at the world of software, networks and information in a wholly new and refreshing way.

In conclusion, pattern usage to create syngernistic subsystems comprising of operating systems, platforms, web servers and rendering architectures becomes in itself a pattern that is greater than the sum its parts.

2.2. Unique Features

Provides a clear path to achieve the 12 listed goals through a super or mega pattern.

3.The Elements.

3.1. Software/Hardware Specifications

Elements of the Proposed Architecture.

The GPL Server Platform

Debian Woody/2000 Server/Professional/Solaris
Zope/PHP
MySQL
CLIPS(AI Shell),Cyphesis AI ported to CLIPS
J2EE webservices
XML-RPC

GPL Client

SDL/OPENGL/XWindows

Commercial Client

.NET running on Windows XP and DirectX 9.0/OpenGL
2.0/CgShader compliant Graphics Card.Possible Doom3 Mod
Development?

3.2. The Architecture



