JOURNAL OF OBJECT TECHNOLOGY

Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2008**

Vol. 7, No. 1, January - February 2008

Next Generation IT – Computing In the Cloud
Life after Jurassic OO Middleware

Dave Thomas

IT CIRCA 20XX

The development of business applications using OO middleware has reached unparalleled complexity. In spite of greatly improved tools and development practices, more and more of the IT budget is wasted in maintenance rather than adding business value. The pressures for next generation applications are demanding alternative approaches to achieve increased business agility. We take a speculative look at the emerging aspects of Next Generation IT, which holds the promise to finally transition from captive hierarchical data centers and complex middleware to Cloud Computing and Agile Application Development.

APPLICATION DEVELOPMENT CHALLENGES

Application development is hard and there are few affordances to make it easier to develop and deploy major IT applications and systems. No matter how one measures today's technology its complexity is increasing and overwhelming developers.

API Surface Area = API x methods

Language Surface Area = Grammar Productions x Languages

Ways of Doing the Same Thing = Platforms x (2 to 4)

API Stability = (Middleware + Upperware + Lowerware) x 3 versions

Accidental Complexity

Developer IDE Features = Editor + Browser + Build & Test + Versioning + Process + Models x (1 to 3)

Klocs per App Delivered

Readability of Code

Locality of Application Code

The situation is further aggravated by the approaching global IT skills shortage with no solution on the horizon.

Cite this column as follows: Dave Thomas, “Next Generation IT – Computing In the Cloud Life after Jurassic OO Middleware!”, in *Journal of Object Technology*, vol. 7 no. 1 January - February 2008, pp. 27-33 http://www.jot.fm/issues/issue_2008_01/column3/

BUSINESS AGILITY – THE DRIVER

Agility enables the business to respond quickly to customers, partners and the competitors. IT is a strategic business hence IT capability. Agility implies the ability to rapidly build and configure tailored solutions which span internal and external systems.

Agility is defined by companies that your CIO or CEO envies such as Progressive Insurance, Google, Amazon, Yahoo, Sales Force, Facebook, MySpace and Apple.

Both Business IT and Software Product Vendors are frustrated with their lack of agility in both the development and deployment of applications and services. They find that even their best people using the best practices, tools and middleware can barely keep pace. They are concerned with their ability to meet the demands of Next Generation Applications and their ability to exploit the cost curves of hardware technology.

THE CHALLENGES OF NEXT GENERATION APPLICATIONS

Real-time Business

Business want to be able to develop applications in real-time and then quickly deploy them globally to any device from sensors to PCs. They also want their applications to execute in real-time. Next generation applications must sift magnitudes more data due to the data volumes from huge numbers of users, RFIDs and large numbers of transactions and interactions and event streams. Flexible, responsive reporting is much easier achieved from raw instance and is the clear alternative to aggregated data, which hides so much in the aggregation, further increasing the processing required.

Collaborative Boundary Free Applications

Applications are no longer confined to a single functional area such as Marketing or Manufacturing. They cross functional boundaries, working for both customers and competitors. We must eliminate the artificial technical and organizational barriers that impede the free flow of applications and information. Applications need to leverage inexpensive non core services from service providers. This requires selective sharing of information using role based access, which can't be supplied easily on an application by application basis. While using common services we need to provide our customers and business partners with customized information and services that match their needs down to the person and their role.

Both IT and IT Product Vendors are coming to realize that the companies they want to emulate are taking a different road with respect to infrastructure, development and delivery that offers them substantially reduced operating costs and increased agility. This realization is the primary driver for what we call the Next Generation IT.

LEAN AND AGILE – THE BEST PRACTICES

Business Agility leverages the best practices of Lean and Agile Development. Lean Software seeks to reduce software waste by ensuring that we try to build the right thing, using the right approach. Typical examples of waste include: Lack of common vocabulary, rhythm and tools; Too many meetings; Lack of Transparency: Defects in requirements, architecture, design, programs or tests; Lack of understanding/training – requirements, design, code, test; Unmanaged supplier, development or requirement risks; Unnecessary Fire Drills – feature request disguised as a critical defect; Excessive component repair vs. timely replacement; Manual Testing; Unnecessary process artifacts; Big Analysis, Architecture, Design, excessive dependencies, coupling; Gold Plating – Requirements, Code or Tests. Agile focuses on delivering quality code assets to the customer on a predictable schedule; Triage Backlogs doing what is important at each sprint; Deliver Frequently at least every quarter; Design Quality In instead of trying to test defects out; use thin slices and short spikes to reduce risk; keep designs simple and refactor them regularly to reduce the entropy; ensure that call code is owned collectively; and that "DONE" means unit, component and acceptance tested continuously.

Barriers to IT Agility – Technology, Techno-cultures and Territories

In practice tiers of technology limit the agility promised by Agile Development. The tooling and technology constraints make it more difficult to employ refactoring and continuous test. While it is possible to easily change the web tier, it is more difficult to change the middleware and even more so to change the data tier. Unfortunately, agility is further inhibited by techno-cultures and associated territorial imperatives. It is therefore difficult to reduce cycle times below 3 months for small applications and one year for major application.

PERVASIVE COMPUTING POWER – THE HARDWARE ENABLER

Next Generation IT is enabled through massive amounts of inexpensive computational resources providing an infrastructure where processors, memory, bandwidth and storage are almost “free” relative to their costs only decades before. Rumors put Google’s CPUs at 1.5 million and growing. Near term projections see 16 cores on the desk top and 100s in servers. There are Oodles of Memory and Gaggles of Disk Storage all connected by increasingly high speed bandwidth within the cloud. Special purpose GPUs promise even more cycles for hungry, calculation intensive applications from games to hedge funds.

Pervasive Connectivity is rapidly being realized across the globe with the Internet everywhere and more and more support for always on, and more importantly, occasionally disconnected devices.

We are just starting to move beyond the keyboard and screen as IO devices as increasing computer power and miniaturization enables Audio and Video Input; Two handed input; Smart Materials and sensor based Environment, Location Awareness.

"HEY! YOU! GET ON TO MY CLOUD!"*

In the wake of the proven Agility of companies like Google, Amazon, Sales Force, all of whom have lived in the clouds for close to a decade, vendors are rushing to offer their own commercial cloud solutions including Dell Cloud Computing Division, IBM on Demand and Blue Cloud and Microsoft MSN Live Cloud. In response to Google's Googleplex in a Container, Sun has announced a mobile data center with blade servers. These innovations allow clouds to be deployed globally very quickly anywhere there is power.

Clouds present the first opportunity in decades to transition from the corporate glass house to Cloud Computing. Making what were formerly grids of super computers into computation and communication utilities. Clouds promise lower cost of ownership, indeed someone else owns it, which provides fault tolerant access for anywhere from hundreds or even thousands of processors. The cloud provides non stop operation, isolating the customer from what are almost continuous HW and SW upgrades. Finally, the more services deployed in the cloud the more opportunities there are to leverage services provided by others in the same or other clouds.

SERVICE ORIENTED COMPUTING INFRASTRUCTURE - THE SOFTWARE ENABLER

SOC provides full access to the Cloud through a simple set of services, reducing the need for complex frameworks and dependence on complex, always changing vendor and corporate middleware. Cloud Services such as Amazon S3 and EC2, Google GDI, and Sales Force Applications API provide simpler "thin" service APIs (< 50) that execute closer to the underlying platform, providing support for scalable, distributed, secure computing.

The major benefit of Application Development of a small service API (thin to no class library & frameworks) is reduced complexity for application development, hence more common solutions using the same APIs. The simple SOC model allows one application team to easily and quickly leverage another application. It removes the barrier between UE Team, Bus Logic Team, Db Team, Security Team and Integration Team and places the full responsibility for the successful deployment and operations of an application on the team that built it. At Amazon, for example, application teams are expected to monitor their own applications, not rely on someone else to carry the beeper.

SUPER CRUD – FUNCTIONAL PROGRAMMING FOR THE MASSES - THE LEVERAGE

Most IT Applications are Still Essentially CRUD!

Despite the advance of technology most IT applications and indeed large portions of even computer games can be still be viewed as CRUD + Compute + Interact against federated data from heterogeneous data sources.

If one ignores middleware objects, heterogeneous data sources and fancy UI it is essentially still a simple 4GL problem

```
BEGIN MyApplication
        SELECT what user and/or application needs from
                WHERE it is stored
        THEN
         Perform more Filtering Rules and Calculations
        THEN
         UPDATE appropriate things WHERE it is needed
        THEN
    Display what is needed
END MyApplication
```

However this simple 4GL model has never been realized in the world of 3 Tier OO middleware and web information delivery. It is clear that if it could be IT applications could be produced much more quickly as they once were with 4GLs.

The benefits of a higher order functional infrastructure are many, as numerous APL, Lisp, Haskell and Smalltalk developers have experienced. In a world of massive computing resources, instead of moving mountains of data and managing the associated complexity of mappings etc. we can leverage data parallelism and send the functions to the data. Data parallelism, as shown by Connection Machine Lisp, allows one to hide the complexity of the underlying machine from the average programmer. Google of course leveraged this powerful idea in its Map-Reduce infrastructure and Yahoo and IBM are now supporting the Apache Hadop project, which is a Java look alike that can be run on top of Amazon EC2 in another Cloud. The functional API reduces the API service area. It enables application developers to think in terms of simple collections independent of their representation or storage. Most importantly it isolates them from the Cloud infrastructure. The power of functional programming in a state full world can be extended by transactional shared memory enabling safer programming by state full sinners.

Of course, this isn't new, as many special purpose higher capability language efforts have demonstrated in previous experiences using high order languages including: Relational Programming - SQL; Vector Programming – APL, NIAL, J, K; Functional Programming – Lisp, Scheme, Haskell, OCaml, Scala, F#; Set

Programming – SETL, Kleisli, and XQuery; Reactive Programming – Erlang. The availability of a simple functional infrastructure enables much simpler programs, albeit at the potential for a somewhat higher learning curve. SQL is taught in school, but it takes 2 - 3 months to become really proficient in doing complex queries. Similarly, it takes time to learn how to think in APL, Scheme or even a subset such as Google Map Reduce. However, the extra time it takes to write high quality code is a price worth paying

DO IT OURSELVES PROGRAMMING – THE EMPOWERMENT

Domain Oriented Programming and Domain Specific Languages

Given a powerful functional substrate it is natural to envision a next generation SQL, say NGFQL, which provides complete Collection Programming - Relations, Sets, Dictionaries, Lists, Arrays and compositions of them. NGFQL will enable the compact expression of powerful applications using small teams. Given a wide spectrum substrate that provides a next generation functional query and update language, DSL language designs can quickly embed DSLs in this substrate much as specialists did in Lisp, Haskell and Smalltalk. By reducing the semantic distance between the problem and the language expression we can foresee the next generation of business driven development. Business teams will include customers, domain experts and embedded Next Gen developers.

Do It Ourselves Programming – The Empowerment

During the past 30 years we have seen many examples of strong, specific programming languages/tools which provide highly productive and compact solutions to important IT problems. In many cases they enabled knowledgeable business users to author in all or in part their own applications. Examples include 4GLs – Synon, Natural, Mapper, ZIM, Cool Gen; Programming by Example – QBE/OBE/SBA, Tinker: Rule Programming and Table Programming – Business Rules, Expert Systems, Decision Tables, State Tables; Spreadsheets – Excel, The Xerox Analyst, AgentSheets; Mathematical Programming - Matlab, Mathematica, Maple; Visual Languages – Prograph, Labview etc.

Recently we see more and more examples of similarly inspired tools to support business programming. It seems very likely to us that Enterprise Mashups will be the Real SOA and that Internet Restful Services will finally provide the path to Applications Assembled from Services. IBM QEDWiki, Yahoo Pipes and Google Mashup Editor are a lot easier to use, more natural and more productive than BPM and BPEL4WS.

SUMMARY

It is painfully clear that current middleware cannot provide agility nor provide cost effective scalable commodity infrastructure. Current IT applications programming technology is too complex and too inefficient to leverage next generation infrastructures

On top of this we are facing an acute shortage of skilled application developers.

Hence we need to consider simpler alternative solutions, of which cloud computing and functional Infrastructure enabled Domain Oriented Programming seem to be the most promising on the horizon. Simple services enable agility and leverage scaleable commodity technology. Functional Services enable rapid application development and enable the service infrastructure to handle concurrency. Domain Oriented Programming enables domain specific service development on top of which Business Programming empowers business teams with embedded developers to deliver applications quickly.

About the author

Dave Thomas is cofounder/chairman of Bedarra Research Labs (www.bedarra.com), www.Online-Learning.com and the Open Augment Consortium (www.openaugment.org) and a founding director of the Agile Alliance (www.agilealliance.com). He is an adjunct research professor at Carleton University, Canada and the University of Queensland, Australia. Dave is the founder and past CEO of Object Technology International (www.oti.com) creator of the Eclipse IDE Platform, IBM VisualAge for Smalltalk, for Java, and MicroEdition for embedded systems. Contact him at dave@bedarra.com or www.davethomas.net.