Agile Architecture in Action

Jason Bloomberg
President
jason@intellyx.com
@theebizwizard

Introducing Bloomberg Agile Architecture

- Bloomberg Agile Architecture™ reinvents Enterprise Architecture (EA) to drive business agility across the organization
- New architecture paradigm
- EA technique focused on architecting agility
**Bloomberg Agile Architecture**

**Mind Map**

- Responsiveness & Resilience
- Business Agility
- Ability to Innovate
- Software Architecture
- Bloomberg Agile Architecture
- Enterprise Architecture
- Agile Manifesto
- Agile Technology / Automation
- Next-Gen SOA
- DevOps
- Cloud Computing
- Big Data

---

**BAA Technique**

- The Bloomberg Agile Architecture™ (BAA) Technique offers **a way of thinking about and doing architecture that is laser-focused on business agility as the fundamental business driver**
- Approach to Enterprise Architecture
- Not a framework or a methodology
- BAA Technique works with & simplifies TOGAF or SAFe or Zachman etc.
- Lays out a particular path through all the options that leads to greater business agility
Cross-Cutting Architecture

- Organization, process, applications, data, infrastructure
- Role for Enterprise Architecture
- Wakeup call for EAs & their organizations
- Essential for addressing silos

Architecting Human-Technology Systems

- Architecting technology isn’t enough
- You have to architect the people as well
- Organizational and process layers rightfully part of EA
- Business architecture also focuses on human behavior
- BAA streamlines and formalizes these approaches
Architecting People?

- Governance: creating, communicating, and enforcing policies
  - Apply to both human & technology behavior
  - Policy-related processes
- Automated & lightweight
- Focus on important policies
  - Security
  - Regulatory compliance
  - Software interface standards

Not a traditional approach to governance!

Agile EA & Architecture Frameworks

- Architecture frameworks (TOGAF, Zachman, DoDAF, etc.) do not adequately address business agility
  - Difficult to use properly
  - Can help achieve baseline or specific business outcomes
- Take subtractive approach
  - You need to figure out which bits to use
- BAA Technique is additive
  - Works within context of frameworks
BAA Introduces “Meta” Thinking

• Remember, we’re architecting for change
• Agile architect doesn’t think about the thing, but about how it changes
• Metaprocesses
  – Processes for creating, managing, changing processes
• Metapolicies
  – Policies for creating, managing, changing policies
  – In other words, policies for how to do governance

Avoiding the Hall of Mirrors

• If a metaprocess is a process for changing processes, how do we change metaprocesses?
• Leads to infinite repeating rat hole
  – Not really a problem, more of an illusion
  – Only takes 2 mirrors to create illusion of infinity
• Generally have only 2 layers, top one is manual
Introducing the BAA Maturity Metamodel

<table>
<thead>
<tr>
<th>Area</th>
<th>Dimension</th>
<th>Level 0</th>
<th>Level 1</th>
<th>Level 2</th>
<th>Level 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business</td>
<td>Transformation</td>
<td>None</td>
<td>IT as Service provider</td>
<td>Cloud-driven</td>
<td>Continuous</td>
</tr>
<tr>
<td>Governance</td>
<td>Hierarchical management</td>
<td>Limited Service domain management</td>
<td>Digital vs. traditional</td>
<td>Coordination of emergent behaviors</td>
<td></td>
</tr>
<tr>
<td>IT Organization</td>
<td>Siloed by division and app</td>
<td>Cross-departmental Service reuse</td>
<td>DevOps</td>
<td>Continuous Delivery</td>
<td></td>
</tr>
<tr>
<td>Governance</td>
<td>Process Automation</td>
<td>Home or scattered</td>
<td>Security &amp; Service abstraction processes automated</td>
<td>RESTful integration with GRC</td>
<td>Dynamic constraint satisfaction</td>
</tr>
<tr>
<td>Architecture</td>
<td>Rigid and artifact-driven</td>
<td>Centralised SDA Team</td>
<td>De-centralised SOA teams</td>
<td>Coordination of emergent behaviors</td>
<td></td>
</tr>
<tr>
<td>Information</td>
<td>Big Data</td>
<td>Isolated BI apps</td>
<td>Big Data pilots</td>
<td>Strategic Big Data initiatives</td>
<td>Emerging RESTful semantic media types</td>
</tr>
<tr>
<td>Syntax</td>
<td>Siloed data models</td>
<td>Fully siloed</td>
<td>Limited fixed abstractions that support some interoperability</td>
<td>MDP/conventional data model</td>
<td>Widespread use of schematic data</td>
</tr>
<tr>
<td>Technology</td>
<td>Coupling</td>
<td>Tight coupling/point-to-point integration predominant</td>
<td>Modest loose coupling based on Service contracts</td>
<td>Uniform interface for HTTP interactions</td>
<td>Dynamic Coupling</td>
</tr>
<tr>
<td></td>
<td>APIs</td>
<td>Not exist</td>
<td>Web Services</td>
<td>REST-based</td>
<td>Dynamic APIs</td>
</tr>
<tr>
<td>Platform</td>
<td>Legacy middleware</td>
<td>ESB</td>
<td>RESTful/well-defined intermediaries</td>
<td>Test-building platform</td>
<td></td>
</tr>
</tbody>
</table>

The Role of Maturity Models

- Coordinate efforts across the organization
- Balance maturity across dimensions
- Establish metrics for measuring maturity
- In some cases, compare maturity from one organization to another
  - Not appropriate for architectural maturity
Maturity Model vs. Maturity Metamodel

- Metamodel is a model for creating models
- Use the BAA Maturity Metamodel to create your own Maturity Model
  - May improve over time
  - Even your Maturity Model can mature!

Be iterative & iconoclastic

Elements of BAA Maturity Model

- Dimensions (rows)
  - Organization, Process, Information, Technology
  - Broad EA categorizations

- Levels (columns)
  - Chaotic State, First-Gen SOA/Centralized, Next-Gen SOA/Cloud-Centric, Agile Architecture
  - Agile Architecture is a journey
  - SOA is part of the story
  - You don’t have to hit all the levels all the time
Improving Maturity in Practice

- Always start with business priorities
- Maturity Model drives your Agile Digital Transformation Roadmap
- Moving up levels should represent your goals, not be goals in themselves
- Too difficult? Iterate!

Iterating a BAA Initiative

- Each iteration (sprint, for example) is well-defined, practical, and manageable
- Leverage iterative best practice from your methodology of choice
- If your problem of the day is too difficult or risky, break up your phases into smaller iterations

Any goal, no matter how difficult, is achievable this way!
Change at the Meta Level

• In fact, your entire architecture subject to change
• BAA is a meta-architecture: an architecture for building inherently dynamic architectures
• The key to thinking like a Bloomberg Agile Architect!

IT is the Business

• BAA is technology-driven
• The business of today’s enterprises is IT
  – Subtract IT from a bank, what’s left?
• Too many organizations have adversarial relationships between business stakeholders & IT
  – Part of the natural chaotic state of the enterprise
• BAA seeks to realign business & IT
  – Continuous business transformation
Continuous Business Transformation

- With BAA there is no to-be architecture
  - Not in the physical sense
- Instead, the focus of the architecture is expecting and supporting ongoing change by specifying technology that is inherently flexible
- Architects must begin at the Meta layer

Digital Strategies & Chasing Rainbows

- Established enterprise challenges:
  - Innovator’s Dilemma – don’t want to jeopardize existing revenues
  - Legacy Dilemma – how to become a next-gen, “digital enterprise” given all the legacy technology?
- Disconnect between the next-gen digital vision and the reality of today’s IT
- Spending time and money on the former without a plan for dealing with the latter is simply chasing rainbows
Enterprise as Complex Adaptive System

Where’s Our Business Agility?

- Organizations must be responsive, resilient, and innovative
- Business agility is a property of the enterprise
- People and technology can be flexible as well, but we must architect our organizations to be agile

Photo Credit: Stuart Berwick https://www.flickr.com/photos/lapleader/793507962/sizes/l
Introducing Complex Adaptive Systems

- Systems of systems that exhibit emergent properties
  - Properties of system as a whole
- Component systems can be human, technology, etc.
- Self-organizing

Enterprise as Complex Adaptive System

- Business agility is property of organization as a whole, hence an emergent property of the organization
- Enterprise is system of systems that exhibits emergent properties
  - System of people and technology
Business Agility as Emergent Property of Enterprise

- We want our organizations to be agile
- Individual subsystems (people or technology) can be flexible, but business agility is a property of the enterprise (or possibly portions of the enterprise)
- Innovativeness, resilience, responsiveness also emergent

Engineering a Complex System

- Architecture focuses on the overall design of a system
  - How all components of the system must work together to achieve business goals
- Engineering focuses on the best way to build and run a system
  - Lifecycle best practices (quality assurance, deployment, updates, etc.)
  - Compliance with architecture

Agile principle: respond to change over following a plan
At the Heart of Agile Architecture

- Think of the enterprise as a system of systems
  - Component systems are people and technology systems
- We must architect the organization so entire enterprise exhibits business agility
- We can only engineer business agility via its component systems
- Change is constant & how changes to component systems affect the enterprise is unpredictable

Complex vs. Traditional Systems

- Avoid emergent properties in traditional systems
- **Traditional system**: given requirements, build to requirements
- **Complex system**: establish initial conditions and rules for behavior of component systems, and emergence takes care of itself
  - Feedback loops are common
The Secret to Emergence

- Rules that affect behavior of component systems can be very simple
- Emergent behaviors often unpredictable
- Try something, see what happens, repeat
- Feedback loops & iterative approaches

Sound Agile to You?

Attractors for Complex Systems

- **Stable**
  - Emergent behaviors die out or never occur
  - Not applicable to enterprise
- **Cyclical**
  - System bounces between two or more well-defined states
  - Also not applicable
- **Chaotic**
  - Unpredictable, few discernable patterns
  - “Normal” state for enterprise
- **Agile**
  - Exhibiting desired emergent behaviors
Architecting Complex Systems

- Coherence of system
  - If emerging behavior strays from desired outcome, have pre-established means for bringing organization back in line
- Feedback loops
  - Gather & analyze data on emergent behavior to correlate with initial constraints and determine next course of action
- Mimics natural selection
  - Favor desired behavior and discourage undesirable behavior, and system will adapt to changed environment

Agility vs. Flexibility

- Flexibility is the ability of a component system to deal with change within a specified set of parameters
  - People can be flexible vs. rigid in their decision making
  - Software can be flexible if it is configurable to behave differently given different requirements
Does Flexibility Lead to Agility?

- If your people and your software are flexible, does that make your organization agile?
  - Necessary, perhaps, but definitely not sufficient
- Being able to change inflexible software quickly doesn’t necessarily make it flexible

Architecting Enterprise as Complex System

- Architect people as well as technology
  - Very different sets of activities
- Emergent behaviors often unpredictable
  - Continual adjustment necessary
  - Automation critical for ensuring agility
- Small changes to component system behavior may lead to significant emergent behaviors
  - Butterfly effect

Too slow and you’re not agile!
Essential Tool: Self-Organization

- Component systems of complex systems are generally self-organizing
- Traditional hierarchical organizational models limit or eliminate self-organization
  - “Tightly connected”
- “Skunk works,” startups, and DevOps organizations work best when self-organized
  - “Loosely connected”

Management challenge: when and how to facilitate self-organization
The Agile Manifesto

Manifesto for Agile Software Development
We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Context: Software Development

- Focus on small teams that crank out code
- Iterative, self-organizing, collaborative
- Focus on creating working software

Well-run Agile development means producing a lot of good software quickly
What about Architecture on Agile Projects?

- *We don’t need no stinkin’ architecture!*
- Architecture lumped in with documentation (less is more)
- Architects slow down developers
- Architects produce artifacts that aren’t Agile
  - Lead to extra work

Widespread belief that Agile and Architecture don’t mix

But we Do Need Architecture!

- **Software** architecture
  - Appropriate modularization of code
  - Selection of appropriate components & libraries
  - Integration strategy
- **Systems** architecture
  - Scalability of distributed application
  - Cloud deployment issues
  - Availability, caching, etc.
Create Minimum Viable Architecture

- All documentation, artifacts, diagrams, etc. are means to an end
- Don’t architect more than the stakeholders want
- Problems:
  - Shortsighted
  - Leads to legacy issues
  - Limits agility

Plan for Technical Debt

- **Bad** technical debt
  - Sloppy work that has to be cleaned up later
  - Rarely if ever a good idea
- **Good** technical debt
  - *Intentional* simplifications or missing functionality
  - Often specific to individual iteration/sprint
  - Well-planned resolution as within backlog/Kanban board

Good technical debt an important part of Agile development
Team Responsibility for Architecture

- Lean principle: everybody is responsible for everything
- Everybody should understand architecture
- Anybody may contribute or refactor architecture
- Adjust based upon skill sets

Everybody isn’t a competent architect!

Avoiding “Ivory Tower” Architectures

- Agile software architecture reaction to Ivory Tower architects who don’t work directly with developers
- Team may have specialists in architecture
  - Involved day-to-day with Agile team
  - Regular or constant feedback on architectural artifacts
Solicit Stakeholder Participation

- Stakeholder participation on traditional Agile projects focuses on functional requirements
- Stakeholder participation in architecture focuses on nonfunctional requirements
- Expect intractable stakeholders

The Lesson of Scrum Buts

- Statements of the form:
  - “We want to follow Scrum BUT retrospectives are too difficult SO we won’t do them.”
- Important part of Agile
  - Responding to change over following a plan!
- Problem: not always a good idea
  - “We want to be iterative BUT it’s too much trouble SO we’ll do all our development in one massive sprint.”
- How do you tell which is which?
Change-Case Agility Model

- Questions for your analysis
  - How much flexibility does your software need?
  - What features will be needed?
  - How flexible must infrastructure be?
  - How dynamic will your data be?
  - Etc.
- Must perform analysis at the meta level to avoid overbuilding

The Challenge of Updating Software

- Cascading & complicated dependencies
- Requirement for coordinated updates
- Challenge updating consumers/clients out of your control
- Backward compatibility/deprecated functionality challenges
Instant Legacy

- Difficult to upgrade
- Systems are inflexible
- Impossible to replace
- Difficult to customize
- Challenge to integrate
- Expensive over time

Extending the Agile Manifesto

- Individuals and interactions over processes and tools
  - As a system of people & technology, the enterprise is more about people
- Working software over comprehensive documentation
  - Technology must actually work to enable agility
- Customer collaboration over contract negotiation
  - Slice across the silos
- Responding to change over following a plan
  - Architect for change
Scaling Agile for the Enterprise

- Enterprise Architecture context
  - Organization, process, technology, and information
- Legacy context
  - Existing applications, middleware, systems of record
- Business focused, iterative, get stuff to work, iconoclastic

Bloomberg Agile Architecture in Practice
### BAA Initiative Phase 0: Getting Prepared

- **Understand business hot spots**
  - Problems the business is willing to spend money to solve
- **Understand as-is technology**
  - Focusing on problem areas – don’t get bogged down
- **Understand political context for transformation**
  - Transformation means changing human behavior
- **Clean up your mess!**
Clean Up Your Mess

- Organizations with legacy issues who want Cloud, mobile, etc.
  - Lack business agility
  - Desperate need of architecture
- Focus initially on establishing the appropriate separations of concerns
- Sketch out a roadmap
  - Tackle issues iteratively
- Establish baseline

Good role for TOGAF

Physical Layer

- What are your:
  - Current assets?
  - Current problems?
  - Current business drivers?
  - Political context?
BAA MM: Technology

<table>
<thead>
<tr>
<th>Coupling</th>
<th>Chaotic State</th>
<th>First-Gen SOA/Centralized</th>
<th>Next-Gen SOA/Cloud-Centric</th>
<th>Agile Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tight coupling/point-to-point integration predominant</td>
<td>Modest loose coupling based on Service contracts</td>
<td>Uniform interface for HTTP interactions</td>
<td>Dynamic Coupling</td>
</tr>
<tr>
<td>APIs</td>
<td>Rats' nest</td>
<td>Web Services</td>
<td>REST-based</td>
<td>Dynamic APIs</td>
</tr>
<tr>
<td>Platform</td>
<td>Legacy middleware</td>
<td>ESB</td>
<td>RESTful/stateless intermediaries</td>
<td>Tool-building platform</td>
</tr>
</tbody>
</table>

Meta Layer

- Top-down & bottom up at the same time
- Architect should always focus on how things change
Abstracted (Logical) Layer

<table>
<thead>
<tr>
<th>ABSTRACTED (LOGICAL)</th>
<th>Abstracted Technology (schemas, software interfaces, etc.)</th>
</tr>
</thead>
</table>

- Is SOA in place?
  - No really, what do you mean by SOA?
- How do you abstract data?
- What is role of the process abstraction in your organization?
- Look for areas of inflexibility
- Keep an eye on agility drivers
  - What should be inherently flexible

Dynamic Layer

<table>
<thead>
<tr>
<th>DYNAMIC</th>
<th>Abstract Models (dynamic schemas, dynamic APIs, etc.)</th>
</tr>
</thead>
</table>

- Core to the BAA Technique
  - Least mature part of technique at organizations overall
- Where much of your work will fall
- Expect gradual progress as your architecture matures
Assembling the Pieces of Agile Architecture

- Architecting for change requires working at higher levels of abstraction
- Abstract models resolvable to capabilities at run time
  - Dynamic coupling
  - Dynamic schemas
  - Extreme late binding

Dynamic Coupling

- Tight coupling
  - Must code all nodes at once
- Loose coupling
  - Interfaces abstract underlying code
- Dynamic coupling
  - Interfaces resolved at run time
### BAA MM: Information

<table>
<thead>
<tr>
<th></th>
<th>Chaotic State</th>
<th>First-Gen SOA/Centralized</th>
<th>Next-Gen SOA/Cloud-Centric</th>
<th>Agile Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Big Data</strong></td>
<td>Isolated BI apps</td>
<td>Big Data pilots</td>
<td>Strategic Big Data initiatives</td>
<td>Data-driven feedback across organization</td>
</tr>
<tr>
<td><strong>Semantics</strong></td>
<td>Fully siloed</td>
<td>Limited fixed abstractions that support some interoperability</td>
<td>Emerging RESTful semantic media types</td>
<td>Abstrated semantic resolution</td>
</tr>
<tr>
<td><strong>Syntax</strong></td>
<td>Siloed data models</td>
<td>MDM/canonical data model initiative in progress</td>
<td>Widespread use of schemaless data</td>
<td>Dynamic Schemas</td>
</tr>
</tbody>
</table>

### Dynamic Schemas

- **Rigid data schemas**
  - Entity-Relational Model, XML Schemas
- **Schemaless data**
  - Problem of implied schemas
- **Dynamic schemas**
  - Metadata modeled at design time, resolved at run time
The Schemaless Data Trap

- JSON often favored over XML because it is schemaless
- No fixed metadata representing data syntax or semantics
- Problem: interpretation of data falls to underlying logic

Dynamic Schemas

- WSDL files, URIs, HTTP verbs, and Internet Media Types don’t contract message semantics
- Metadata describing such semantics scattered about
  - Reintroduces tight coupling & inflexibility
- Dynamic schemas abstract all such semantic metadata in a consistent way
Resolve Dynamic Schemas at Run Time

- Abstract representations of semantic metadata
- Platform resolves metadata references at run time
- Perhaps the most advanced (and powerful) part of the BAA Technique

Late Binding

- Mechanism in which the method being called upon an object is looked up by name at runtime (Wikipedia)
  - Useful in Microsoft’s Component Object Model (COM), because compilers wouldn’t have to reference libraries at compile time
- Related to dynamic dispatch
  - Selecting which implementation of a polymorphic operation to call at runtime
How Late is your Binding?

- Early binding
  - Traditional integration
- Late binding
  - Web Service discovery
  - Content-based routing on ESBs
  - DNS resolution for RESTful endpoints
- Extreme late binding
  - Fully dynamic discovery
  - Resolve URIs & metadata references in real time

Extreme Late Binding

- Metadata-driven, late-bound schemas
- Schemas (and all other metadata) bound in real time
  - Across the organization
  - Regardless of change
- Business entities maintain separate, dynamic business contexts
Supporting Digital Transformation with BAA

Who is the Digital Professional?

- **Old days:**
  - Web programmers, graphic designers, UI/UX people, information architects

- **Today:**
  - All of the above, plus marketing/product people, Agile teams, operations people, enterprise architects, etc.

*If you touch something your customer touches, you’re a digital professional*
The Importance of the Consumer

- Consumer behavior & preferences driving digital
  - Even for B2B providers
- The Facebook effect
  - Everybody expects personalization, asynchronous interaction, & high performance

The Buck Stops with the Consumer

Omnichannel: Retail’s Digital Challenge

The customer sees only one channel
Omnichannel: Beyond Retail

- Healthcare
  - Phone/tablet/desktop/medical device integration for physicians

- Utilities, logistics, etc.
  - Field technicians

- Manufacturing
  - Factory controls

- Too many others to list!
Innovation is not a typical business process!

How do You Manage?

- Each Line of Business/Division has its own goals & business outcomes
- LoB Management drives toward optimizing those outcomes
- Maximize shareholder value/profit/revenue
- **Better-Faster-Cheaper**, then repeat
The Problem with Better-Faster-Cheaper

- Digital BFC pushes technology to its breaking point
- Technology team must focus on dealing with failure
- Resilience becomes top priority
- Requires holistic view of organization because weakest link could be anywhere

Organizing for Innovation

- Move from high connectivity to low connectivity
  - Provide flexible technology
  - Autonomous, self-organizing teams
  - Teams must cut across organizational silos
What about the Data?

- Huge quantities of Big Data about customer preferences & behavior every minute
- No end of fancy tools for analyzing those data
- For every Sony Walkman or Apple iPad, hundreds of less successful innovations
- With all these Big Data, why aren’t we better at innovating?

Data & Innovation at Cross Purposes

- Data analysis better suited for optimization than innovation
- As we leverage data to get better at marketing, we’re making decisions that stifle innovation
- Incessant march toward “better-faster-cheaper” making us more brittle, not more innovative
Mistake #1: Focusing Digital on Technology

- Easier than changing organization & culture
- Leads to “tone deaf” Digital efforts
  - Unresponsive Mobile
  - Social Media that put off or anger customers
  - Customer “service” snafus

Recipe for Agility

Bifurcation between Digital & IT

Emergence of Business Agility

Rigid & Brittle Organizations No Transformation

“Tone Deaf” Digital Efforts
Mistake #2: Shifting Digital Away from IT

- Leads to Digital efforts disconnected from IT
- ‘Legacy’ systems of record act as limitations
- No effective cooperation between Digital & IT (CDO & CIO for example)
- Leaving legacy behind moves organization away from business agility

“Demise” of Enterprise IT Department

Out:
- IT as cost center
- IT as gatekeeper, slowing things down
- IT as owner of all technology

In:
- IT as service provider
- IT as marketplace
- IT as facilitator of security, governance, and access to systems of record
Better Way to Manage

• Build cross-organizational teams
• Leverage data to understand when to optimize and when to innovate
• Embrace disruption
• Give people the tools they need and get out of their way

Thank You!

Jason Bloomberg
President, Intellyx
jason@intellyx.com
@theebizwizard
<table>
<thead>
<tr>
<th>Area</th>
<th>Dimension</th>
<th>Level 0</th>
<th>Level 1</th>
<th>Level 2</th>
<th>Level 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization</td>
<td>Business Transformation</td>
<td>None</td>
<td>IT as Service provider</td>
<td>Cloud-driven</td>
<td>Continuous</td>
</tr>
<tr>
<td></td>
<td>Organizational Governance</td>
<td>Hierarchical management</td>
<td>Limited Service domain management</td>
<td>Digital vs. traditional</td>
<td>Coordination of emergent behaviors</td>
</tr>
<tr>
<td></td>
<td>IT Organization</td>
<td>Siloed by division and app</td>
<td>Cross-departmental Service reuse</td>
<td>DevOps</td>
<td>BizDevOps</td>
</tr>
<tr>
<td></td>
<td>Governance Process</td>
<td>None or scattered</td>
<td>Security &amp; Service abstraction</td>
<td>RESTful integration with GRC tooling</td>
<td>Dynamic constraint satisfaction</td>
</tr>
<tr>
<td></td>
<td>Automation</td>
<td></td>
<td>processes automated SOA Repository</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Architecture</td>
<td>Rigid and artifact-driven</td>
<td></td>
<td>Centralized SOA Team</td>
<td>Decentralized SOA teams</td>
<td>Coordination of emergent behaviors</td>
</tr>
<tr>
<td>Information</td>
<td>Big Data</td>
<td>Isolated BI apps</td>
<td>Big Data pilots</td>
<td>Strategic Big Data initiatives</td>
<td>Data-driven feedback across organization</td>
</tr>
<tr>
<td></td>
<td>Semantics</td>
<td>Fully siloed</td>
<td>Limited fixed abstractions that support some interoperability</td>
<td>Emerging RESTful semantic media types</td>
<td>Abstracted semantic resolution</td>
</tr>
<tr>
<td></td>
<td>Syntax</td>
<td>Siloed data models</td>
<td>MDM/canonical data model initiative in progress</td>
<td>Widespread use of schemaless data</td>
<td>Dynamic Schemas</td>
</tr>
<tr>
<td>Technology</td>
<td>Coupling</td>
<td>Tight coupling/point-to-point integration predominant</td>
<td>Modest loose coupling based on Service contracts</td>
<td>Uniform interface for HTTP interactions</td>
<td>Dynamic Coupling</td>
</tr>
<tr>
<td></td>
<td>APIs</td>
<td>Rats’ nest</td>
<td>Web Services</td>
<td>REST-based</td>
<td>Dynamic APIs</td>
</tr>
<tr>
<td></td>
<td>Platform</td>
<td>Legacy middleware</td>
<td>ESB</td>
<td>RESTful/stateless intermediaries</td>
<td>Tool-building platform</td>
</tr>
</tbody>
</table>

Bloomberg Agile Architecture™ Maturity Metamodel

Copyright © Intellyx LLC. Permission to distribute granted as long as no changes are made and this notice remains. For more information email agility@intellyx.com.